

Министерство науки и высшего образования Российской Федерации  
федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«Кузбасский государственный технический университет  
имени Т. Ф. Горбачева»

Институт профессионального образования  
Кафедра информатики и информационных систем

Константин Андреевич Кулиничев

## **ПРОГРАММНЫЕ РЕШЕНИЯ ДЛЯ БИЗНЕСА НА ПЛАТФОРМЕ 1С**

Методические материалы к практическим занятиям,  
лабораторным и самостоятельным работам

Рекомендовано цикловой методической комиссией  
специальности СПО 09.02.07 Информационные системы  
и программирование в качестве электронного издания  
для использования в образовательном процессе

Кемерово 2024

Рецензенты: К. В. Ерошевич – преподаватель кафедры информатики и информационных систем ИПО ФГБОУ ВО «Кузбасский государственный технический университет имени Т. Ф. Горбачева»

**Кулиничев, К.А. Программные решения для бизнеса на платформе 1С:** методические материалы к практическим занятиям, лабораторным и самостоятельным работам для обучающихся специальности СПО 09.02.07 «Информационные системы и программирование» / сост. К. А. Кулиничев, Кузбасский государственный технический университет имени Т. Ф. Горбачева. – Кемерово, 2024. – Текст: электронный.

Методические материалы к практическим занятиям, лабораторным и самостоятельным работам для дисциплины «Программные решения для бизнеса на платформе 1С» описывают содержание практических, лабораторных занятий и самостоятельной работы, перечень вопросов на защиту выполненных работ.

© Кузбасский государственный  
технический университет  
имени Т. Ф. Горбачева, 2024  
© Кулиничев К. А.,  
составление, 2024

## ОГЛАВЛЕНИЕ

|   |     |
|---|-----|
| ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №1. МОДЕЛИРОВАНИЕ ПРОЦЕССОВ<br>НА ОСНОВЕ СТАНДАРТА IDEF0. ....                               | 5   |
| ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №2. ПРОВЕДЕНИЕ СТОИМОСТНОГО<br>АНАЛИЗА ПРОЦЕССА .....  | 57  |
| ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №3. РАБОТА С КОМАНДНЫМ<br>ИНТЕРФЕЙСОМ СИСТЕМЫ 1С ПРЕДПРИЯТИЕ .....                           | 67  |
| ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №4. РАБОТА С ПОЛЬЗОВАТЕЛЯМИ<br>СИСТЕМЫ.....  | 82  |
| ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №5. РАБОТА С ПРИКЛАДНЫМИ<br>ОБЪЕКТАМИ ССЫЛОЧНОГО ВИДА .....                                  | 98  |
| ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №6. ЗАПРОСЫ К ТАБЛИЦАМ-<br>ИСТОЧНИКАМ ДАННЫХ ПРИКЛАДНЫХ ОБЪЕКТОВ.<br>УСТАНОВКА ОТБОРОВ ..... | 105 |
| ЛАБОРАТОРНАЯ РАБОТА №1. УПОРЯДОЧИВАНИЕ.<br>ПОЛУЧЕНИЕ СВОДНОЙ ИНФОРМАЦИИ.....                                      | 113 |
| ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №7. РАБОТА С ОБЩИМИ ФОРМАМИ<br>.....   | 121 |
| ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №8. РАБОТА С УПРАВЛЯЕМЫМИ<br>ФОРМАМИ.....  | 135 |
| ЛАБОРАТОРНАЯ РАБОТА №2. ИСПОЛЬЗОВАНИЕ МАКЕТА<br>ПЕЧАТНОЙ ФОРМЫ .....  | 147 |
| ЛАБОРАТОРНАЯ РАБОТА №3. РАБОТА С СИСТЕМОЙ<br>КОМПОНОВКИ ДАННЫХ.....   | 162 |
| ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №9. РАБОТА С ПЕРЕЧИСЛЕНИЯМИ<br>И СПРАВОЧНИКАМИ НА ОСНОВЕ ОБЪЕКТНОЙ МОДЕЛИ.....               | 176 |

|  |     |
|--|-----|
| ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №10. РАБОТА С ДАННЫМИ НА<br>ОСНОВЕ ТАБЛИЧНОЙ МОДЕЛИ. ЗАПРОСЫ И ОБРАБОТКИ В<br>СРЕДЕ 1С ПРЕДПРИЯТИЕ..... | 188 |
| ЛАБОРАТОРНАЯ РАБОТА №4. РАБОТА С ДОКУМЕНТАМИ В<br>СИСТЕМЕ 1С ПРЕДПРИЯТИЕ .....   | 200 |
| ЛАБОРАТОРНАЯ РАБОТА №5. РАБОТА С РЕГИСТРАМИ<br>ОПЕРАТИВНОГО УЧЁТА.РЕГИСТРЫ НАКОПЛЕНИЯ. РЕГИСТРЫ<br>СВЕДЕНИЙ .....            | 213 |
| ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №11. ИСПОЛЬЗОВАНИЕ<br>МЕХАНИЗМА ОПОВЕЩЕНИЯ.....   | 224 |
| ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №12. ФОРМИРОВАНИЕ<br>ПРЕДУПРЕЖДЕНИЙ И ВОПРОСОВ ПОЛЬЗОВАТЕЛЮ .....                                       | 233 |
| ЛАБОРАТОРНАЯ РАБОТА №6. РАБОТА С ОТЧЁТАМИ .....  | 246 |
| СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ .....  | 258 |
| УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ..  | 259 |

## **ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №1. МОДЕЛИРОВАНИЕ ПРОЦЕССОВ НА ОСНОВЕ СТАНДАРТА IDEF0.**

### **Определение свойств проекта и модели**

В качестве учебного примера предметной области рассмотрим организацию работы библиотеки в традиционном представлении. Рассмотрим этот пример в упрощенном виде, имея в виду то интуитивное представление о работе библиотеки, которое имеется у студентов.

Допустим, что работа библиотеки по обслуживанию абонентов осуществляется по такому сценарию.

Приобретенные библиотекой книги размещаются в книгохранилище. При этом в справочно-библиографическом отделе регистрируется их библиографические данные.

Абонент на абонементе обращается в справочно-библиографический отдел с запросом о наличии книги в библиотеке и в ответ получает справку.

При наличии книги в библиотеке абонент оформляет заявку в книгохранилище. Затребованная книга при условии наличия ее экземпляра в книгохранилище, пересылается на абонемент и выдается абоненту.

Выполним моделирование работы библиотеки на основе методологии IDEF0, используя в качестве инструментальной среды Ramus Educational.

Программа Ramus имеет сравнительно простой, интуитивно понятный интерфейс пользователя. После ее запуска пользователю будет предложено или создать новый файл проекта или открыть уже готовый (Рисунок 1.1).

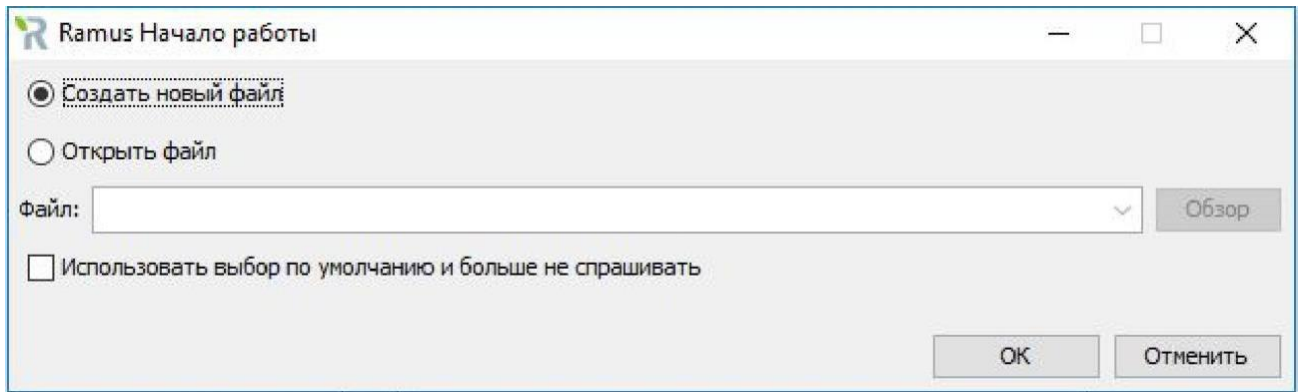


Рисунок 1.1 - Диалог начала работы

После выбора «Создать новый файл» и нажатия экранной кнопки «Ок» осуществляется запуск мастера проекта. Мастер поведет пользователя по некоторой процедуре, состоящей из пяти шагов.

На первом шаге предлагается указать автора проекта, название проекта и название модели, а также выбрать нотацию модели IDEF0 или DFD (Рисунок 1.2).

На втором шаге предлагается указать название организации, где будет использован данный проект (Рисунок 1.3).

На третьем шаге предлагается привести описание проекта. В описании проекта можно указать его краткую характеристику (Рисунок 1.4).

На четвертом шаге предлагается создать так называемые классификаторы. Классификаторы используются для упорядочения информации об объектах таких, как документы, персонал, оборудование и т.д.

В виду простоты нашей учебной предметной области классификаторы можно не создавать (Рисунок 1.5).

На пятом, заключительном шаге, предлагается выбрать те из созданных классификаторов, элементы которых будут содержаться в перечне собственников процессов (этот шаг также можно пропустить) (Рисунок 1.6).

Мастер "Свойства проекта"

Укажите автора и название проекта.

Автор:  
Иванов И.И.

Название проекта:  
Учебный проект

Название модели:  
Библиотека

☒ IDEF0 ☐ DFD

Этап 1 из 5

Назад Дальше Отменить Окончить

Рисунок 1.2 - Диалог настройки свойств проекта (шаг 1)

Мастер "Свойства проекта"

Укажите предприятие, организацию, подразделение или другое место где используется проект.

Используется в:  
УрФУ в учебном процессе

Этап 2 из 5

Назад Дальше Отменить Окончить

Рисунок 1.3 - Диалог настройки свойств проекта (шаг 2)

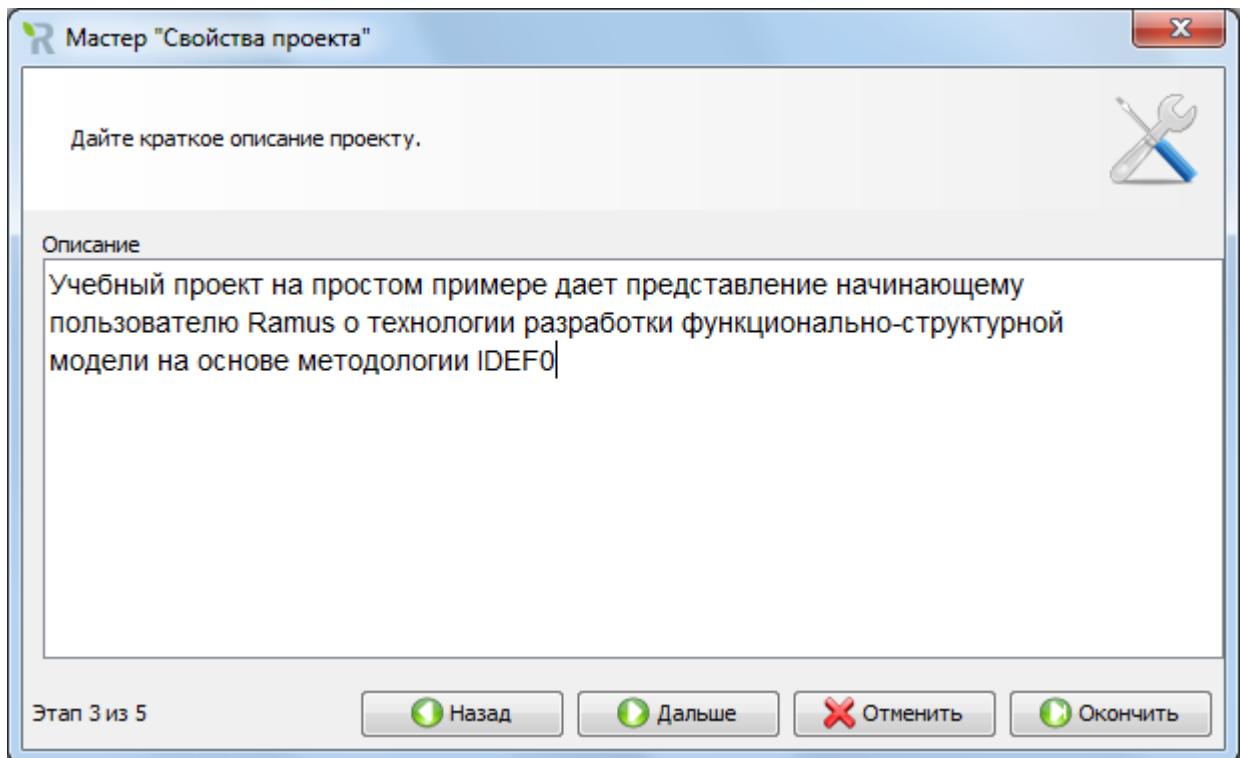


Рисунок 1.4 - Диалог настройки свойств проекта (шаг 3)

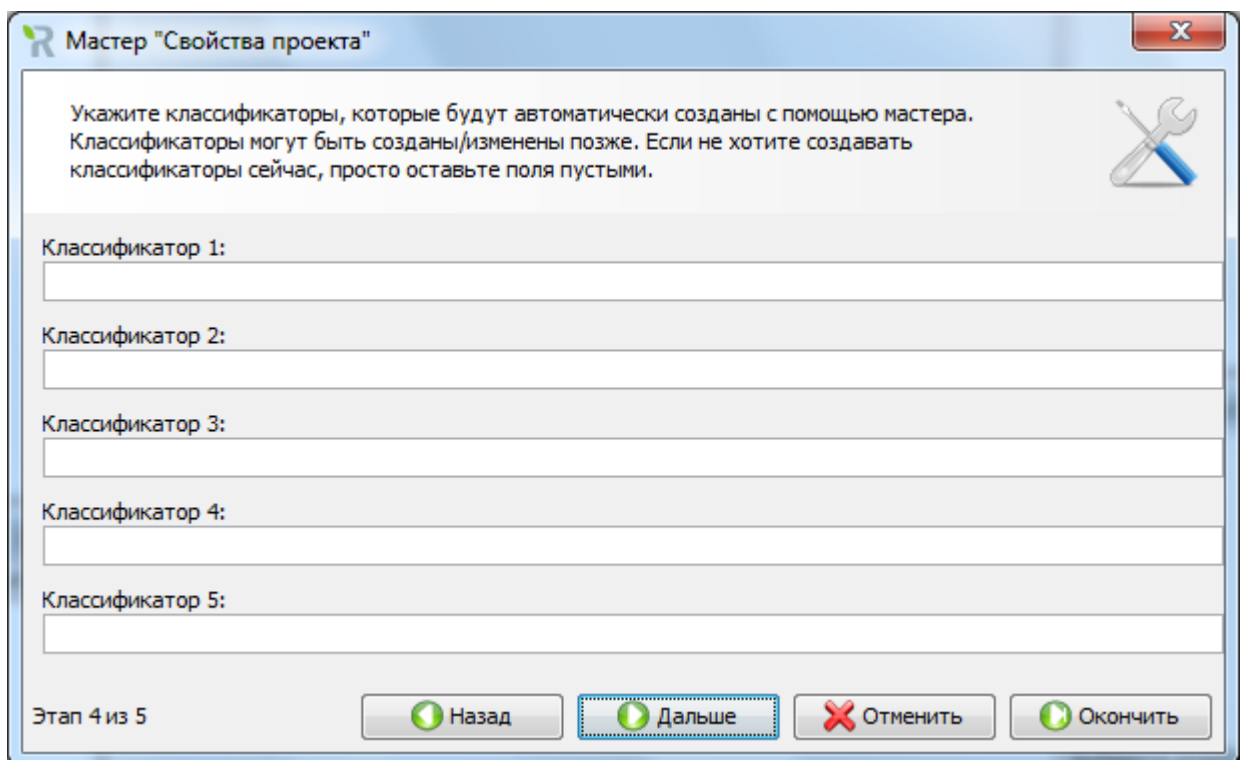


Рисунок 1.5 - Диалог настройки свойств проекта (шаг 4)



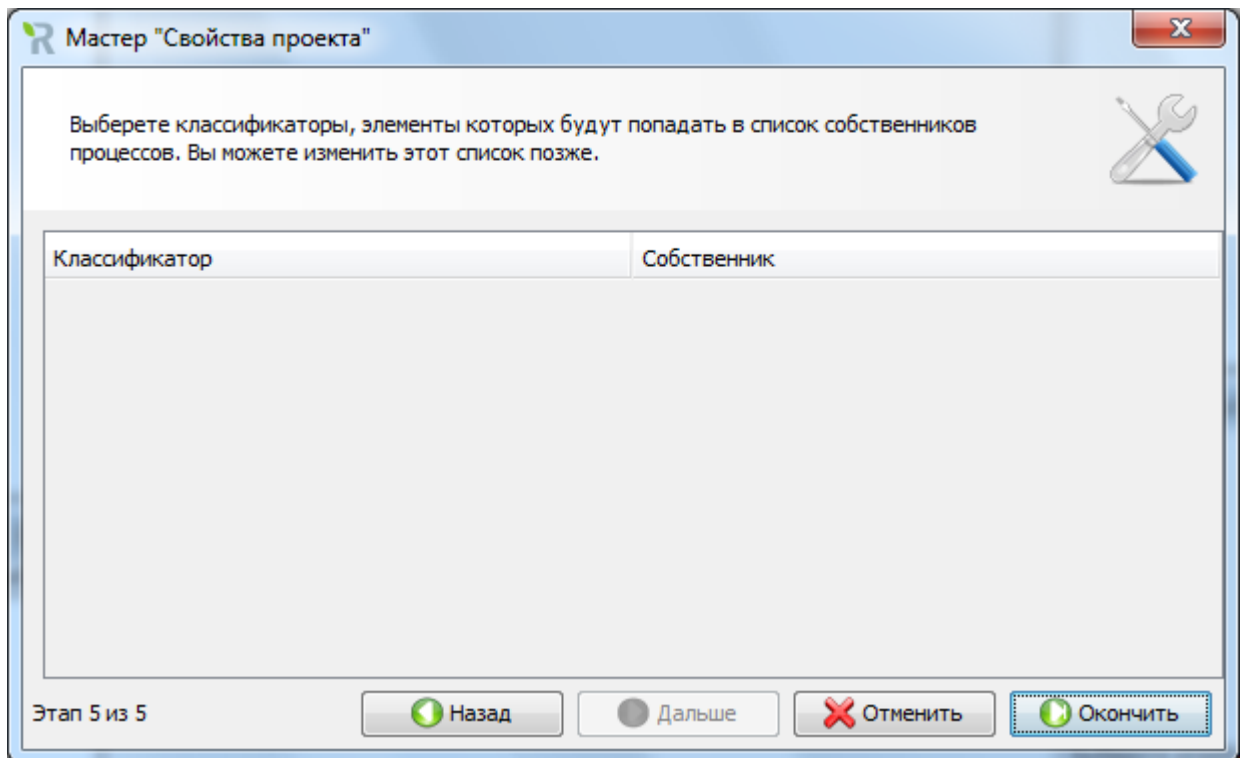


Рисунок 1.6 - Диалог настройки свойств проекта (шаг 5)

Работа с мастером заканчивается по нажатию кнопки «Окончить».

По завершении работы мастера пользователь переводится в режим «Диаграммы» и может приступить к построению диаграмм модели.

Диаграммы оформляются в рабочем пространстве шаблона, в котором указаны свойства модели и ее статус (Рисунок 1.7).

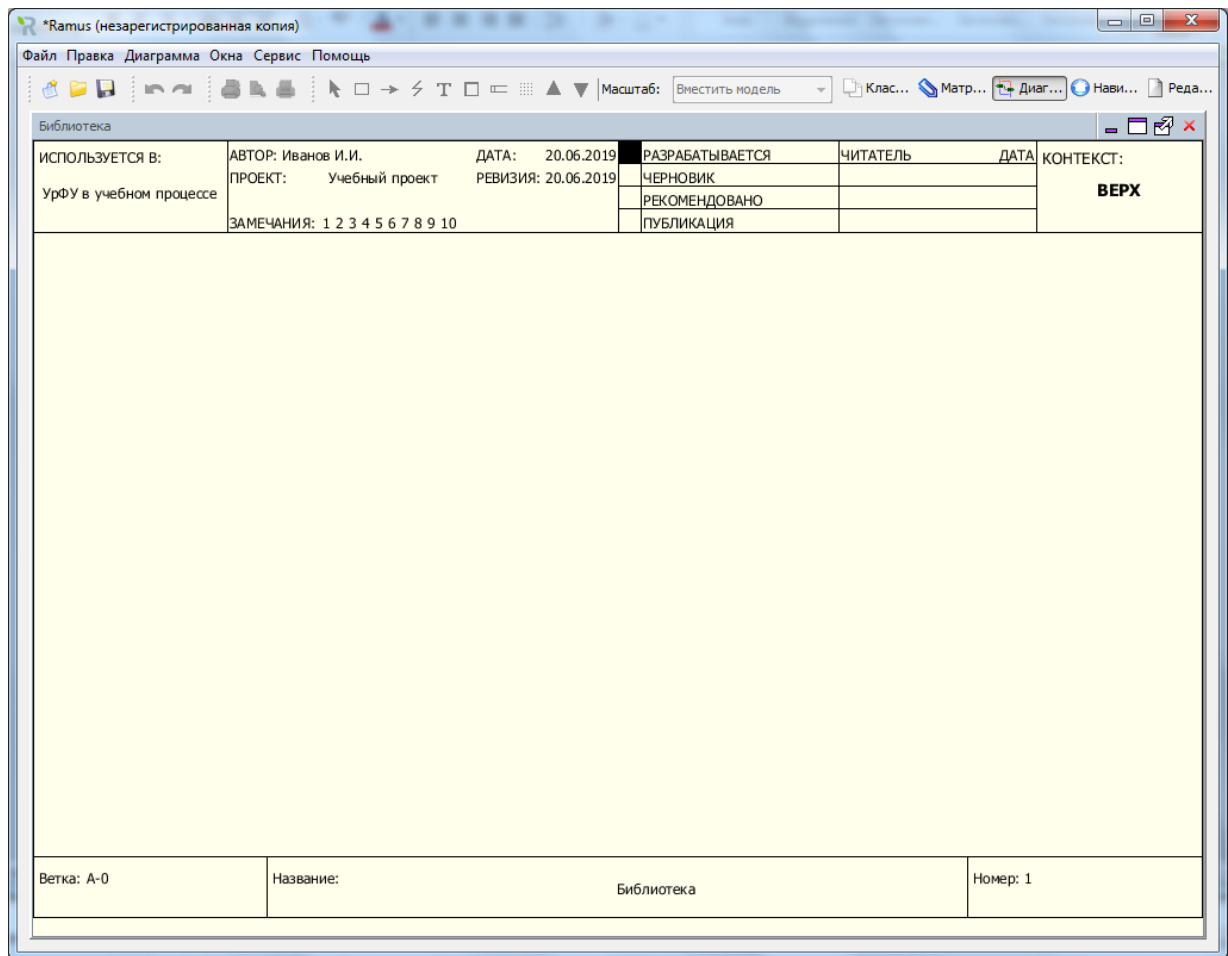


Рисунок 1.7 - Окно режима диаграммы

Построение модели осуществляется при помощи панели инструментов (Рисунок 1.8):



Рисунок 1.8 - Панель инструментов

**Режим курсора (редактирования)** – основной режим работы с диаграммами. В этом режиме осуществляется манипулирование объектами диаграммы – функциональными блоками и стрелками, в том числе: перемещение, изменение размеров, редактирование свойств и другие.

**Режим добавления функциональных блоков** – этот режим позволяет добавлять на диаграмме новые блоки при помощи левой клавиши мыши.

**Режим работы со стрелками** – дает возможность размещать на диаграмме стрелки и соединять их с соответствующими блоками.

**Режим размещения тильд (выносок)** – может использоваться для соединения стрелок с их именами. Выноски являются необязательными элементами диаграмм, но упрощают их чтение.

**Режим добавления текста** – позволяет размещать в любом месте диаграммы произвольный текст, в частности включать комментарии.

**Сетка** – является факультативным режимом и может использоваться для выравнивания функциональных блоков и текстовых областей.

**Переход к родительской диаграмме** – позволяет перемещаться по иерархии связанных диаграмм от дочерней к родительской.

**Переход к дочерним диаграммам** – позволяет перемещаться от текущего функционального блока к диаграмме его декомпозиции.

Перед построением модели по команде **Сервис | Свойства программы | Диаграмма** можно настроить свойства ее структурных элементов, такие как цвета полотна, блоков, стрелок, а также шрифты подписей блоков, стрелок и текста, например, так, как показано на Рисунке 1.9. Цвета оставьте такими, как они заданы по умолчанию, а шрифты - какие указаны на Рисунке 1.9.

При желании в последствии эти свойства можно будет переопределить в режиме редактирования элементов.

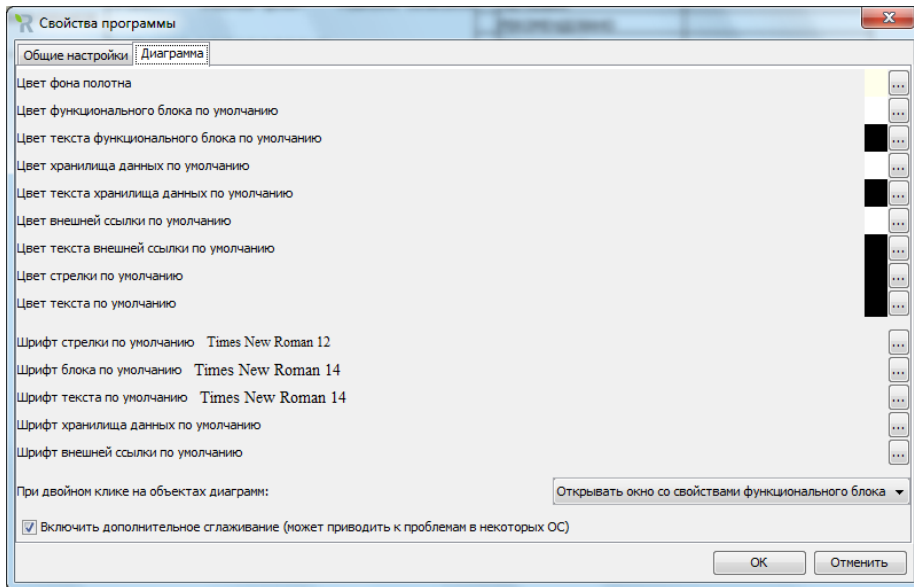


Рисунок 1.9 - Настройка свойств программы

Построение модели начинается с определения контекста, который включает в себя определение цели моделирования, границ предметной области и точки зрения профессионала, который видит предметную область в нужном для достижения цели аспекте.

Для настройки свойств модели используется опция «Свойства модели» меню «Диаграмма» (Рисунок 1.10).

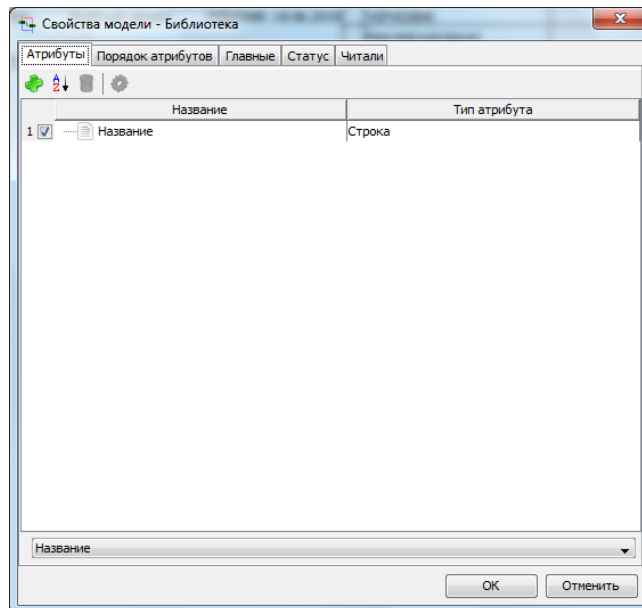


Рисунок 1.10 - Диалог настройки свойств модели

В окне настройки свойств модели предлагается:

1. Указать список атрибутов для функциональных блоков и порядок их отображения.
2. Среди атрибутов всегда присутствует по умолчанию атрибут «Название», которому присваивается значение имени активного элемента. Другие атрибуты могут быть добавлены дополнительно.
3. Указать название проекта, для которого создается модель.
4. Указать автора проекта и место его использования.
5. Указать букву модели.
6. Дать лаконичное описание проекта.
7. Указать статус диаграммы.
8. Указать читателей модели.

Свойства с 1 по 4, а также 6 определяются по умолчанию на основе спецификаций проекта, заданных в начале работы.

Свойство 5 предполагает задать букву, которая будет использоваться в качестве префикса номеров функциональных блоков.

Статус диаграммы (свойство 7) будет изменяться в процессе ее экспертизы.

В списке читателей модели (свойство 8) указываются фамилии экспертов, принимавших участие в ее обсуждении, а также даты экспертиз.

Окно «Свойства модели» можно также открыть с помощью контекстного меню, вызываемого щелчком правой клавиши мыши по заголовку активной диаграммы.

Несколько подробнее об атрибутах активных элементов диаграммы.

Атрибуты представляют собой свойства активных элементов. Доступ к атрибутам возможен разными путями. Например, как уже упоминалось, через «Свойства модели» меню «Диаграмма». Возможен также доступ через окно «Набор атрибутов» опции «Показать окно» меню «Окна». В окне «Набор атрибутов» имеется панель инструментов со следующими экранными кнопками:



создать атрибут;



сортировать по названию;



удалить атрибут;



свойства атрибута.

Диалог «Набор атрибутов» можно вызвать также для активного элемента диаграммы при помощи опции «Редактирование активного элемента» контекстного меню, инициируемого щелчком правой клавиши мыши по активному элементу диаграммы.

Создание атрибутов выполняется через специальное диалоговое окно (Рисунок 1.11), в котором следует указать имя атрибута и его тип данных. Для спецификации атрибутов используются следующие типы данных:

**Строка** - атрибуты такого типа могут содержать обычный текст.

**Число** - атрибуты такого типа могут содержать числа (в том числе и дробные). Такой атрибут имеет точность примерно в 16 знаков и экспоненциальное значение в диапазоне от -308 до 324

**Целое число** - атрибуты такого типа могут содержать только целые числа в диапазоне от -9223372036854775808 до 9223372036854775807.

**Дата** - атрибуты такого типа могут содержать значение даты в формате ДД.ММ.ГГГГ.

**Деньги** - атрибуты такого типа могут содержать числа (аналогично типу атрибута «Число»), но при выводе такого атрибута будет указан вид валюты и количество знаков после запятой в соответствии с системными настройками.

**Элемент классификатора** - атрибуты такого типа могут содержать одно значение атрибута другого классификатора. Пример использования: атрибут «Ответственный за заполнение», в котором указывается значение атрибута «Должность» классификатора «Персонал».

**Набор вариантов** - атрибуты такого типа могут содержать значение, которое выбирается из фиксированного списка текстовых вариантов. Пример использования: атрибут «Происхождение документа» со списком вариантов: внешний; внутренний. Следует отметить, что данный атрибут не рекомендуется использовать, если перечень вариантов обширен, и есть возможность использовать атрибут «Элемент классификатора», который заполняется из


специального классификатора, в котором и будет содержаться перечень вариантов.

**Присоединенный файл** - атрибуты такого типа могут содержать ссылку на любой файл. Пример использования: атрибут «Образец», который содержит ссылки на файлы образцов документов.

**Описание** - атрибуты такого типа могут содержать форматированное, текстовое описание любой длины. Для редактирования такого типа атрибута можно использовать внешний редактор (например, Microsoft® Word).

**Таблица** - атрибуты такого типа могут содержать табличные данные произвольной формы. Столбцы таблицы формируются из доступных в проекте атрибутов любого типа.

Создайте для модели атрибуты: цель, границы, точка зрения, например по команде **Диаграмма | Свойства модели**:

1. В диалоге «Набор атрибутов» по нажатию кнопки «Создать атрибут»  создайте атрибут «Цель», который характеризует цель моделирования, как данное строкового типа (Рисунок 1.11).

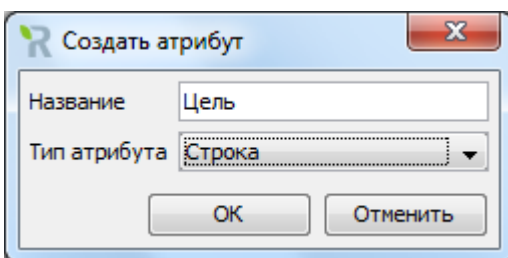


Рисунок 1.11 - Диалог создания атрибута "Цель"

2. Создайте также атрибут «Границы», который определяет, что входит в предметную область, а что находится за ее пределами. Присвойте этому атрибуту строковый тип данных (Рисунок 1.12).

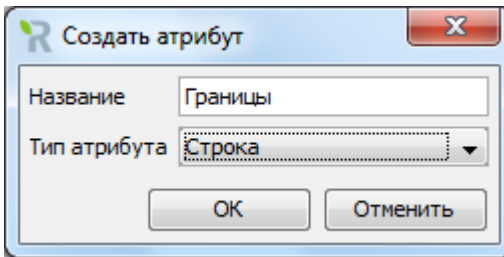


Рисунок 1.12 - Диалог создания атрибута "Границы"

3. Создайте атрибут «Точка зрения», обозначающий позицию, с которой следует рассматривать предметную область. Присвойте этому атрибуту строковый тип данных (Рисунок 1.13).

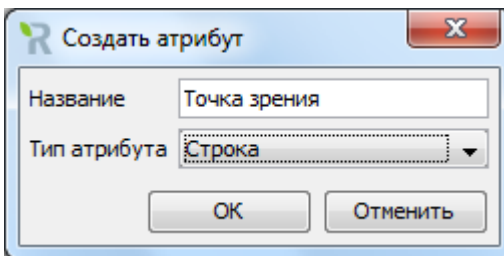


Рисунок 1.13 - Диалог создания атрибута «Точка зрения»

Контекстное описание модели (цель, границы, точка зрения) можно явно указать как **значения соответствующих атрибутов** для функционального блока контекстной диаграммы или **разместить в виде текста** на контекстной диаграмме.

4. Для функциональных блоков кроме атрибута «Название» будет полезным атрибут «Описание» семантику того действия, которое в виде лаконичного текста представлено в названии функционального блока. В диалоге свойств модели для функциональных блоков добавьте атрибут «Описание» строкового типа (Рисунок 1.14).



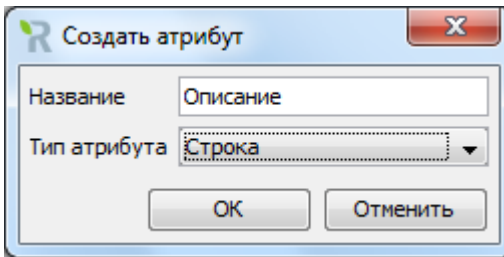


Рисунок 1.14 - Диалог создания атрибута «Описание»

По завершении создания атрибутов «включите» соответствующие флажки на вкладке «Атрибуты» (Рисунок 1.15).

На вкладке «Порядок атрибутов» можно изменить порядок следования атрибутов (Рисунок 1.16).

На вкладке «Главные» можно наблюдать, а при желании и изменить свойства модели, заданные при спецификации проекта (Рисунок 1.17). В окно «Буква модели» введите латинскую А (Activity – работа), которая традиционно будет играть роль префикса номеров функциональных боков модели.

На вкладке «Статус» задается статус диаграммы модели, который она будет принимать в ходе ее экспертизы (Рисунок 1.18).

На вкладке «Читали» можно задать список экспертов, которые принимали участие в обсуждении модели (Рисунок 1.19).

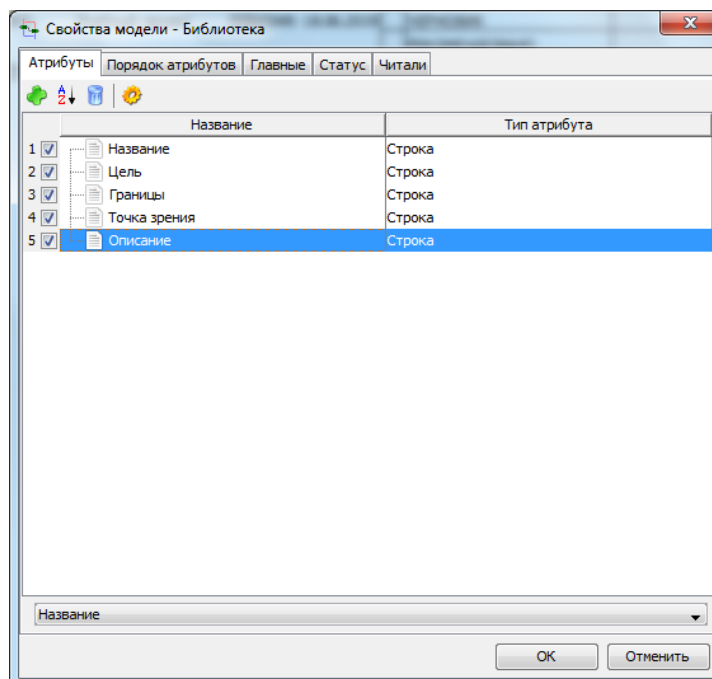


Рисунок 1.15 - Диалог инициации атрибутов модели

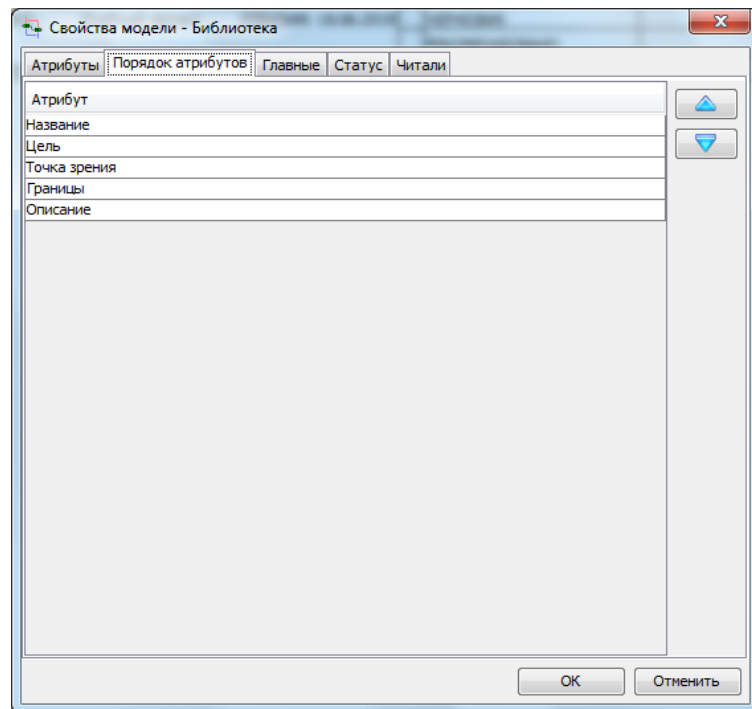


Рисунок 1.16 - Диалог изменения порядка атрибутов

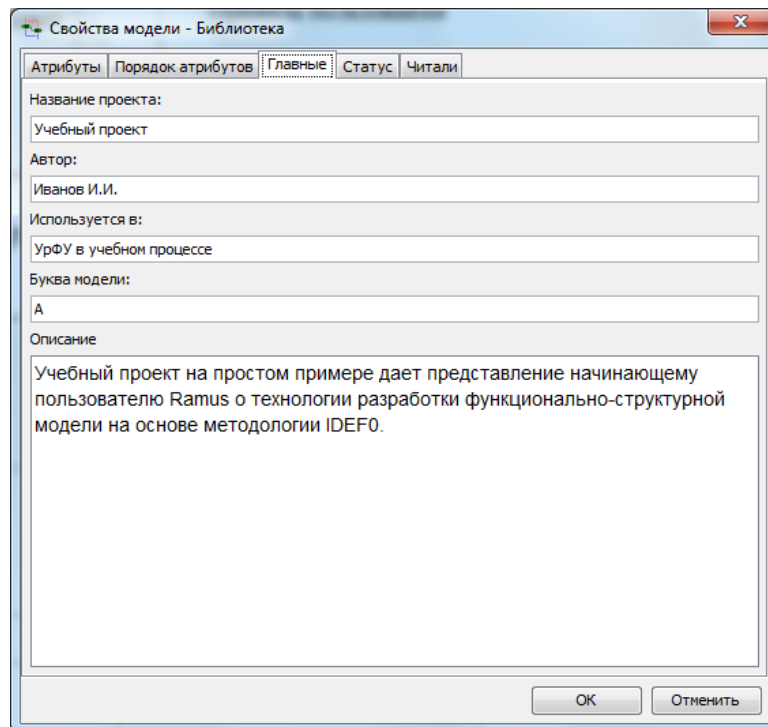


Рисунок 1.17 - Диалог настройки свойств модели

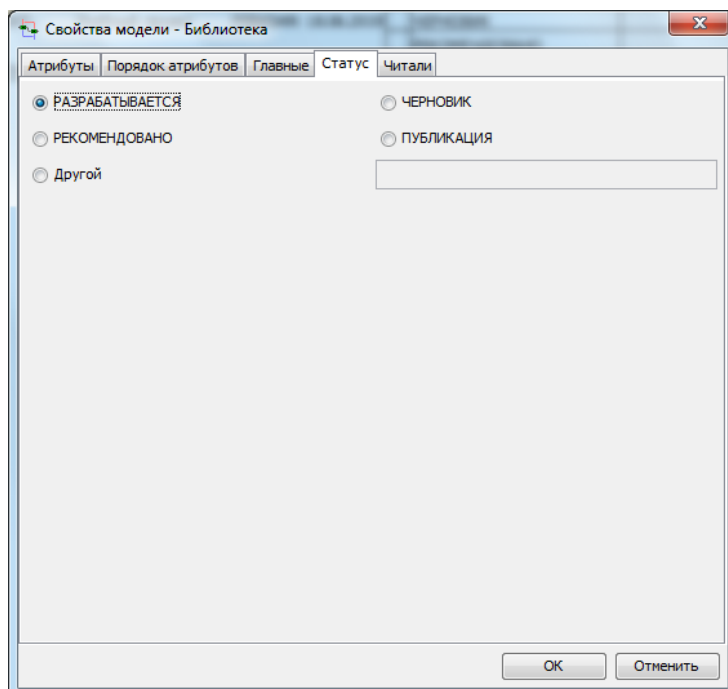


Рисунок 1.18 - Диалог настройки статуса модели

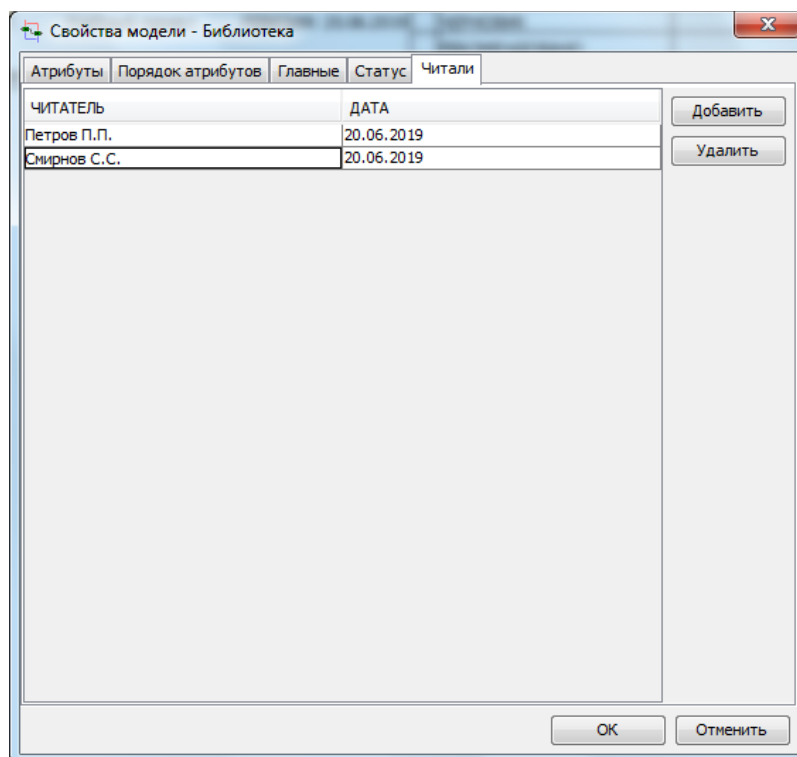


Рисунок 1.19 - Диалог настройки списка читателей модели

## Построение контекстной диаграммы

Моделирование начинается с построения контекстной диаграммы, которая дает самое общее представление о моделируемой предметной области.

Функциональному блоку контекстной диаграммы должно быть присвоено содержательное имя и другие свойства. Для этого следует

1. На панели инструментов выбрать пиктограмму функционального блока и при помощи мыши задать его положение на диаграмме.
2. Щелчком правой клавиши мыши по шаблону блока, вызвать контекстное меню и выбрать опцию «Редактировать активный элемент».
3. В соответствии с каноном IDEF0 имя (название) функциональному блоку должно быть присвоено в форме глагола. Ramus не контролирует соблюдение синтаксиса русского языка, поэтому в названии функциональных блоков часто используются не глаголы, а отглагольные существительные, например «Работа библиотеки» (Рисунок 1.20).

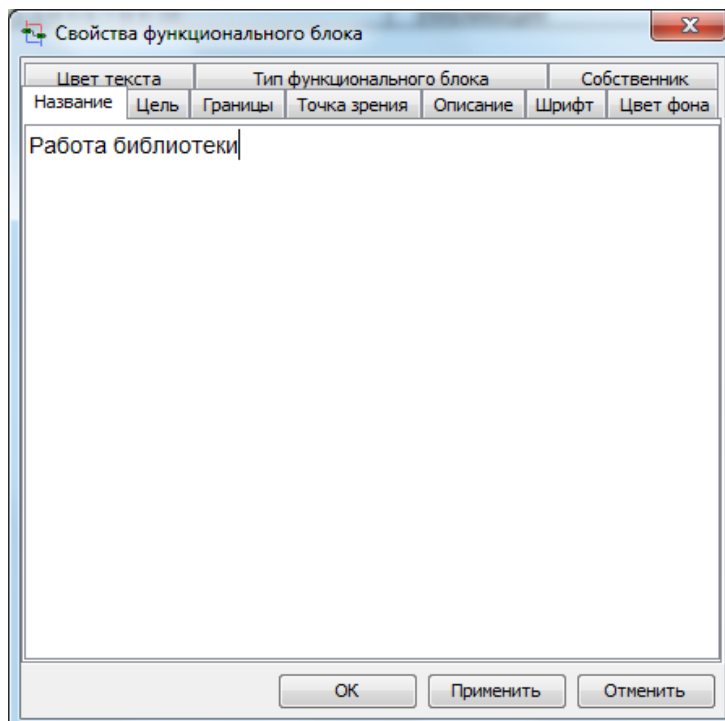


Рисунок 1.20 - Диалог определения названия функционального блока

4. Для функционального блока измените шрифт, задайте, например, Times New Roman, 18 (Рисунок 1.21).

5. Присвойте атрибутам свойств модели значения:

Цель: «Описать функционирование библиотеки для составления спецификаций информационной модели» (Рисунок 1.22).

Границы: «Технологические аспекты деятельности библиотеки» (Рисунок 1.23).

Точка зрения: «Руководитель библиотеки» (Рисунок 1.24).

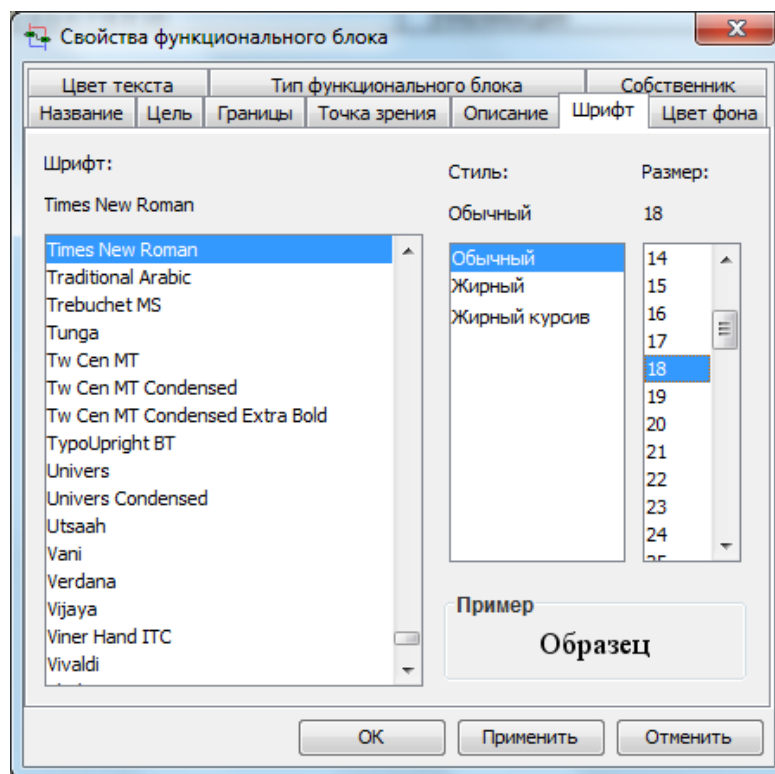


Рисунок 1.21 - Диалог настройки свойств функционального блока

Свойства функционального блока

| Цвет текста  |      | Тип функционального блока |              |          | Собственник |           |
|--|------|---------------------------|--------------|----------|-------------|-----------|
| Название   | Цель | Границы                   | Точка зрения | Описание | Шрифт       | Цвет фона |
| Описать функционирование библиотеки для составления спецификаций информационной модели |      |                           |              |          |             |           |

OK Применить Отменить

Рисунок 1.22 - Определение атрибута «Цель»

Свойства функционального блока

| Цвет текста                                     |      | Тип функционального блока |              |          | Собственник |           |
|---|------|---------------------------|--------------|----------|-------------|-----------|
| Название  | Цель | Границы                   | Точка зрения | Описание | Шрифт       | Цвет фона |
| Технологические аспекты деятельности библиотеки |      |                           |              |          |             |           |

OK Применить Отменить

Рисунок 1.23 - Определение атрибута «Границы»

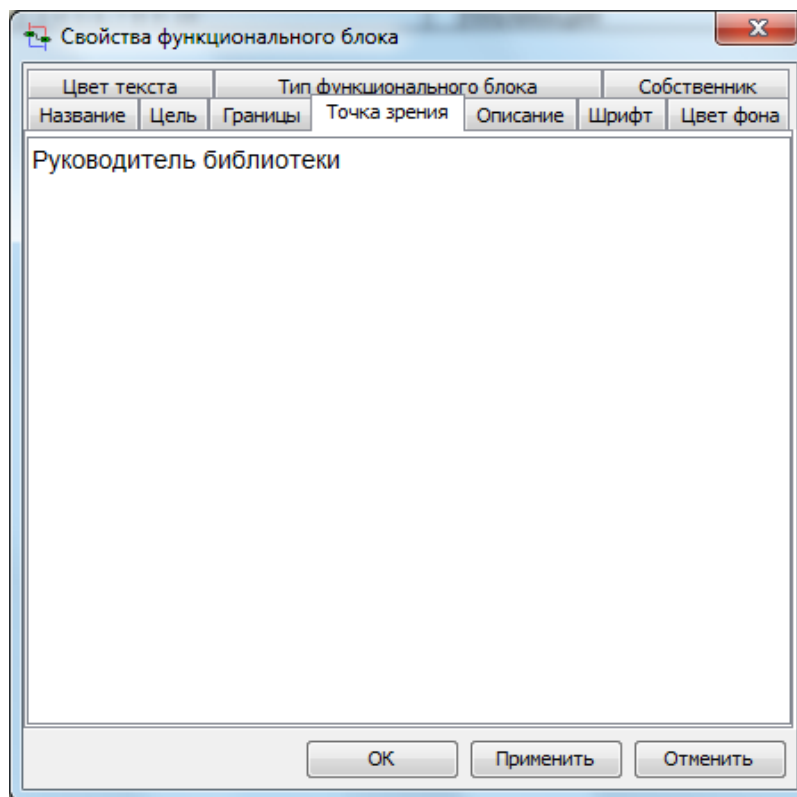


Рисунок 1.24 - Определение атрибута «Точка зрения»

Атрибут «Название» элементов диаграммы присутствует по умолчанию всегда. Созданный ранее в дополнение к нему атрибут «Описание» строкового типа будем использовать для краткого описания назначения этого блока. И возьмите за правило для всех структурных элементов модели задавать такой атрибут, содержащий полезную метаинформацию (таблица 1).

6. В диалоге «Редактировать активный элемент» контекстного меню функционального блока «Работа библиотеки» появится закладка «Описание». Присвойте одноименному атрибуту значение «Под работой библиотеки имеются в виду технологические аспекты ее функционирования» (Рисунок 1.25).

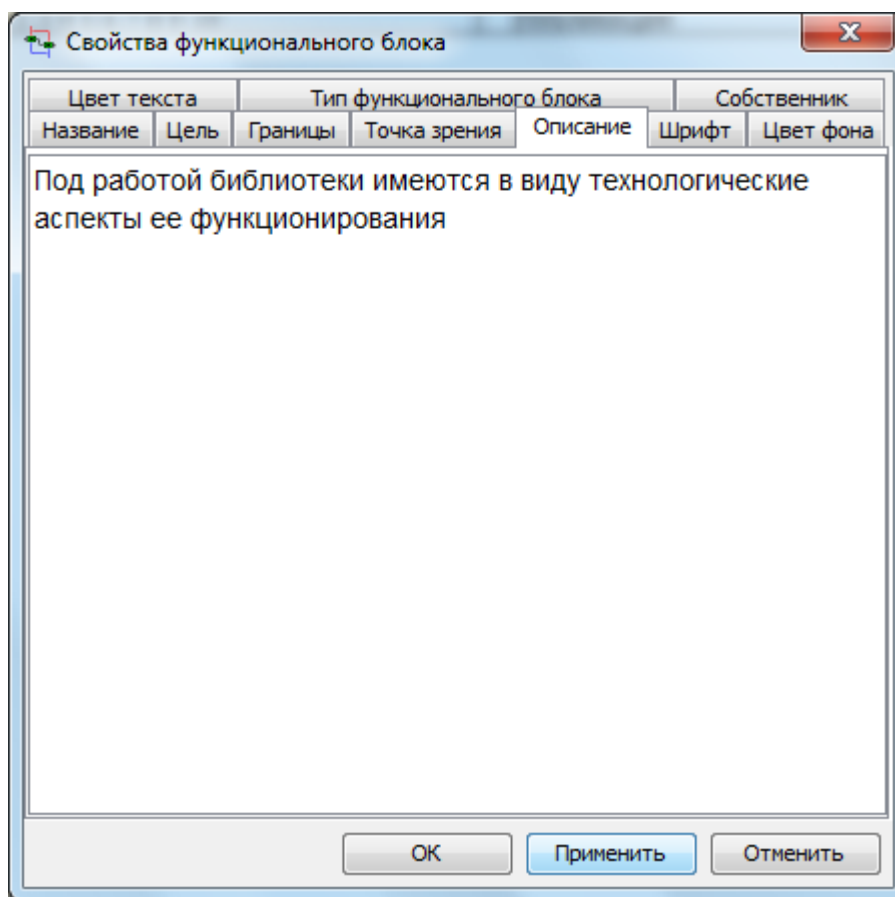


Рисунок 1.25 - Определение атрибута «Описание»



Таблица 1. Описание функциональных блоков

| Название блока                            | Номер блока | Описание   |
|---|-------------|--|
| Работа библиотеки                         | A0          | Под работой библиотеки имеются в виду технологические аспекты ее функционирования  |
| Комплектование библиотеки и хранение книг | A1          | Комплектование библиотеки предполагает приобретение новых книг, их хранение и списание.  |
| Комплектование библиотеки                 | A11         | Комплектование библиотеки предполагает приобретение новых книг и списание пришедших в негодность. При комплектовании каждому экземпляру книги присваивается инвентарный номер. |
| Хранение книг                             | A12         | Экземпляры книг хранятся в хранилище и выдаются по заявкам абонентов во временное пользование.   |
| Справочно-библиографическое обслуживание  | A2          | Справочно-библиографическое обслуживание предполагает занесение сведений о книгах в каталог и поиск книг в каталоге.   |
| Занесение в каталог                       | A21         | Вновь приобретенные книги регистрируются в каталоге  |
| Библиографический поиск                   | A22         | По запросу абонента осуществляется поиск информации о книге в каталоге.  |
| Абонементное обслуживание                 | A3          | Абонементное обслуживание в том числе: 1) запись на абонемент; 2) поиск книг в каталоге; 3) оформление заявки в хранилище; 4) выдача книг; 5) прием возвращенных книг;         |
| Запись на абонемент                       | A31         | Посетители библиотеки могут быть записаны в качестве ее абонентов.   |
| Выдача книг                               | A32         | Затребованные книги при наличии их в хранилище могут быть выданы.  |
| Поиск книг                                | A33         | Поиск сведений о книге выполняется по заявке абонента  |
| Возврат книг                              | A34         | Выданные книги подлежат возврату и размещению их в хранилище   |
| Оформление заявки                         | A35         | При наличии свободного экземпляра книги в хранилище оформляется заявка на затребованную книгу.   |

Взаимодействие библиотеки с окружающей средой моделируется при помощи стрелок, которые в том или ином качестве представляют данные, сопутствующие выполнению функции. Для создания стрелок необходимо:

1. Щелчком экранной кнопки с горизонтальной стрелкой на панели инструментов инициировать режим работы со стрелками.
2. Нажать левой клавишей мыши на обрамление диаграммы с соответствующей стороны (появится черная полоса) и перевести

курсор внутрь функционального блока на соответствующий сегмент в виде черного треугольника.

Стрелкам должно быть присвоено имя. Для этого следует щелчком правой клавиши мыши по стрелке, вызвать контекстное меню и выбрать опцию «Редактировать активный элемент». На контекстной диаграмме имеет смысл изменить размер шрифта подписи стрелки, задайте, например, Times New Roman, 12 (Рисунок 1.26). В соответствии с канонem IDEF0 название стрелки должно быть именем существительным (Рисунок 1.27).

Приступим к размещению на контекстной диаграмме граничных стрелок.

В основе работы библиотеки лежат операции с книгами<sup>2</sup>. Книги поступают в библиотеку извне и выводятся из нее. Таким образом, входом и выходом функционального контекстного блока являются книги. Для того чтобы показать эти разные состояния книг рекомендуется присвоить им разные имена, например «Книги на входе» и «Книги на выходе».

Другой важной функцией библиотеки является формирование контингента абонентов. Стрелки входа «Посетители» и выхода – «Абоненты» представляют предпосылки и результаты этой деятельности. Работа библиотеки осуществляется под влиянием факторов, регламентирующих эту деятельность.

Таковыми факторами, по крайней мере, являются «Бюджет» (бюджетное финансирование) и «Правила пользования», которые на контекстной диаграмме играют роль управления. Наконец исполнителями всех видов работы библиотеки являются ее сотрудники (стрелка «Персонал»).

В процессе построения модели можно редактировать внешний вид (шрифт, цвет и др.) ее элементов. Для этого достаточно правой клавишей мыши вызвать контекстное меню и выбрать опцию «Редактировать активный элемент».

Разместите на контекстной диаграмме граничные стрелки по выше, приведенному алгоритму (таблица 2).

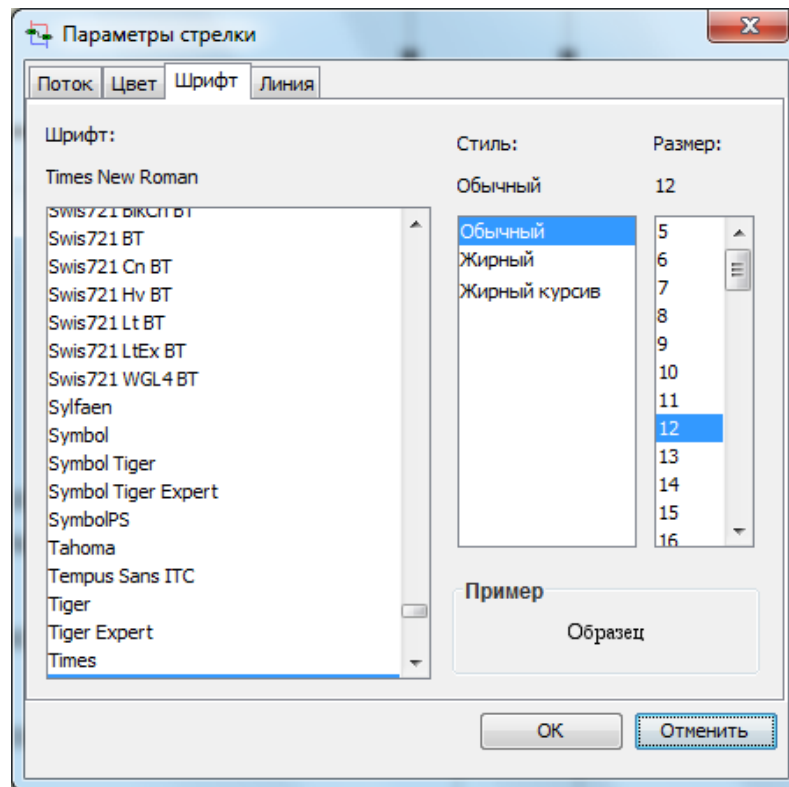


Рисунок 1.26 - Диалог настройки свойств стрелок

Свойства модели, заданные как атрибуты контекстного функционального блока, могут быть продублированы в виде текста на контекстной диаграмме.

В результате должна получиться контекстная диаграмма, подобная приведенной на Рисунок 1.28.

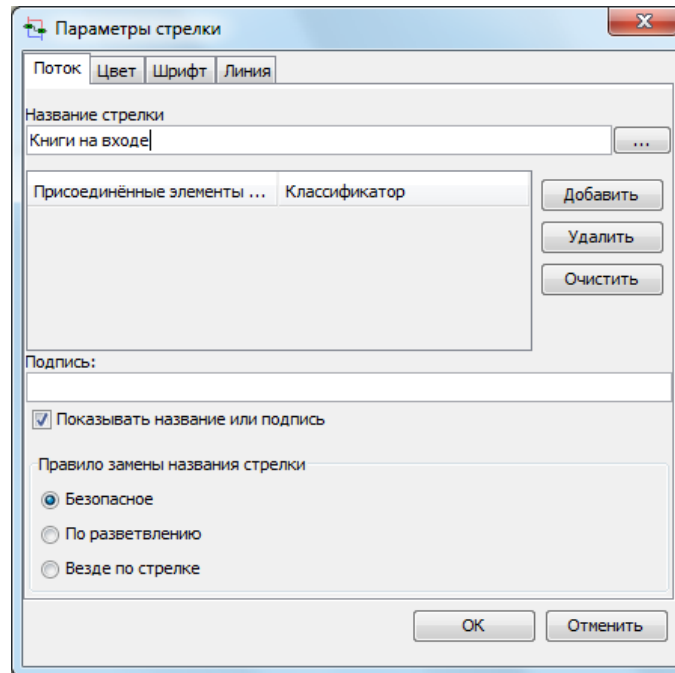


Рисунок 1.27 - Диалог определения названия стрелки

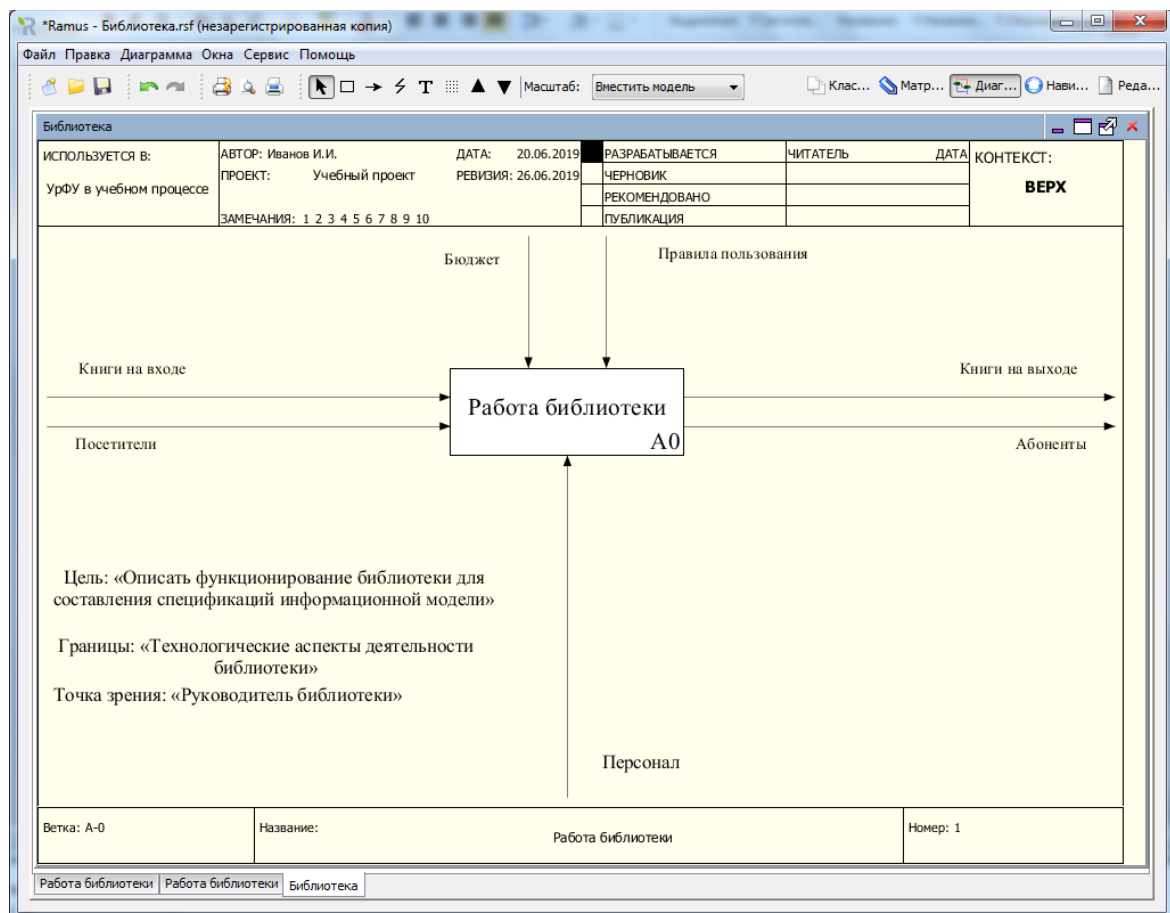


Рисунок 1.28 - Контекстная диаграмма

Таблица 2. Описание стрелок

| Название стрелки         | Описание  |
|--------------------------|---|
| Абоненты                 | Абоненты – это, зарегистрированные клиенты библиотеки.<br>После регистрации они приобретают права законных пользователей.   |
| Библиографическая карта  | Каталожная карточка с библиографическими данными книги.   |
| Бюджет                   | Бюджет регламентирует все виды работ в библиотеке.  |
| Возвращаемые книги       | Книги, возвращаемые абонентами, один из вариантов поступления книг  |
| Возвращенные книги       | Возвращенные на абонемент книги размещаются в хранилище   |
| Выданные книги           | Выданные книги – это, один из вариантов книг на выходе.   |
| Запрос                   | Перед оформлением заявки выполняется запрос на поиск информации о книге в каталоге.   |
| Зарегистрированные книги | Статус зарегистрированной приобретает книга после ее занесения в каталог. Книга может быть зарегистрирована, но ни разу не затребована. Это один из вариантов книг на выходе. |
| Затребованные книги      | При наличии свободного экземпляра по заявке затребованная книга поступает на абонемент.   |
| Заявка                   | При наличии свободного экземпляра книги   |

|                          |   |  |
|--------------------------|---|--|
|                          |   | оформляется заявка на ее получение во временное пользование  |
| Информация об абоненте   |   | Персональные данные абонента, зафиксированные при его записи в библиотеку.                                     |
| Книги на входе           |   | Источники книг на входе библиотеки: 1) новые поступления; 2) возвращенные книги.                               |
| Книги на выходе          |   | Книги на выходе это: 1) зарегистрированные, но не востребованные книги; 2) выданные книги; 3) списанные книги. |
| Неудовлетворенная заявка |   | Затребованной книги нет в свободном доступе.   |
| Новые книги              |   | Новые книги – это один из вариантов поступления книг в библиотеку.   |
| Персонал                 |   | Сотрудники библиотеки.   |
| Посетители               |   | Библиотеку могут посещать клиенты, не являющиеся ее абонентами.  |
| Потребность в информации | в | Стимул поиска книг в библиотеке  |
| Правила пользования      |   | Правила пользования распространяются только на справочно- библиографическое и абонементное обслуживание.       |
| Списанные книги          |   | Книги, пришедшие в негодность, подлежат списанию. Это один из вариантов книг на выходе.                        |
| Справка                  |   | Справка – это, результат справочно-библиографического поиска по запросу абонента.                              |
| Учтенные книги           |   | После поступления новой книге присваивается инвентарный номер, и она приобретает статус учтенной книги.        |
| Хранимые книги           |   | Вновь приобретенным книгам присваивается   |

|  |   |
|--|---|
|  | инвентарный номер, и они приобретают статус хранимых книг, которые должны быть зарегистрированы в каталоге. |
|--|---|


### Построение диаграммы декомпозиции

Контекстная диаграмма дает обобщенное представление о предметной области, недостаточное для понимания того, в чем состоит суть работы библиотеки. Для более детального описания функционирования библиотеки необходимо выполнить декомпозицию ее главной функции.

Исходя из общих соображений о библиотеке, можно выделить следующие функции, детализирующие представление о ее работе:

- комплектование и хранение книг;
- справочно-библиографическое обслуживание;
- абонементное обслуживание.

Для детализации представления о работе библиотеки необходимо построить диаграмму декомпозиции первого уровня.

Для перехода к диаграмме декомпозиции необходимо выделить блок, подлежащий детализации, и на панели инструментов нажать кнопку перехода к дочерним диаграммам в виде треугольника, направленного вниз  .

В раскрывшемся диалоге (Рисунок 1.29) следует указать число функциональных блоков на диаграмме декомпозиции (в нашем случае 3). Если появится необходимость, недостающие блоки можно будет добавить дополнительно с панели инструментов.

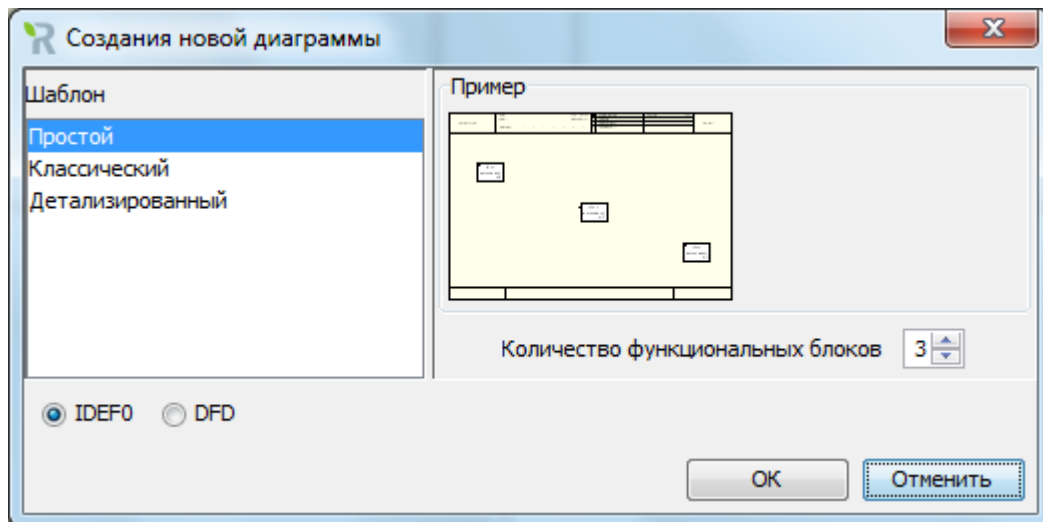


Рисунок 1.29 - Диалог создания диаграммы декомпозиции

При выполнении декомпозиции на дочернюю диаграмму вместе с функциональными блоками с родительской диаграммы мигрируют граничные стрелки, которые нужно связать с соответствующими блоками (Рисунок 1.30).

Функциональным блокам по известным правилам следует присвоить соответствующие имена и описания (таблица 1), например:

- «Комплектование и хранение книг»
- «Справочно-библиографическое обслуживание»
- «Абонементное обслуживание».

На диаграмме декомпозиции для имен функциональных блоков задайте шрифт Times New Roman, 14, для стрелок - Times New Roman, 12. При редактировании объектов диаграммы не рекомендуется злоупотреблять цветами.



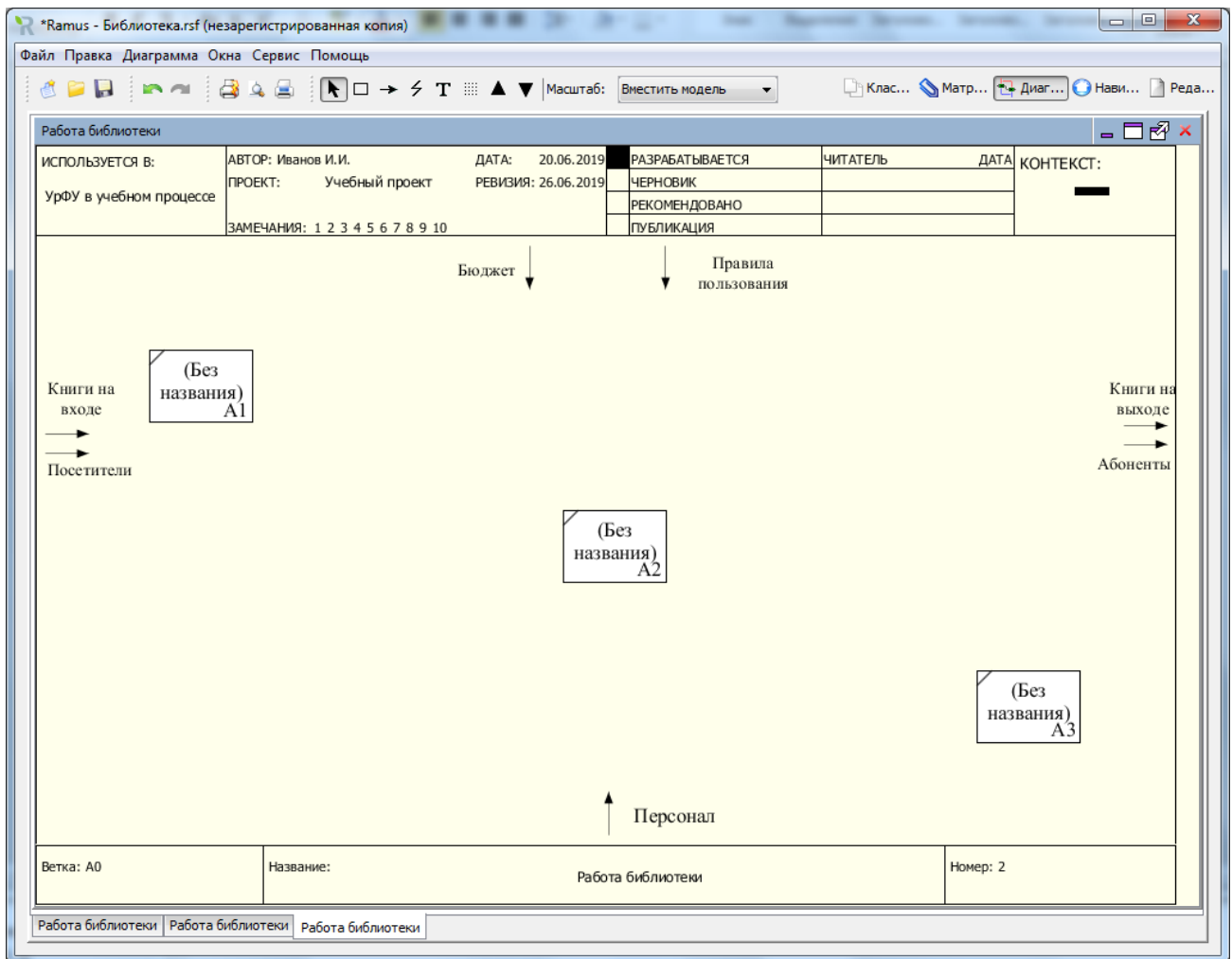


Рисунок 1.30 - Шаблон построения диаграммы декомпозиции

Граничные стрелки необходимо развести по соответствующим блокам.

Для соединения стрелок с блоками следует в режиме работы со стрелками нажать левой клавишей мыши по стрелке и перевести курсор внутрь функционального блока на соответствующий сегмент в виде черного треугольника. Для соединения выходной стрелки с блоком следует перевести курсор от правой грани блока к стрелке выхода. При размещении стрелок на диаграмме декомпозиции рекомендуется следить за тем, чтобы стрелки минимально переплетались и не затрудняли чтение диаграммы. Для усиления читабельности диаграммы смелее используйте в режиме редактирования перемещение, изменение размеров, выноски и другие преобразования объектов диаграммы.

При размещении стрелок на диаграмме следует помнить, что стрелки представляют данные, которые сопутствуют выполнению соответствующих функций. Рассмотрим последовательно функциональные блоки на диаграмме декомпозиции и разместим соответствующие им стрелки.

Начнем с блока «Комплектование и хранение книг» (Рисунок 1.31).

На вход блока естественно будет направить граничную стрелку «Книги на входе», которая представляет вновь приобретенные библиотекой книги.

С выхода следует вывести стрелку «Книги на выходе», которая в данном контексте представляет списанные книги. В качестве исполнителя комплектования и хранения участвует персонал, граничную стрелку «Персонал» следует подвести снизу. Выполнение функции регламентируется бюджетным функционированием, поэтому в качестве стрелки управления следует подвести стрелку «Бюджет». Правила пользования библиотекой на ее комплектование не распространяются. В качестве одного из результатов функции «Комплектование и хранение книг» будет также передача библиографических данных новых книг в справочно-библиографический отдел.

Вновь приобретенные книги поступают в фонд библиотеки, их переплетам присваиваются инвентарные номера, после чего они приобретают статус «Приобретенные книги». Соответствующая стрелка представляет результат-выход блока «Комплектование и хранение книг», а библиографические данные этих книг служат в качестве входа для блока «Справочно-библиографическое обслуживание». Стрелка «Приобретенные книги» является внутренней стрелкой диаграммы декомпозиции этого уровня. Для ее отображения на диаграмме следует в режиме работы со стрелками мышью соединить выход блока источника с входом блока приемника.

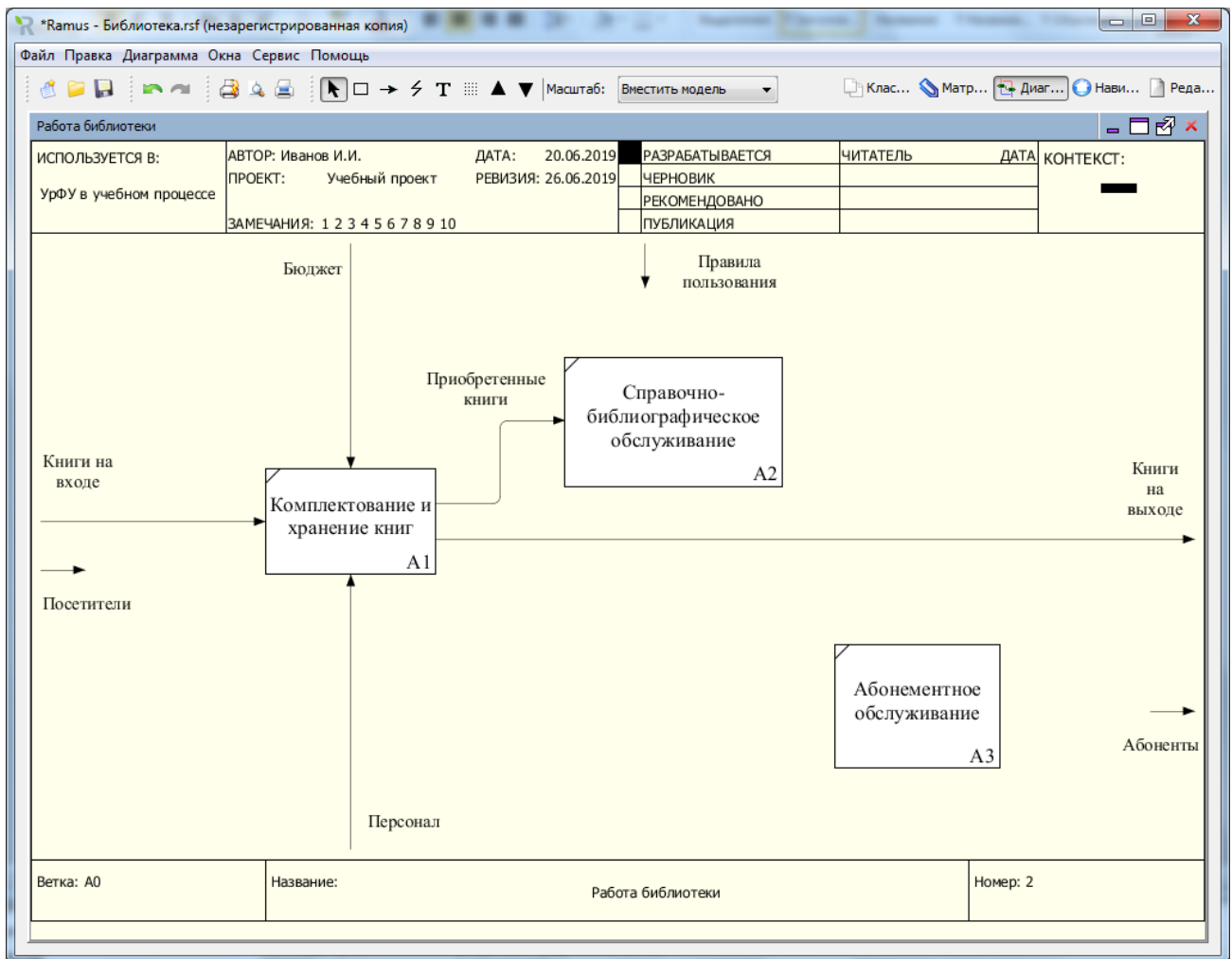


Рисунок 1.31 - Спецификация блока «Комплектование и хранение книг»

Результатом работы справочно-библиографического отдела (выход блока «Справочно-библиографическое обслуживание») является формирование каталога, где размещаются библиографические данные зарегистрированных книг, даже если они ни разу не будут востребованы. Соответствующая стрелка «Зарегистрированные книги» представляет собой один из вариантов «Книги на выходе» (Рисунок 1.32). Особое состояние этой категории книг также как списанных книг может быть представлено слиянием ветвей в общую стрелку «Книги на выходе». Для слияния ветвей следует в режиме работы со стрелками стрелку выхода блока «Справочно-библиографическое обслуживание» соединить со стрелкой «Книги на выходе». Ветвям до

слияния рекомендуется присвоить имена «Зарегистрированные книги» и «Списанные книги», которые отражают специфическое состояние «Книг на выходе». В качестве исполнителя справочно-библиографического обслуживания участвует персонал. Поэтому стрелку «Персонал» следует ответить и направить в качестве механизма к блоку «Справочно-библиографическое обслуживание». Для разветвления стрелок следует мышью щелкнуть по исходной стрелке, а затем по блоку, к которому подходит ветвь. На выполнение справочно-библиографического обслуживания влияют, и бюджетное финансирование и правила пользования библиотекой. Поэтому соответствующие стрелки нужно ответить и направить на верхнюю грань блока «Справочно-библиографическое обслуживание».

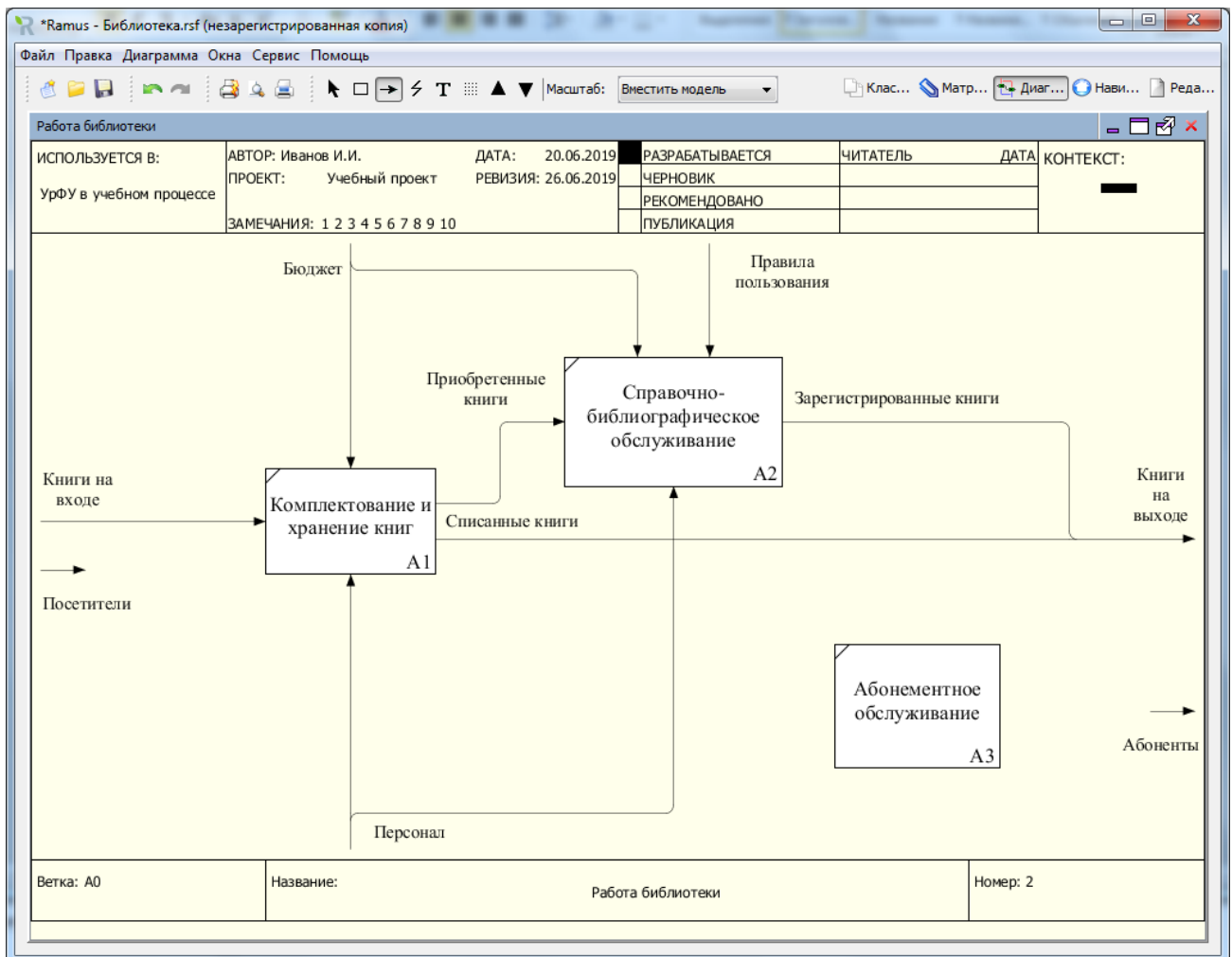


Рисунок 1.32 - Спецификация блока «Справочно-библиографическое обслуживание»

Одной из основных функций абонементного обслуживания является запись потенциальных пользователей библиотеки на абонемент. Граничную стрелку «Посетители» естественно будет направить на вход блока «Абонементное обслуживание», а стрелку «Абоненты» вывести из его правой грани (Рисунок 1.33).

Алгоритм работы на абонементе можно представить себе в виде такой последовательности действий.

Абонент делает запрос в справочно-библиографический отдел о наличии интересующего его библиографического источника. Внутренняя стрелка «Запрос» представляет собой один из результатов (выходов) блока «Абонементное обслуживание» и вход блока «Справочно-библиографическое обслуживание».

В ответ на запрос с выхода блока «Справочно-библиографическое обслуживание» на вход «Абонементное обслуживание» передается ответ в виде стрелки «Справка» (Рисунок 1.33).

При наличии затребованной книги из абонемента как результат работы «Абонементное обслуживание» на вход «Комплектование и хранение книг» передается заявка на затребованную книгу (внутренняя стрела «Заявка»).

В ответ на заявку из книгохранилища (выход блока «Комплектование и хранение книг») на абонемент (вход блока «Абонементное обслуживание») передаются внутренние стрелки «Затребованная книга» и «Неудовлетворенная заявка» (в случае отсутствия книги в фонде) (Рисунок 1.35).

Результатом работы абонемента являются также книги, выданные абонентам и ушедшие за пределы библиотеки. В случае удовлетворенной заявки затребованная книга выводится из библиотеки как результат «Выданная книга» (стрелка выхода блока «Абонементное обслуживание»).

Выданная книга является одним из вариантов книги на выходе, этот факт следует отобразить на диаграмме в виде слияния стрелки «Выданная книга» со стрелками «Зарегистрированные книги» и «Списанные книги» (Рисунок 1.33).

Книги, выданные на время абонентам, возвращаются в библиотеку как вариант книг на входе. Соответствующая стрелка

«Возвращаемые книги» должна быть ответвлена от стрелки «Книги на входе» и направлена на вход блока «Абонементное обслуживание». Ветвь, направленная на вход блока «Комплектование и хранение книг», должна быть поименована, например, как «Новые книги». Имена ветвям должны быть присвоены после точки разветвления. Если после разветвления имя стрелки не указано, то предполагается, что ее имя совпадает с именем стрелки до разветвления.

И, наконец, книги, которые вернули абоненты, со статусом «Возвращенные книги» возвращаются в книгохранилище (выход блока «Абонементное обслуживание» и вход блока «Комплектование и хранение книг») (Рисунок 1.33).

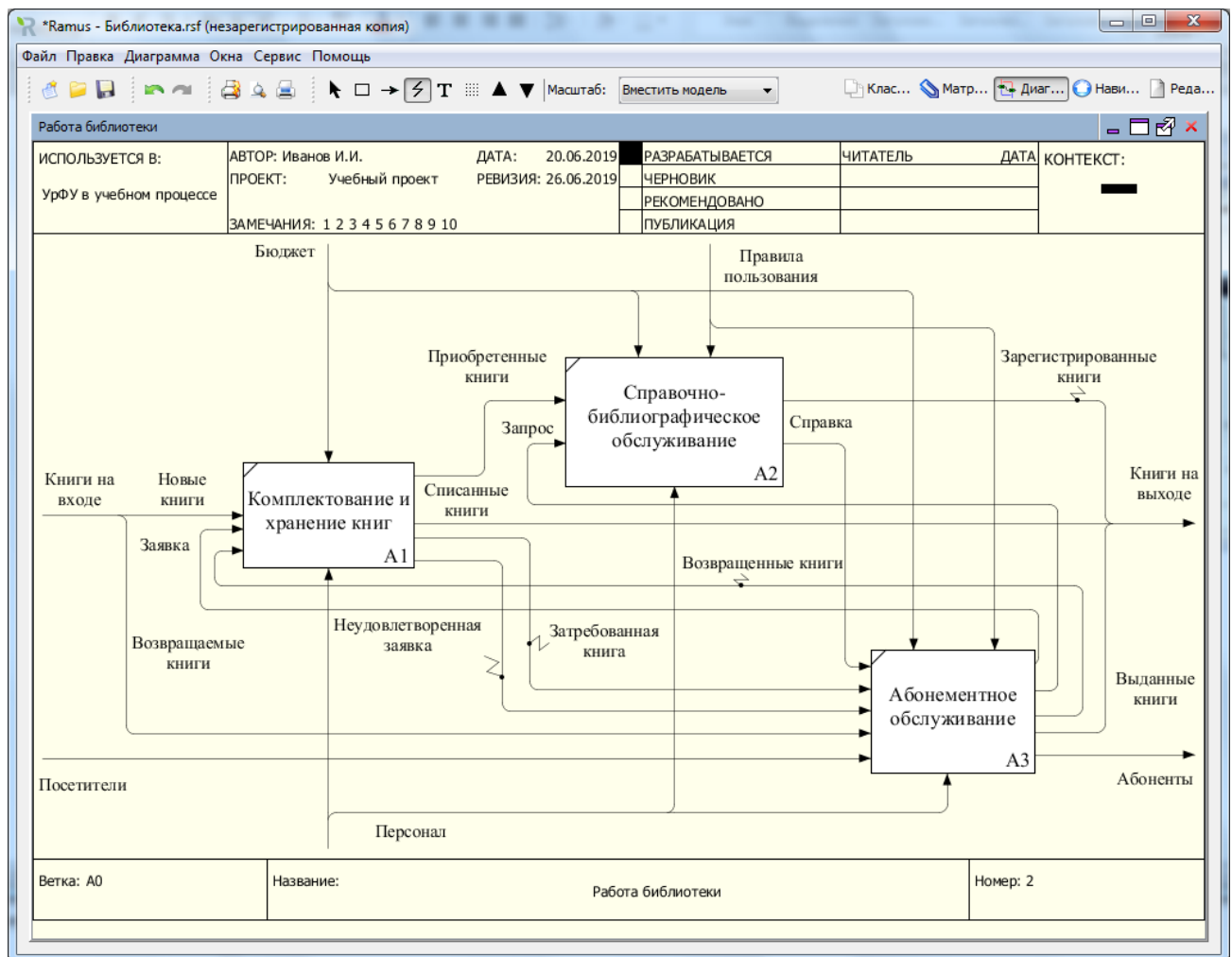



Рисунок 1.33 - Диаграмма декомпозиции контекстного блока «Работа библиотеки»

В качестве исполнителя абонементного обслуживания участвует персонал. Поэтому стрелку «Персонал» следует ответить и направить в качестве механизма к блоку «Абонементное обслуживание». На выполнение абонементного обслуживания влияют, и бюджетное финансирование и правила пользования библиотекой. Поэтому соответствующие стрелки нужно ответить и направить на верхнюю грань блока «Абонементное обслуживание».

В результате диаграмма декомпозиции первого уровня будет выглядеть примерно, как на Рисунке 1.33.

Роль граничной стрелки «Персонал» обсудим особо. Персонал является исполнителем всех видов работ. Присутствие стрелки «Персонал» на диаграмме декомпозиции мало информативно. Поэтому область действия этой стрелки может быть ограничена только контекстной диаграммой без передачи ее на диаграмму декомпозиции. Такое состояние стрелки называется туннелированием «не в дочерней диаграмме». Для туннелирования стрелки по такому принципу необходимо:

1. Удалить стрелку «Персонал» на диаграмме декомпозиции. В результате на материнской диаграмме конец стрелки будет обрамлен квадратными скобками.
2. Перейти к материнской диаграмме. Для перехода наверх необходимо на панели инструментов нажать кнопку перехода к материнской диаграмме с треугольником, направленным вверх .
3. На материнской диаграмме по квадратным скобкам конца стрелки следует правой клавишей мыши вызвать контекстное меню (Рисунок 1.34) и в нем выбрать опции «Туннель» и «Обозначить круглыми скобками (показывать в отчетах для дочерних элементов)». В результате на дочерней диаграмме стрелка будет отсутствовать, а на материнской диаграмме конец стрелки будет обрамлен круглыми скобками (Рисунок 1.35). Это признак того, что стрелка затуннелирована без передачи ее на дочернюю диаграмму. При этом роль затуннелированной стрелки «Персонал» будет неявно учитываться на дочерней диаграмме.

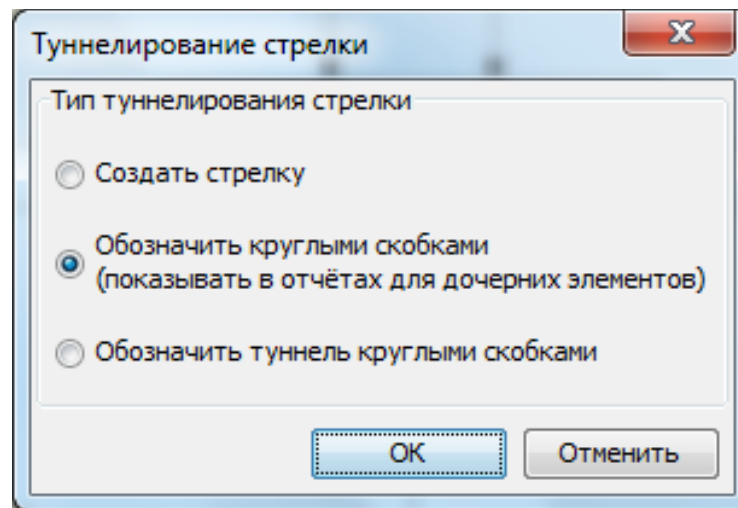


Рисунок 1.34 - Диалог туннелирования стрелок

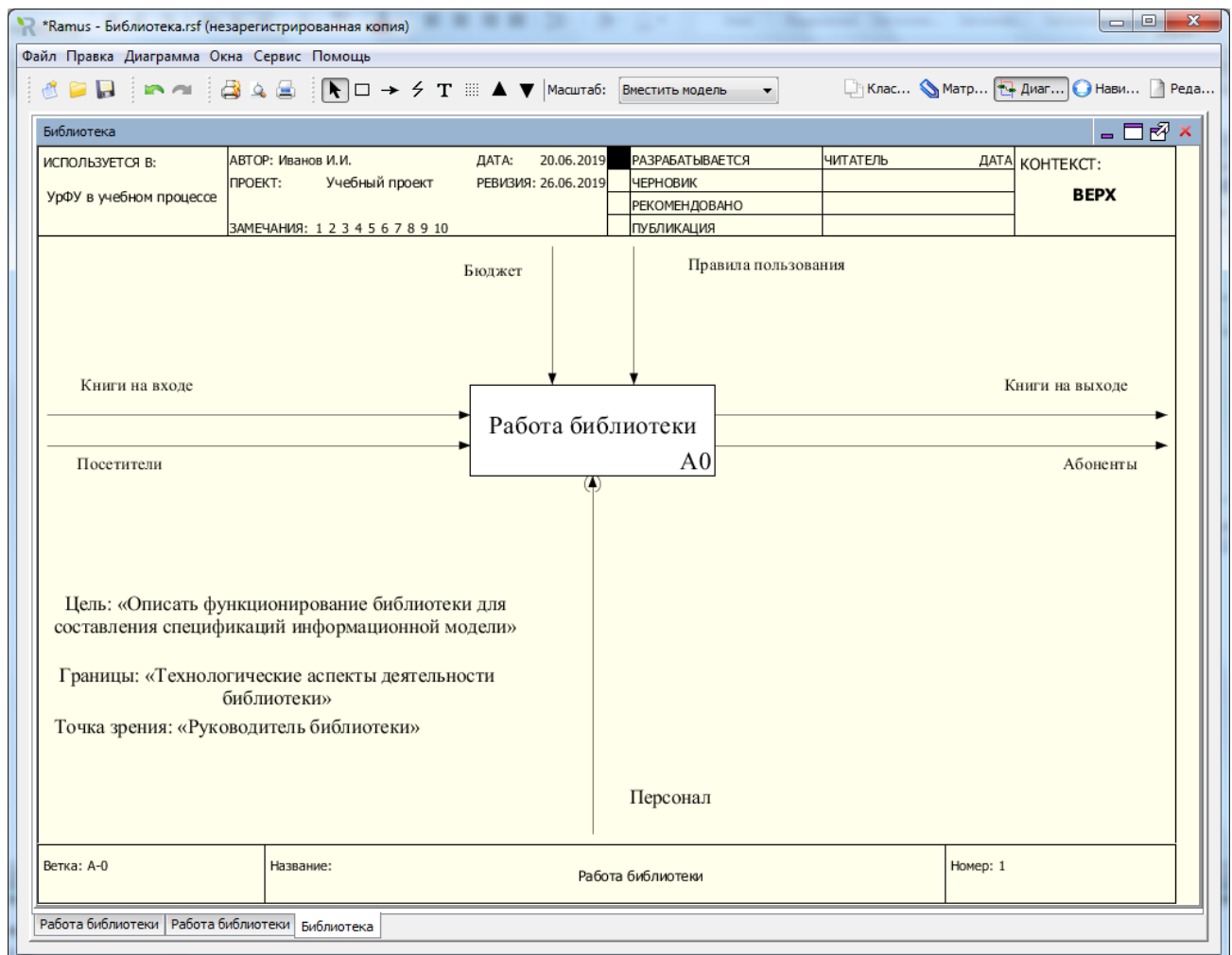


Рисунок 1.35 - Контекстная диаграмма с туннелированной стрелкой



## **Построение диаграмм декомпозиции второго уровня**

Если детализация функций на диаграмме декомпозиции первого уровня недостаточна, процедуру декомпозиции можно применить к ее функциональным блокам, используя те же принципы построения диаграмм.

Например, блок «Комплектование и хранение книг» можно подвергнуть декомпозиции на два блока «Комплектование библиотеки» и «Хранение книг».

Граничные стрелки блока «Комплектование и хранение книг» материнской диаграммы мигрируют на дочернюю диаграмму (Рисунок 1.36).

Стрелку «Новые книги» будет естественно направить на вход блока «Комплектование библиотеки». Стрелки «Списанные книги» и «Приобретенные книги» следует связать с выходом этого блока.

Граничные стрелки «Заявка» и «Возвращенные книги» являются входами для блока «Хранение книг». Стрелки «Затребованные книги» и «Неудовлетворенная заявка» играют роль выходов блока «Хранение книг».

После регистрации вновь приобретенные книги передаются на хранение в книгохранилище. Внутренняя стрелка «Учтенные книги» учитывает это обстоятельство.

В качестве управления используется только стрелка «Бюджет», которую следует разветвить и направить сверху к обоим блокам декомпозиции.

В качестве исполнителя фигурирует персонал, участие которого неявно подразумевается (стрелка «Персонал» была затуннелирована на контекстной диаграмме).

Функциональный блок «Справочно-библиографическое обслуживание» на диаграмме первого уровня также может быть подвергнут декомпозиции на два блока «Занесение в каталог» и «Библиографический поиск» (Рисунок 1.37).

Граничные стрелки блока «Справочно-библиографическое обслуживание» материнской диаграммы мигрируют на соответствующую дочернюю диаграмму.

Стрелку «Приобретенные книги» следует направить на вход блока «Занесение в каталог». С выхода этого блока следует снять стрелку «Зарегистрированные книги».

Стрелку «Запрос» будет правильно задать как входную стрелку блока «Библиографический поиск». Тогда выходом этого блока послужит стрелка «Справка».

На примере декомпозиции блока «Справочно-библиографическое обслуживание» рассмотрим еще один случай особой роли стрелок.

На некоторых диаграммах могут быть применены стрелки, которым соответствуют малозначимые данные, актуальные только на данном уровне декомпозиции без передачи их на уровень материнской диаграммы. Такое состояние стрелки называется туннелированием «не в материнской диаграмме». Например, такая малозначимая информация как содержание библиографической карты может быть полезна только для библиографического поиска. Соответствующая стрелка «Библиографическая карта» может быть полезна как вход блока «Библиографический поиск». Чтобы не усложнять материнскую диаграмму такую стрелку можно затуннелировать на уровне дочерней диаграммы.

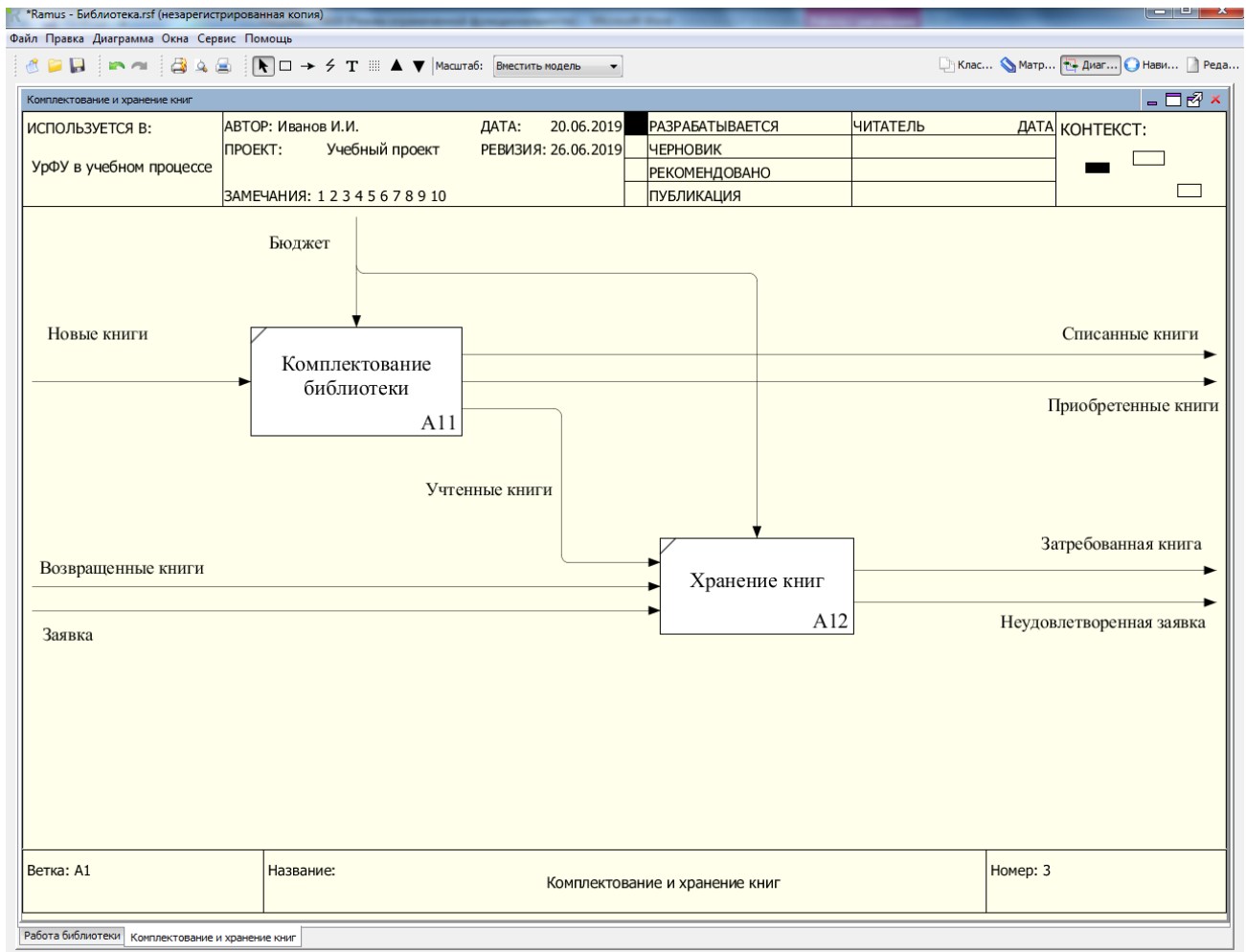


Рисунок 1.36 - Диаграмма декомпозиции блока «Комплектование и хранение книг»

Для этого нужно:

1. Создать стрелку входа «Библиографическая карта» для блока «Библиографический поиск». В результате начало такой стрелки будет обрамлено квадратными скобками.
2. На дочерней диаграмме по квадратным скобкам начала стрелки следует правой клавишей мыши вызвать контекстное меню и в нем выбрать опции «Туннель» и «Обозначить туннель круглыми скобками». В результате на дочерней диаграмме начало стрелки будет обрамлено круглыми скобками. Это признак того, что стрелка затуннелирована без передачи ее на материнскую диаграмму.

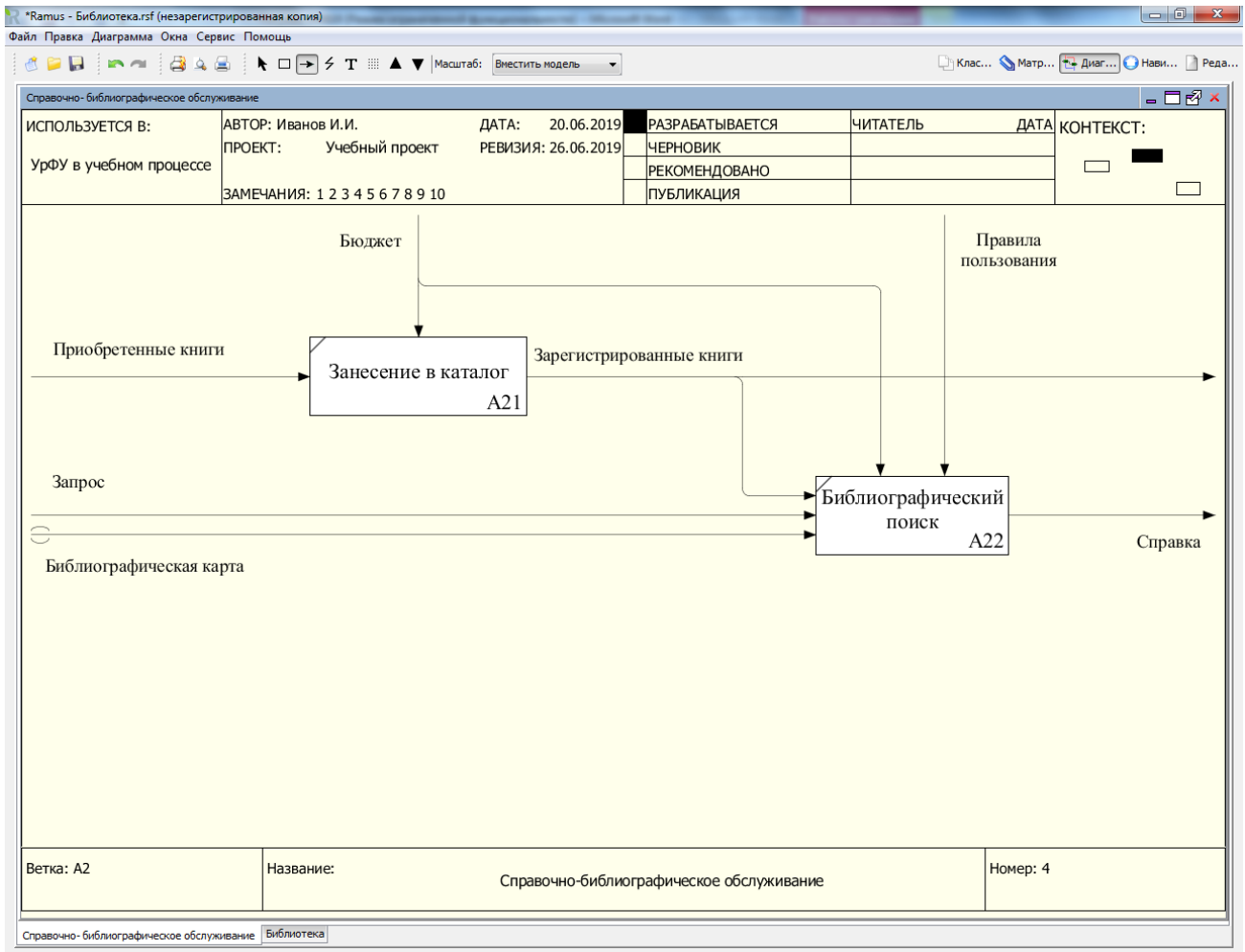


Рисунок 1.37 - Диаграмма декомпозиции блока "Справочно-библиографическое обслуживание"

Наконец осталось обозначить связь функциональных блоков «Занесение в каталог» и «Библиографический поиск». Роль связующей играет внутренняя стрелка «Зарегистрированные книги». Эта стрелка является также стрелкой выхода блока «Занесение в каталог». Поэтому внутреннюю стрелку можно отвести от выходной стрелки и оставить то же самое имя. Если стрелки после разветвления имеют один и тот же смысл, их можно не именовать специально, а оставить общее имя «Зарегистрированные книги». При этом общее имя должно быть указано до точки разветвления стрелок.

В качестве управления используется стрелки «Бюджет» и «Правила пользования». Стрелку «Бюджет» следует разветвить и направить сверху к обоим блокам декомпозиции. Правила пользования распространяются только на библиографический поиск,

поэтому стрелку «Правила пользования» следует направить на верхнюю грань блока «Библиографический поиск».

Наконец, функциональный блок «Абонементное обслуживание» диаграммы декомпозиции первого уровня может быть подвергнут декомпозиции для уточнения того, что есть абонементное обслуживание.

Естественно будет предположить реализацию на абонементе следующих функций:

- запись на абонемент;
- поиск книг;
- оформление заявки;
- выдача книг;
- возврат книг.

Таким образом, при построении диаграммы декомпозиции блока «Абонементное обслуживание» можно предусмотреть пять соответствующих функциональных блоков (Рисунок 1.38).

Граничные стрелки блока «Абонементное обслуживание» мигрируют на диаграмму декомпозиции. И их нужно связать с функциональными блоками диаграммы декомпозиции.

Начнем с блока «Запись на абонемент». Очевидно, что стрелка «Посетители» послужит для него входом, а «Абоненты» - выходом. Кроме того, при записи на абонемент библиотеки фиксируется информация об абонентах (т.н. формуляр). Поэтому результатом записи на абонемент является не только изменение статуса потенциального пользователя библиотеки, но и сохранение его регистрационных данных. Последний факт на диаграмме представляет стрелка выхода блока «Запись на абонемент». – «Информация об абоненте».

Предпосылкой (входом) блока «Поиск книг» являются «Потребность в информации» и/или «Неудовлетворенная заявка (отсутствие в свободном доступе экземпляра книги)». Стрелку «Потребность в информации» как малозначимую на данном уровне декомпозиции можно затуннелировать без передачи на материнскую диаграмму. Результатом или выходом блока «Поиск книг» является «Запрос», адресованный абонентом в справочно- библиографический

отдел. Участником и исполнителем поиска книги является абонент. Этот факт представлен стрелкой механизма, ответвленной от стрелки выхода «Абонент» и направленной на нижнюю грань блока «Поиск книг». Имя стрелки «Абонент» распространяется на все ее ветви и поэтому должно быть указано до ее первого разветвления. В таком случае все ветви могут оставаться непоименованными в явном виде.

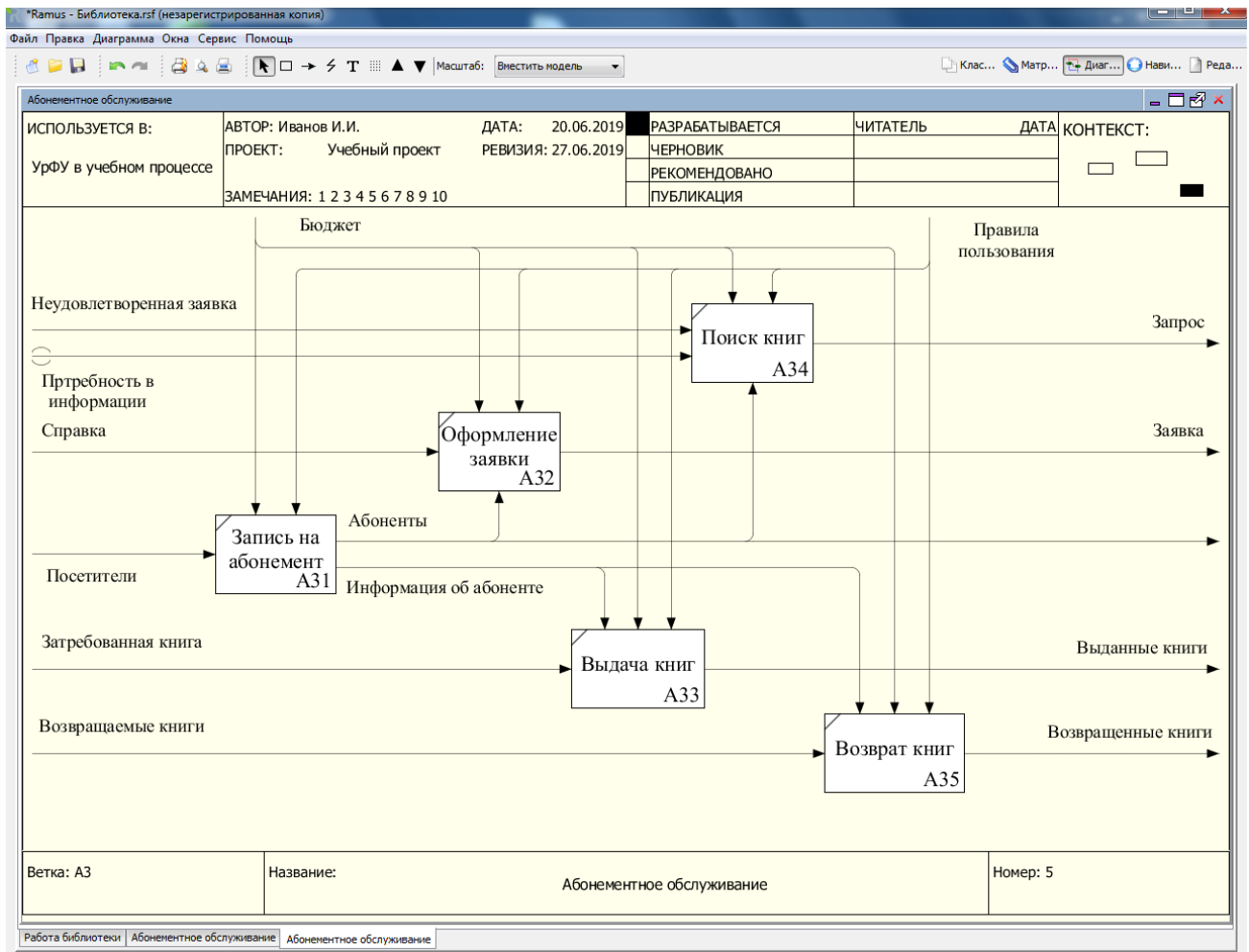


Рисунок 1.38 - Диаграмма декомпозиции блок «Абонементное обслуживание»

Для блока «Оформление заявки» стрелка «Справка» послужит входом, а «Заявка» - выходом. Участником и исполнителем оформления заявки является абонент. Этот факт представлен стрелкой механизма, ответвленной от стрелки выхода «Абонент» и направленной на нижнюю грань блока «Оформить заявку».

Входом для блока «Выдача книги» послужит стрелка «Затребованные книги», а выходом – «Выданные книги». Обсудим роль стрелки «Информация об абоненте». Для блока «Выдача книг» она послужит управлением, отражая тот факт, что речь идет не вообще о какой-то выдаче книг, а о выдаче книг конкретному абоненту с его реквизитами на абонементе.

Для блока «Возврат книг» входом послужит стрелка «Возвращаемые книги», а выходом – «Возвращенные книги». Возврат книг также как и выдача книг обусловлен реквизитами конкретного абонента. Поэтому стрелка «Информация об абоненте» должна ответвиться и направиться как управление на верхнюю грань блока «Возврат книг».

Что касается управления, то все блоки декомпозиции «Абонементное обслуживание» находятся под влиянием бюджетного финансирования и правил пользования. Поэтому соответствующие стрелки должны быть подведены как стрелки управления ко всем блокам.

Участие сотрудников библиотеки в абонементном обслуживании подразумевается при помощи стрелки «Персонал», которая была затуннелирована на контекстной диаграмме.

Итак, мы построили диаграммы трех уровней детализации представления о работе библиотеки. Если детализации на диаграммах декомпозиции третьего уровня оказывается недостаточно, процедуру можно продолжить в отношении тех функциональных блоков, которые требуют детализации понимания.

Построенная нами совокупность диаграмм представляет собой функционально- структурную модель предметной области «Работа библиотеки».

На основании функционального анализа нами построена функциональная модель AS-IS (как есть). Назначение модели этого типа состоит в том, чтобы дать представление адекватное действительному состоянию предметной области с тем, чтобы эта модель была полезна для реорганизации бизнес- процессов в области профессиональной деятельности.

## Разработка модели ТО-ВЕ

На основании критического анализа модели AS-IS может быть построена модель ТО-ВЕ (как будет), учитывающая недостатки действующих бизнес- процессов и предлагающая пути решения проблем. Возможны разные варианты совершенствования бизнес- процессов в предметной области. Поэтому одной модели AS-IS могут соответствовать несколько моделей ТО-ВЕ. Построим хотя бы одну из них. При построении модели ТО-ВЕ будем использовать уже обсуждавшиеся приемы работы в Ramus Educational.

Даже при поверхностном взгляде на разработанную нами функциональную модель, описывающую работу библиотеки, видно, что процедура получения затребованной книги, оказывается слишком затянутой по времени. Связано это с тем, что в ее реализации участвуют разные структурные подразделения библиотеки, каждое из которых оперирует своими собственными данными, недоступными для других участников процесса.

В идеальном случае желательно иметь интегрированные данные с коллективным доступом к ним организованные, например в виде базы данных на компьютере. Желательно, чтобы сотрудники библиотеки имели доступ к интегрированной базе данных в соответствии с их компетенциями, а абоненты могли бы получить сведения об интересующей их книге и возможности ее получения. Еще лучше было бы иметь возможность получить книгу не только в печатном виде, но и в электронном. Электронные библиотечные системы сейчас очень популярны в библиотечном деле. Они не требуют книгохранилища и позволяют копировать книги практически неограниченное число раз. Однако наряду с электронными книгами, по-видимому, останутся и печатные, основанные на традиционной схеме обслуживания абонентов библиотеки. Все это нужно учесть при реорганизации работы библиотеки.

Итак, наши намерения реструктуризации работы библиотеки будут основаны на ее компьютерной модели в виде базы данных. Наверное, возможны альтернативные нашему варианту схемы реорганизации работы библиотеки, но рассмотрим наши предложения.



Начнем, как принято, с построения контекстной диаграммы. Формально она претерпит незначительные изменения. Все соображения по поводу структуры контекстного блока и граничных стрелок остаются в силе.

Единственным изменением будет дополнительная стрелка механизма «Компьютеры», которая учитывает компьютерную реализацию базы данных (Рисунок 1.39).

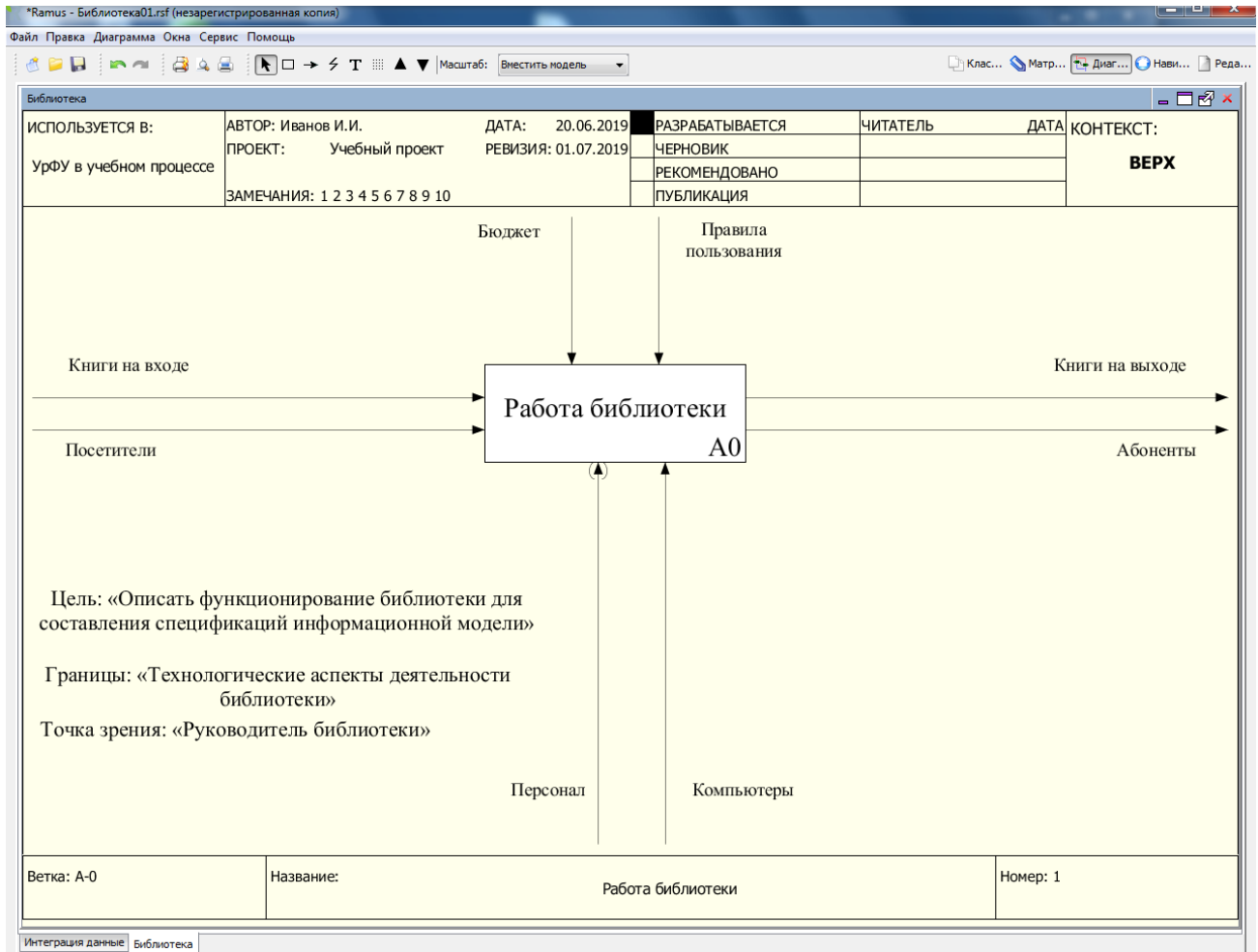


Рисунок 1.39 - Контекстная диаграмма модели ТО-ВЕ

На диаграмме декомпозиции первого уровня к функциональным блокам, представляющим традиционную схему работы, добавится новый блок «Интеграция данных» (Рисунок 1.40).

Рассмотрим каждый функциональный блок подробно с учетом предлагаемых изменений.

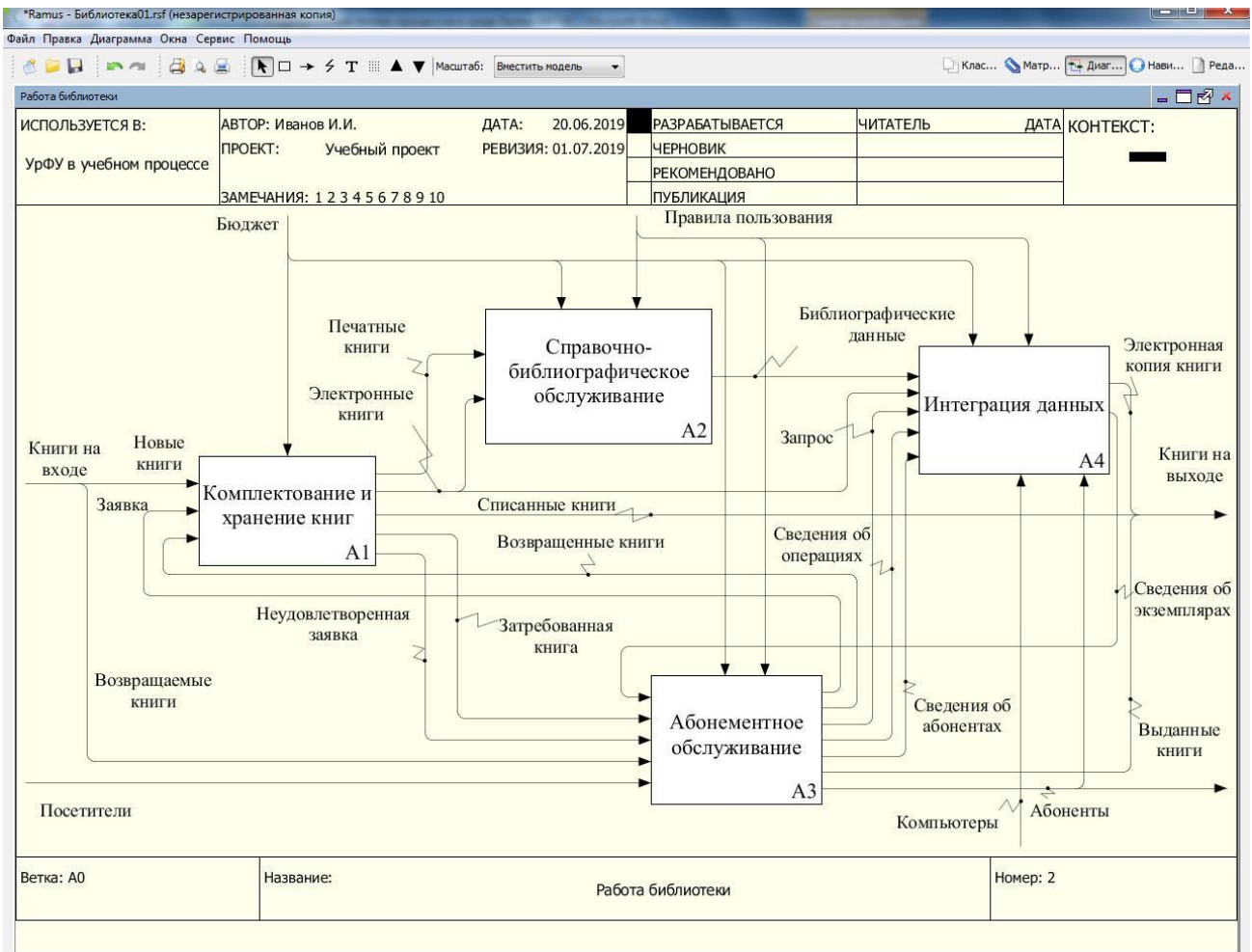


Рисунок 1.40 - Диаграмма декомпозиции контекстного блока модели ТО-ВЕ

Начнем с блока «Комплектование и хранение книг». Изменения будут касаться выходов блока. Комплектование в новой редакции предполагает приобретение библиотекой не только печатных, но электронных изданий.

Поэтому на выходе этого блока появятся стрелки «Печатные книги» и «Электронные книги». Причем «Электронные книги» вводятся в базу данных сразу как вход блока «Интеграция данных».

Эти же стрелки являются входами блока «Справочно-библиографическое обслуживание». А результатом (выходом) этого блока являются «Библиографические данные», актуальные для обоих типов книг. Эти сведения в совокупности с инвентарными номерами экземпляров книг вводятся в базу данных как вход блока «Интеграция данных».

Входами блока «Абонементное обслуживание» являются в основном те же данные (стрелки входа). Выходами этого блока являются не только

реальные люди («Абоненты») и выданные материальные книги («Выданные книги»), но и сведения о них. Сведения об абонентах также как и библиографические данные вводятся в базу данных. Стрелка «Сведения об абонентах» в совокупности с «Библиографическими данными» используются в процедурах выдачи и возврата книг, а информация об этих процедурах в виде стрелки «Сведения об операциях» (кто взял, какую книгу, когда вернул и т.д.) также вводятся в базу данных. Запрос о наличии интересующей книги (стрелка «Запрос») адресуется не в справочно-библиографический отдел, а напрямую в базу данных, чем ускоряется поиск книги.

Стрелки входа блока «Интеграция данных» мы рассмотрели, как выходы других блоков. Выходами использования базы данных являются «Электронная копия книги» и «Сведения об экземплярах». Первый представляет собой один из вариантов «Книги на выходе», поэтому эту стрелку следует слить с другими однородными стрелками. «Сведения об экземплярах» представляет информацию о каждом переплете книг в текущий момент времени и как реакция на «Запрос» может фигурировать в качестве входа «Абонементного обслуживания» для принятия решения о повторном поиске или оформлении заявки в книгохранилище.

В качестве управления для всех блоков используются те же стрелки «Бюджет» и «Правила пользования».

Исполнителями всех работ является «Персонал», участие которого неявно подразумевается. Ресурсами работы с базой данных, кроме того, являются «Компьютеры» и «Абоненты».

Детализация блоков «Комплектование и хранение книг» и «Справочно- библиографическое обслуживание» на диаграммах декомпозиции второго уровня интереса не представляют, т.к. они мало отличаются от предыдущих вариантов.

В блоке «Абонементное обслуживание» произошли изменения, поэтому рассмотрим его структуру подробно (Рисунок 1.41). Рассмотрим только существенные изменения.

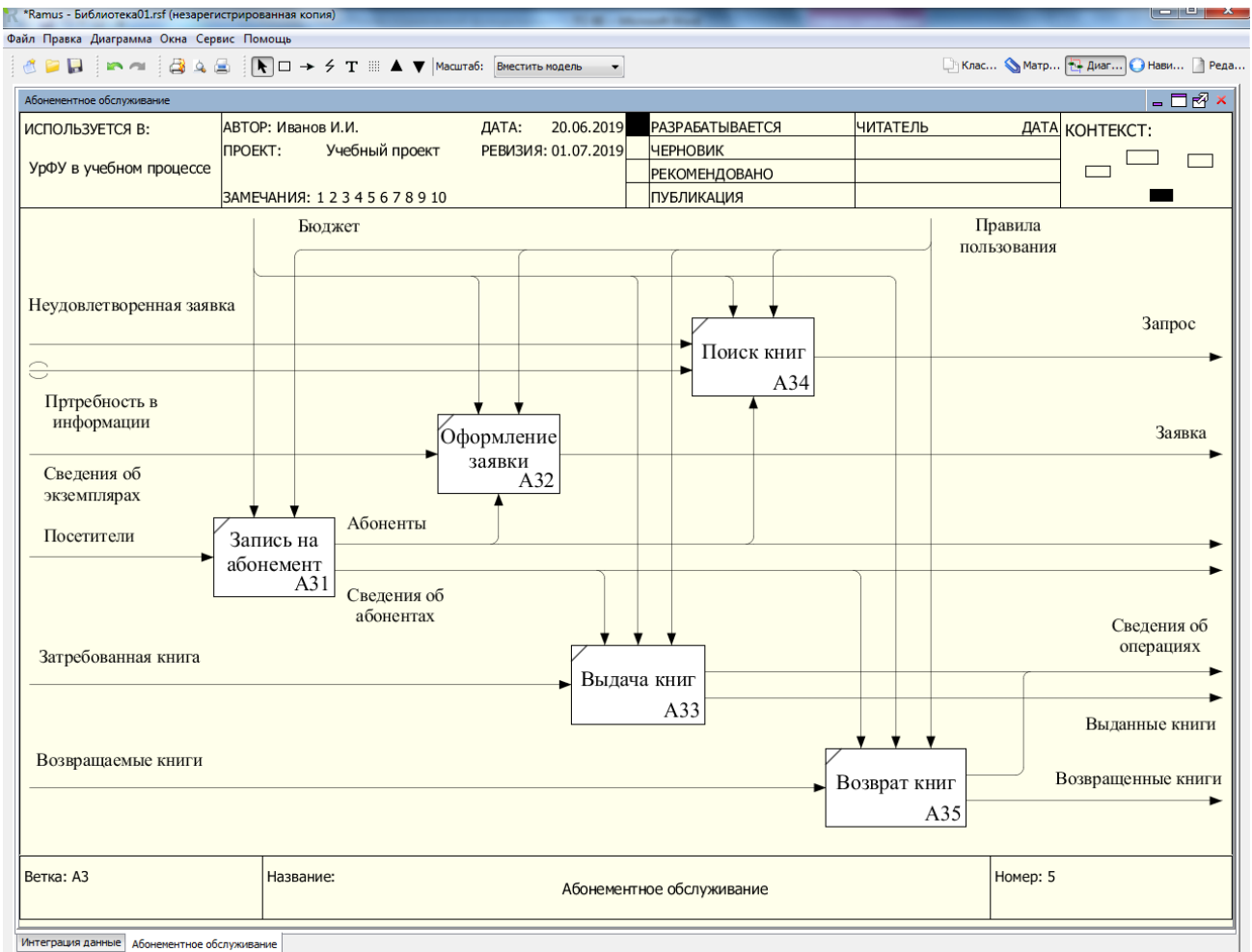


Рисунок 1.41 - Диаграмма декомпозиции блока «Абонементное обслуживание» модели TO-VE

В блоке «Поиск книг» на основании потребности в информации и/или неудовлетворенной заявки формируется «Запрос», который теперь реализуется в базе данных.

«Оформление заявки» выполняется на основе «Сведений об экземплярах», которые извлекаются из базы данных. При отсутствии электронного варианта книги «Заявка» направляется в книгохранилище.

«Запись на абонемент» не только меняет статус потенциального пользователя библиотеки, но и формирует «Сведения об абонентах», которые также передаются в базу данных. «Абоненты» играют роль исполнителей «Поиска книг» и «Оформления заявки». «Сведения об абонентах» проявляются как управление в функциях «Выдача книг» и «Возврат книг».

«Затребованные книги» в блоке «Выдача книг» превращаются в «Выданные книги». А «Возвращаемые книги» в блоке «Возврат книг» превращаются в «Возвращенные книги». И в том, и в другом блоке кроме материальных результатов в виде переплетов книг формируются «Сведения об операциях», которые также поступают в базу данных.

Диаграмма декомпозиции нового блока «Интеграция данных» приведена на Рисунке 1.42.

Детализация этого блока предполагает выполнение как минимум трех функций, которые представлены соответствующими блоками.

«Регистрация книг» предполагает ввод в базу данных электронных книг и библиографических данных о книгах. «Электронная копия книги» может быть при желании извлечена абонентом для удовлетворения его запроса. «Данные о книгах» играют роль управления при выполнении операций выдачи и возврата книг.

«Регистрация абонентов» предполагает ввод «Сведений об абонентах» в базу данных. «Данные об абонентах» как результат этого блока играют роль управления при выполнении операций выдачи и возврата книг.

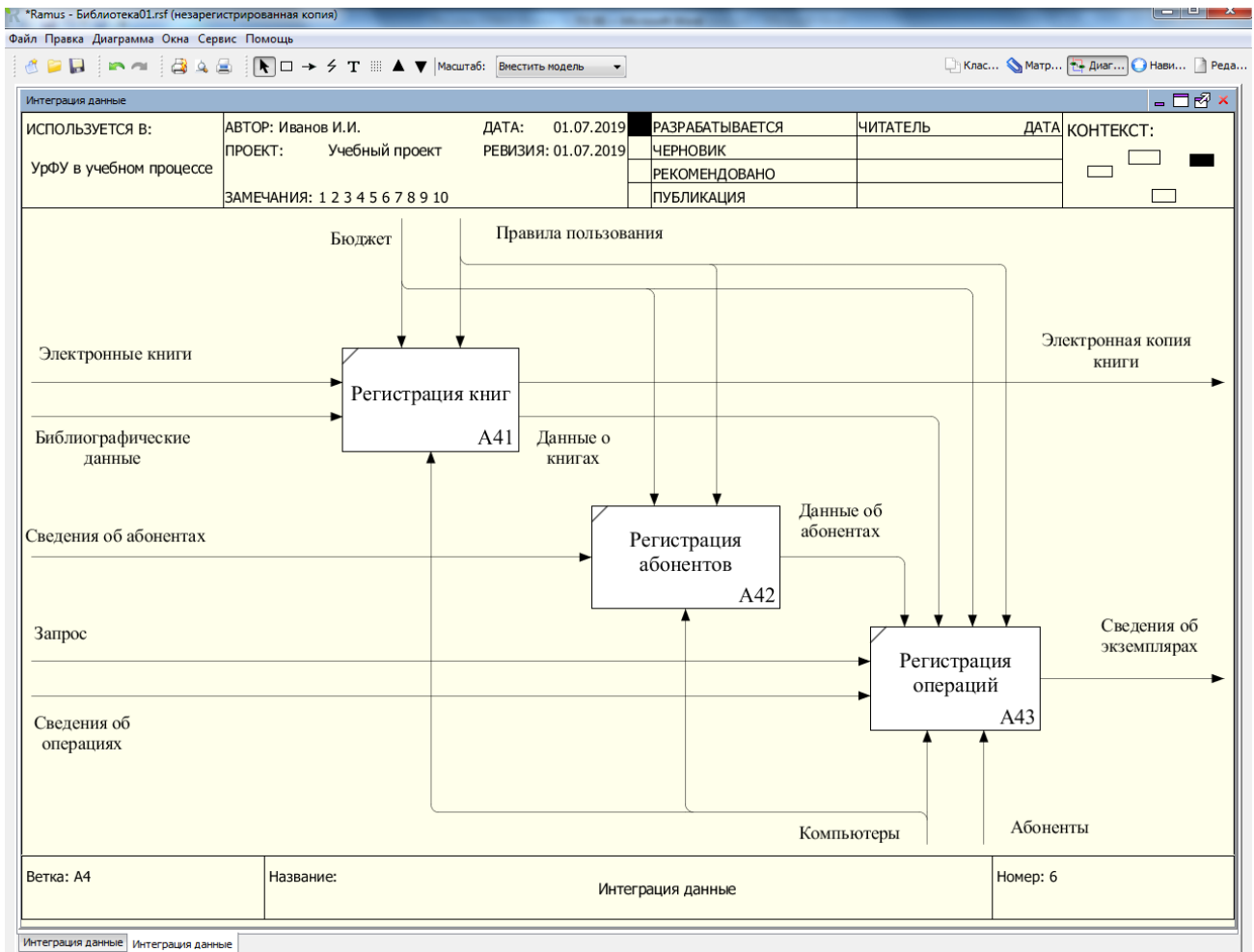


Рисунок 1.42 - Диаграмма декомпозиции блока «Интеграция данных» модели ТО-ВЕ

Для блока «Регистрация операций» вход «Сведения об операции» (тип, дата и другие) вводятся в базу данных и меняют статус соответствующего экземпляра книги («свободен» или «выдан»). По «Запросу» абонента «Сведения об экземплярах» будут доступны запрашивающему. В качестве исполнителей функции регистрации операций выступают сотрудники абонементов и абонент. По запросу последнего из базы данных извлекаются данные о состоянии экземпляров книг, в частности имеется ли в библиотеке затребованная книга, доступны ли ее экземпляры для выдачи на абонементе. Средством реализации всех функций «Интеграции данных» являются компьютеры, которые в виде одноименной стрелки механизма подходят снизу ко всем блокам.

Работа библиотеки в соответствии с ее усовершенствованной моделью может осуществляться примерно по таким сценариям.

Сотрудники отдела комплектования заносят в базу данных приобретенные электронные книги и передают учетные инвентарные номера приобретенных экземпляров печатных книг в справочно-библиографический отдел.

Сотрудники справочно-библиографического отдела заносят в базу данных библиографические данные и инвентарные номера экземпляров печатных книг с указанием их статуса («свободен»).

Абонент обращается с запросом к базе данных за сведениями об интересующих его книгах. Это может быть или одна конкретная книга или подмножество тематически связанных книг. Если имеется электронная версия книги, абонент копирует ее и на этом процедура заканчивается. Если имеется свободный экземпляр печатной книги, абонент на абонементе оформляет заявку в книгохранилище и получает затребованную книгу после процедуры оформления выдачи и занесения в базу данных сведений об изменившемся статусе экземпляра выданной книги.

### **Задание**

С помощью CASE-средства Ramus разработать иерархические функциональные модели следующих предметных областей:

1. Прохождение медосмотра (с точки зрения поликлиники);
2. Устройство на проживание в общежитии;
3. Устройство на работу (с точки зрения устраивающегося)
4. Постановка автомобиля на учёт (снятие авто с учёта)
5. Получение прав на вождение автомобиля (с точки зрения задействованных субъектов);
6. Обеспечение проживания в общежитии (со стороны администрации);
7. Деятельность магазина города;
8. Деятельность предприятия города;
9. Организация и проведение свадьбы;
10. Деятельность ЖКХ.

11. Внедрение системы 1с Предприятия в организации. С точки зрения внедряющей организации.
12. Деятельность библиотеки.
13. Деятельность студенческой столовой.
14. Деятельность кинотеатра.
15. Деятельность проходной.

### **Контрольные вопросы**

1. Для чего выполняется функционально-структурный анализ предметной области?
2. Какое предназначение имеет методология IDEF0?
3. Какая фундаментальная концепция лежит в основе методологии IDEF0?
4. Какие типы диаграмм формируют функциональную модель IDEF0?
5. Какие факторы определяют контекст функционального моделирования?
6. Какие конструктивные элементы используются при построении диаграмм IDEF0 и чему они соответствуют в реальной действительности?
7. Какие типы стрелок (дуг) представляют на диаграмме взаимодействие предметной области с окружающей средой и какое значение они имеют?
8. Какие типы стрелок (связей) функциональных блоков могут присутствовать на диаграммах?
9. Какой смысл имеют слияние и разветвление стрелок и как этот смысл передается в наименовании стрелок?
10. Что такое «туннелирование стрелок» и какие типы туннелирования бывают?
11. Что такое «модель AS-IS» и «модель TO-BE», чем они отличаются?
12. Какие CASE-системы используются для реализации методологии IDEF0?



## ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №2. ПРОВЕДЕНИЕ СТОИМОСТНОГО АНАЛИЗА ПРОЦЕССА

**Цель работы:** получить практические навыки в оценке бизнес-процесса по методологии функционально-стоимостного анализа с помощью пакета BPWin.

### Порядок выполнения работы.

#### ***1. Выбор задания, моделирование бизнес-процесса.***

Выберите бизнес-процесс, стоимостные характеристики которого Вы будете анализировать. Это может быть процесс, для которого Вы создавали IDEF0-модель в предыдущем семестре. Используйте ранее созданную IDEF0-модель процесса или создайте новую. Модель должна содержать не менее трех уровней.

#### ***2. Знакомство с основами функционально-стоимостного анализа***

Функционально-стоимостной анализ или стоимостный анализ, основанный на работах (Activity Based Costing, ABC), является широко распространенной методикой, используемой международными корпорациями и государственными организациями для идентификации истинных источников затрат в организации.

Стоимостный анализ представляет собой соглашение об учете, используемое для сбора данных о затратах, связанных с работами, с целью определить общую стоимость процесса. Стоимостный анализ основан на модели работ, потому что количественная оценка невозможна без детального понимания функциональности предприятия. Обычно ABC применяется для того, чтобы понять происхождение выходных затрат и облегчить выбор нужной модели работ при реорганизации деятельности предприятия. С помощью стоимостного анализа можно решить такие задачи, как: определение действительной стоимости производства продукта; определение действительной стоимости поддержки клиента; идентификация работ, которые стоят больше всего (те, которые должны быть улучшены в первую очередь); обеспечение менеджеров финансовым обоснованием предлагаемых изменений и др.

АВС включает следующие основные понятия:

- объект затрат – результат ("*Готовое изделие*", Рисунок 2), ради которого работа выполняется, обычно, основной выход работы, суммарная стоимость объектов затрат есть стоимость работ;

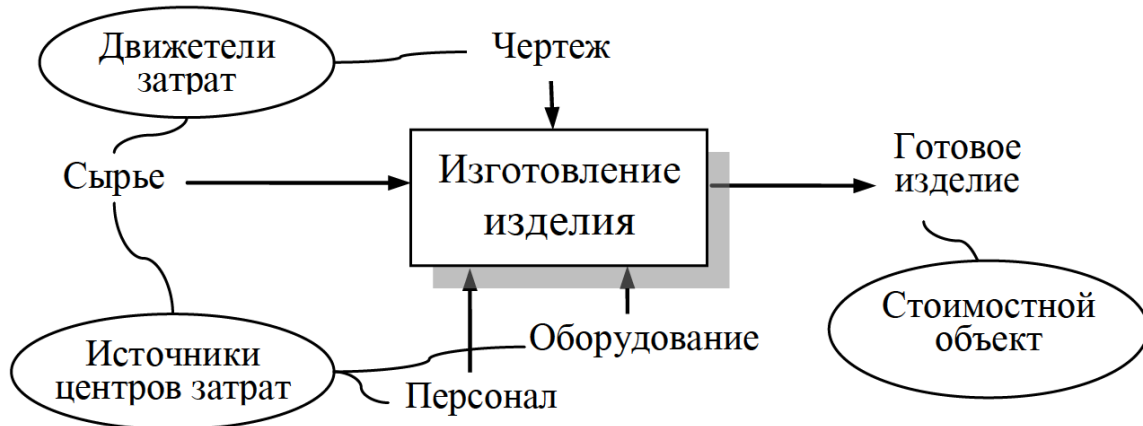


Рисунок 2 - Иллюстрация терминов АВС

- движители затрат – характеристики входов и управлений работы ("*Сырье*", "*Чертеж*", Рисунок 2), которые влияют на то, как выполняется и как долго длится работа; - центры затрат, которые можно трактовать как статьи расхода.

### 3. Задание единиц измерения и центров затрат

При проведении стоимостного анализа в ВРwin сначала задаются единицы измерения времени и денег. Для задания единиц измерения следует вызвать диалог Model Properties (меню Edit/Model Properties), вкладка ABC Units (Рисунок 2.1).

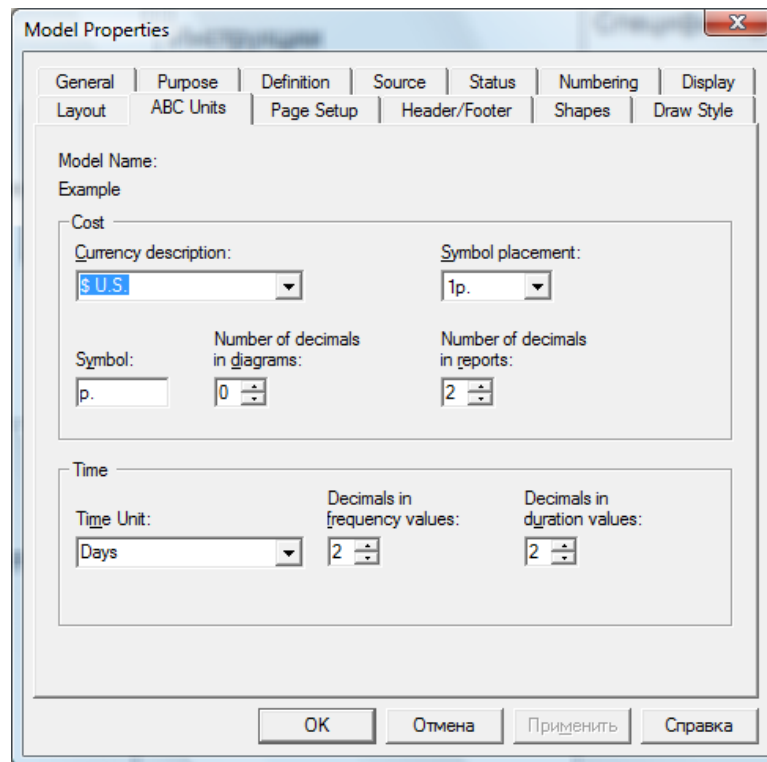


Рисунок 2.1- Настройка единиц измерения валюты и времени

Если в списке выбора отсутствует необходимая валюта (например, рубль), ее можно добавить. Символ валюты по умолчанию берется из настроек Windows.

Диапазон измерения времени в списке Time Unit достаточен для большинства случаев – от секунд до лет.

Затем описываются центры затрат (cost centers). Это стандартные категории расходов, общие для всех работ (функциональных блоков). Они включают в себя расходы на используемые ресурсы, представленные как входные дуги, дуги управления и механизмов.

Примеры центров стоимости:

- Рабочая сила – зарплата исполнителей работы;
- Оборудование – амортизационные отчисления за используемое оборудование;
- Помещение – оплата за используемое помещение;
- Материалы – оплата расходных материалов, комплектующих;
- Управление – затраты на управление (составление графика работ, планирование и т.д.).

Для внесения центров затрат необходимо выбрать в меню пункт Model /Cost Center Editor, откроется диалог Cost Center Editor (Рисунок 2.2). Добавить новый центр затрат можно с помощью кнопки Add, для редактирования выделенного центра служит кнопка Update, для удаления – кнопка Delete. Каждому центру затрат следует дать подробное описание в окне Definition. После того как введете все центры, закройте окно по кнопке Close.

Можно задать центры затрат и в справочнике Cost Center Dictionary (меню Dictionary /Cost Center), но в нем русские шрифты могут отображаться некорректно.

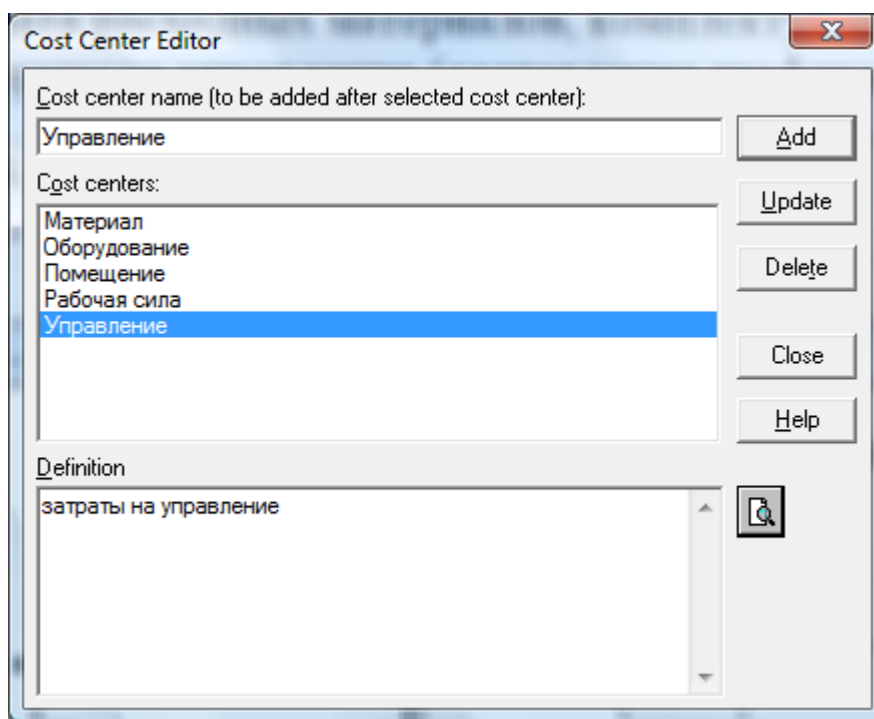


Рисунок 2.2 - Диалог Cost Center Dictionary

#### ***4. Расчет стоимости работ***

Для задания стоимости работы (для каждого функционального блока на диаграмме декомпозиции) следует щелкнуть правой кнопкой мыши по работе и на всплывающем меню выбрать Costs. Откроется диалог Activity Properties (Рисунок 2.3). Во вкладке Costs для каждого центра затрат нужно ввести сумму, затрачиваемую на данную работу по соответствующей статье расходов. Общие затраты рассчитываются как сумма по всем центрам затрат. На этой же вкладке указывается

частота проведения данной работы в рамках общего процесса (Frequency) и продолжительность (Duration). Затем нажмите кнопку Применить.

| Cost Center  | \$ U.S. |
|--------------|---------|
| Материал     | 500,00  |
| Оборудование | 30,00   |
| Помещение    | 50,00   |
| Рабочая сила | 550,00  |
| Управление   | 70,00   |

This Activity has NO Decomposition.

☐ Override decompositions      Total cost: 1 130,00  
☐ Compute from decompositions      Total cost x Frequency: 1 130,00

Frequency:      

Duration:  Days

Duration x Frequency      3,00 Days

OK    Отмена    Применить    Справка

Рисунок 2.3 - Задание стоимости работ в диалоге Activity Cost

Таким же образом задается стоимость остальных работ (функциональных блоков). Если в процессе назначения стоимости возникает необходимость внесения дополнительных центров затрат, диалог Cost Center Editor вызывается прямо из диалога Activity Cost соответствующей кнопкой.

На диаграммах в левом нижнем углу прямоугольника работы может показываться либо стоимость (по умолчанию), либо продолжительность, либо частота проведения работы (Рисунок 2.4). Настройка отображения осуществляется в диалоге Model Properties

(меню Model/Model Properties), вкладка Display, опции ABC Data и ABC Units.

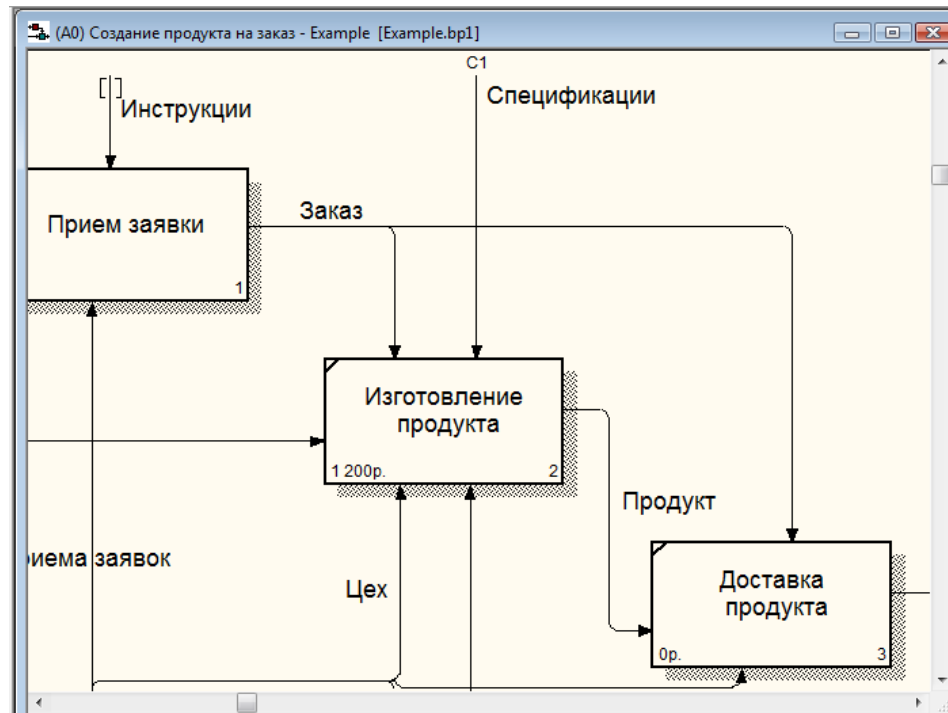


Рисунок 2.4 - Отображение стоимости работы

При вычислении затрат вышестоящей (родительской) работы сначала вычисляется произведение затрат дочерней работы на частоту работы (число раз, которое работа выполняется в рамках проведения родительской работы), затем результаты складываются. Если во всех работах модели включен режим Compute from Decompositions, подобные вычисления автоматически проводятся по всей иерархии работ снизу вверх (Рисунок 2.5). Аналогично вычисляется время выполнения родительской работы.

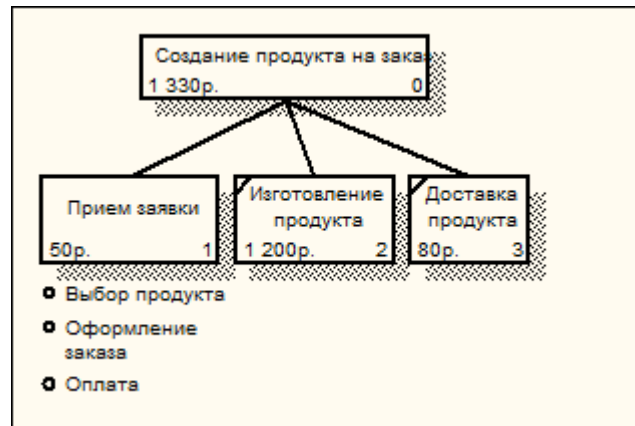


Рисунок 2.5 - Вычисление затрат родительской работы

Этот достаточно упрощенный принцип подсчета справедлив, если работы выполняются последовательно. Встроенные возможности BPwin позволяют разрабатывать упрощенные модели стоимости, которые, тем не менее, оказываются чрезвычайно полезными для предварительной оценки затрат. Если схема выполнения более сложная (например, работы производятся альтернативно), можно отказаться от подсчета и задать итоговые суммы для каждой работы вручную (Override Decompositions). В этом случае результаты расчетов с нижних уровней декомпозиции будут игнорироваться, при расчетах на верхних уровнях будет учитываться сумма, заданная вручную. На любом уровне результаты расчетов сохраняются независимо от выбранного режима, поэтому при выключении опции Override Decompositions расчет снизу вверх производится обычным образом.

Для проведения более тонкого анализа можно воспользоваться специализированным средством стоимостного анализа EasyABC (ABC Technology, Inc.). BPwin имеет двунаправленный интерфейс с EasyABC.

Задайте стоимости всех работ на самом нижнем уровне. Задание стоимости может выполняться в расчете: на единицу выпускаемой продукции (на обслуживание одного клиента); на партию продукции (на обслуживание группы клиентов); для вида продукции вне зависимости от количества (для вида услуги) в месяц (квартал, год).

Количество повторений работы в рамках выполнения родительской функции не всегда равно единице. Рассмотрим пример. На Рисунок

2.6 приведена диаграмма декомпозиции работы «Контроль качества». После каждого из этапов контроля (внешний осмотр, пробное включение, испытание на стенде) часть изделий уходит в брак. Допустим, вероятность выявления брака на всех этапах одинакова – 50%. Тогда при проверке партии из восьми изделий количество повторений:

- работы «Внешний осмотр» – 8,
- работы «Пробное включение» – 4 (половина партии уже ушла в брак),
- работы «Испытание на стенде» – 2 (еще половина ушла в брак).

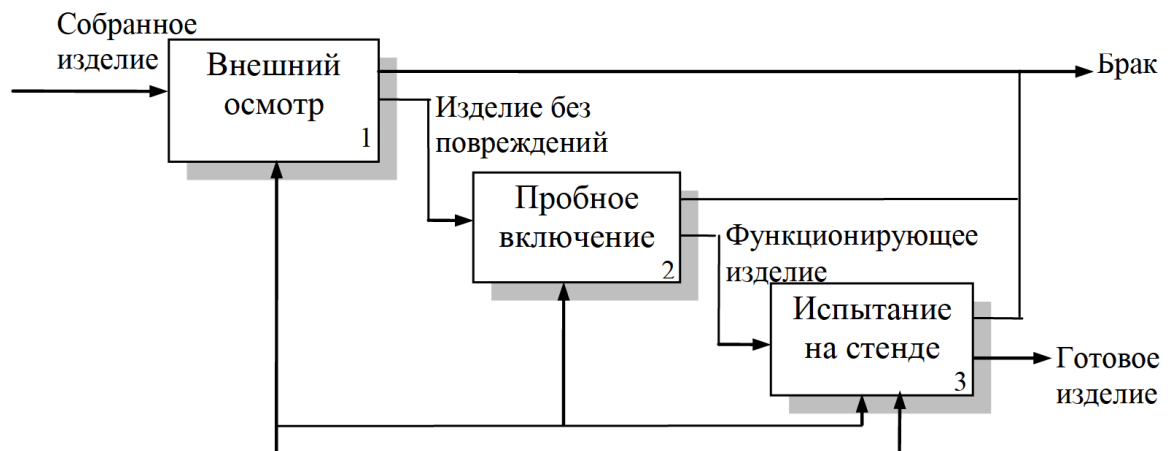


Рисунок 2.6 - Фрагмент диаграммы декомпозиции работы «Контроль качества»

После определения стоимости работ нижнего уровня посмотрите результат автоматического подсчета стоимости работы верхнего уровня. Создайте диаграмму дерева узлов (меню Diagram/Add Node Tree) и посмотрите стоимости всех работ.

## 5. Формирование отчета по стоимости работ

Результаты стоимостного анализа наглядно представляются на специальном отчете BPwin – Activity Cost Report (меню Tools/Report/Activity Cost Report). Отчет позволяет документировать имя, номер, определение и стоимость работ, как суммарную, так и отдельно по центрам затрат (Рисунок 2.7).



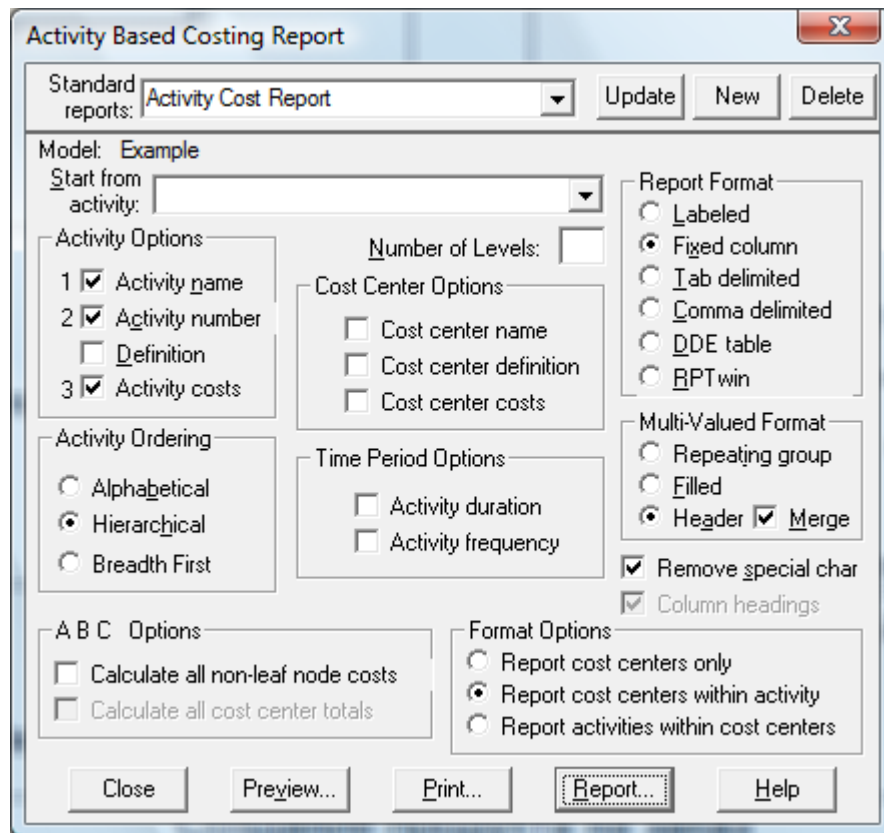


Рисунок 2.7 - Диалог настройки отчета по стоимости работ

Выберите в поле Standard reports – Activity Cost Report, нажмите кнопку Report..., задайте имя файла (с расширением txt), который будет содержать сгенерированный отчет. Откройте файл и просмотрите отчет (Рисунок 2.8).

| Activity Name              | Activity Number | Activity Cost (\$ U.S.) |
|----------------------------|-----------------|-------------------------|
| Создание продукта на заказ | 0               | 1 330,00                |
| Прием заявки               | 1               | 50,00                   |
| Выбор продукта             | 11              | 15,00                   |
| Оформление заказа          | 12              | 20,00                   |
| Оплата                     | 13              | 15,00                   |
| Изготовление продукта      | 2               | 1 200,00                |
| Доставка продукта          | 3               | 80,00                   |

Рисунок 2.8 - Отчет Activity Cost Report

### ***6. Завершение функционально-стоимостного анализа***

Завершите ABC - анализ бизнес-процесса, выбранного вами на шаге 1 в качестве индивидуального задания. Должны быть заданы стоимости всех работ на самом нижнем уровне. Стоимости всех работ должны отображаться на диаграммах и в дереве узлов. Должен быть сформирован отчет, отображающий стоимости всех работ.

### **Контрольные вопросы**

1. Какие виды инструментов для оценки модели вы знаете?
2. Что такое объект затрат?
3. Что такое движитель затрат?
4. Что такое центры затрат?
5. Для чего предназначены свойства, определяемые пользователем?
6. Кратко описать типы свойств
7. Как рассчитываются общие затраты по работе?

## **ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №3. РАБОТА С КОМАНДНЫМ ИНТЕРФЕЙСОМ СИСТЕМЫ 1С ПРЕДПРИЯТИЕ**

**Цель работы:** изучить основные принципы построения командного интерфейса 1с: Предприятия.

### **Теоретические положения**

#### ***1. Создание отдельного решения на основе платформы 1с: Предприятия.***

В архитектуре программного комплекса 1с: Предприятия выделяются следующие составляющие:

- платформа;
- информационная база;
  - база данных;
  - конфигурация.

Платформа – набор компонентов, обеспечивающих возможность работы с системой. Информационная база – это конкретное описание того, какие данные будут использоваться, алгоритмов работы с данными и сами данные. Одна платформа может быть связана с несколькими информационными базами.

В информационной базе совокупность описаний используемых данных, алгоритмов работы с ними, используемых диалоговых и печатных форм выделяется под понятием конфигурация.

Для создания одного прикладного решения, то есть экземпляра программного комплекса, обеспечивающего решение отдельной задачи по учёту деятельности, необходимо:

- Установить платформу
- Создать информационную базу.

Так как с одной платформой может быть связано несколько информационных баз, то можно сказать, что одно решение определяется отдельной информационной базой.

Установка платформы производится на основе отдельного экземпляра установочного пакета, обеспечиваемого ключом защиты.

Создание информационной базы производится из стартового меню 1с: Предприятия.

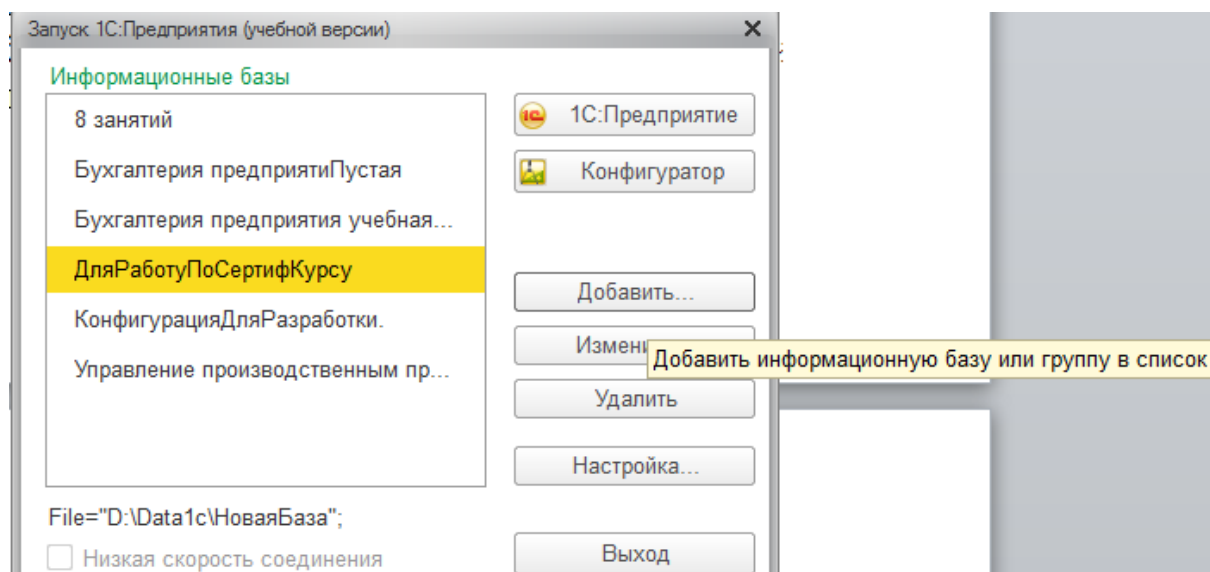


Рисунок 3 - Вызов команды создания новой информационной базы

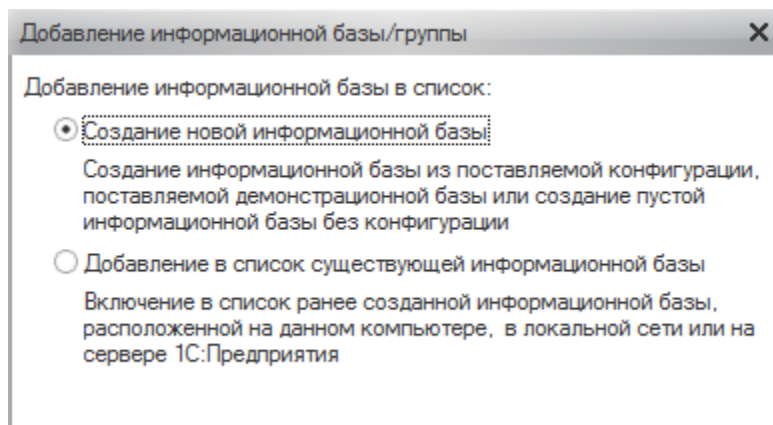


Рисунок 3.1 - Выбор создания информационной базы.

Создание новой информационной базы может быть выполнено на основе "шаблона", ранее созданное описание объектов информационной базы или без шаблона, "с чистого листа". Шаблоны должны быть загружены и зарегистрированы в системе предварительно. Для регистрации шаблонов используется команда окна запуска "Настройка". При этом откроется окно, в котором можно выбрать шаблон для регистрации. В данной работе будет рассматриваться случай создания новой информационной базы без шаблона. Для этого нужно выбрать соответствующий переключатель.

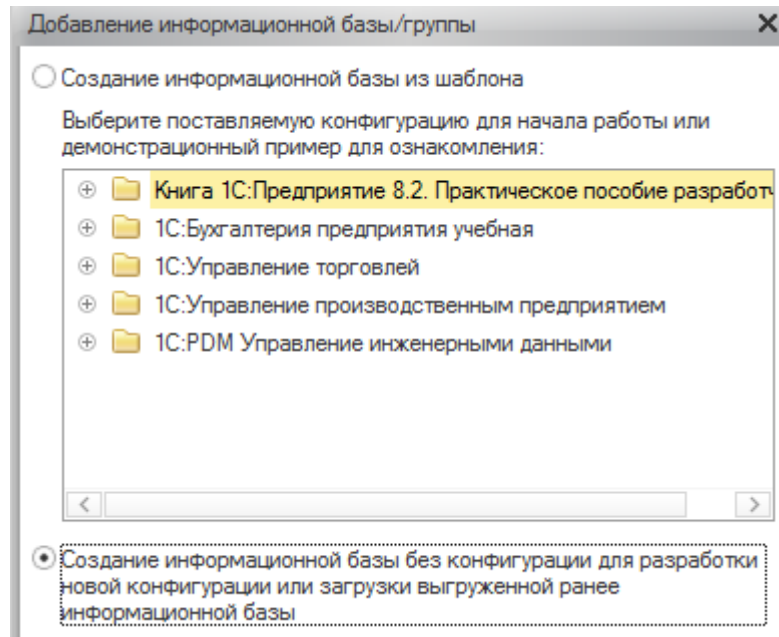


Рисунок 3.2 - Создание информационной базы без шаблона

Далее система предложит выбрать имя и расположение информационной базы.

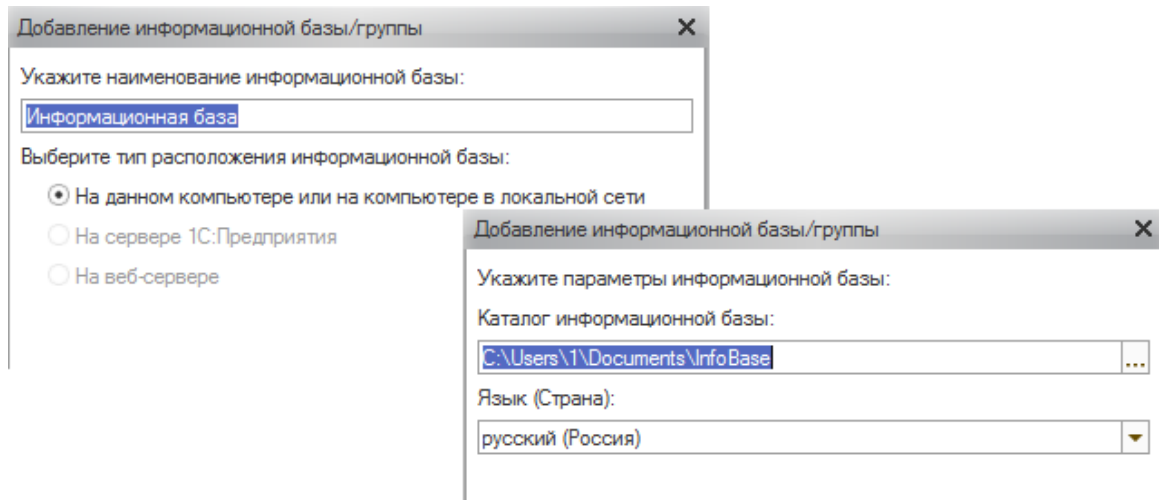


Рисунок 3.3 - Задание имя и расположения информационной базы.

После задания места и расположения информационной базы будет предложено задать параметры запуска информационной базы.

Рисунок 3.4 - Задание параметров запуска информационной базы.

Среди параметров запуска необходимо выделить, прежде всего режимы запуска. В данном случае режим запуска подразумевает какой вид клиента будет запускаться. В системе 1с: Предприятие начиная с версии 8.2 используется два вида клиента "Толстый клиент" и "Тонкий клиент". Каждый вид клиента подразумевает использование характерного для него режима работы.

Виды клиентов:

- "Толстый клиент" – обычный режим.
- "Тонкий клиент" – управляемое приложение.

## ***2. Режимы работы программного комплекса 1с: Предприятия.***

Приложение «1С: Предприятие» (версия 8.2,8.3) может работать в двух режимах:

Обычное приложение (режим, в котором работали предыдущие версии «1С: Предприятие 8»). Управляемое приложение (новый режим). формы в данном режиме построены и работают аналогично формам веб приложения. В настоящее время режим "Управляемого приложения" является основным.

В каком режиме будет работать приложение, может быть определено в конфигурации, при создании конфигурации или в панели свойств, как показано на рисунке.

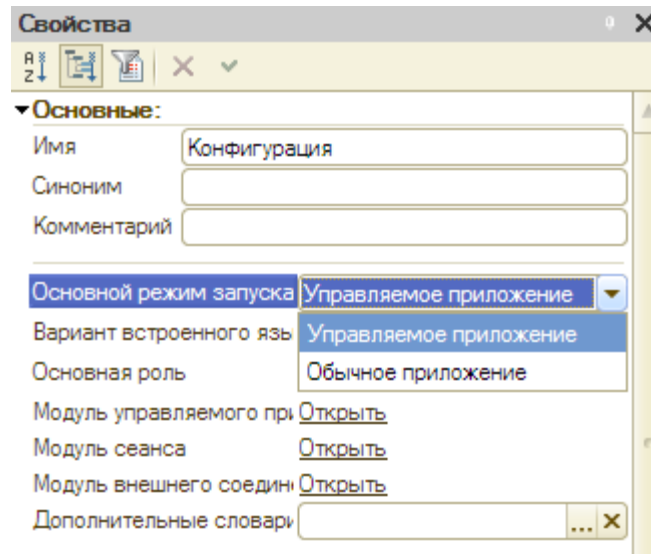


Рисунок 3.5 - Задание режимов запуска

Необходимо выбрать «Управляемое приложение». Однако, режим запуска также можно назначать каждому пользователю, также режим можно указывать в параметрах подключения информационной базы.

### ***3. Командный интерфейс программного комплекса 1с: Предприятие.***

Командный интерфейс – это основное средство доступа пользователя к функциональности приложения, средство, которое позволяет перемещаться между формами и выполнять те или иные действия. Одной из важных особенностей командного интерфейса является то, что он описывается декларативно. Разработчик не прорисовывает его в деталях, а просто описывает правила его формирования.

В интерфейсе могут быть выделены следующие составляющие  
Рабочий стол - на нём выполняется работа. Открываются документы, отображаются списки и т.д.

- Панель разделов – отображает выделенные в конфигурации функциональные группы, соответствует выделенным подсистемам. Кроме того, дополнительная кнопка – рабочий стол.
- Панель действий. Выводит для быстрого доступа команды, выполняемые в данном разделе. Для каждого раздела панель действий своя и может настраиваться.
- Панель навигации. – выводит списки основных объектов, используемых в текущем разделе. Для каждого раздела (подсистемы) панель навигации своя и может настраиваться.
- Главное меню. – через главное меню можно настроить панель навигации и панель действий для каждого раздела и настроить информацию, которая будет выводиться на рабочем столе. И многие другие режимы интерфейса.
- История – отображает историю работы – какие объекты были созданы, изменены, удалены (какая работа над какими объектами проводилась).
- Информационная панель – последний созданный документы.

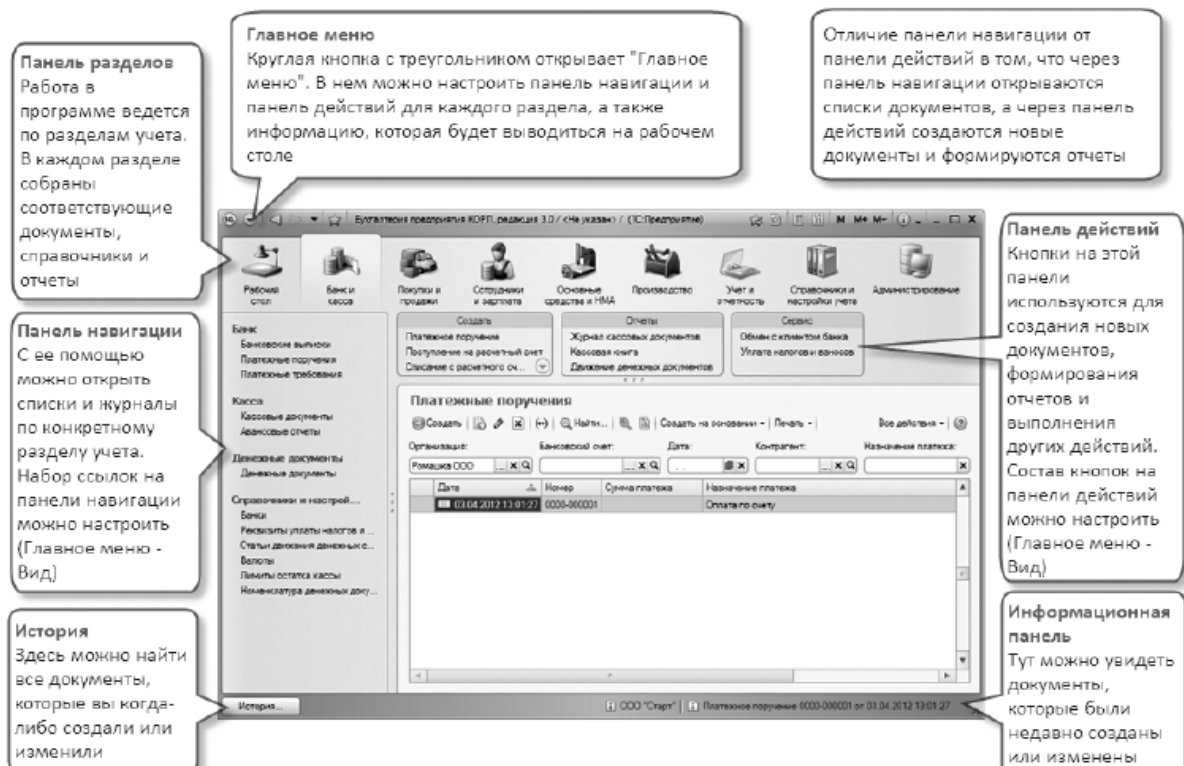


Рисунок 3.6 - Структура интерфейса 1с



Прежде всего, состав интерфейса определяется составом подсистем, определённых в системе. Определённые в системе подсистемы отображаются в виде подразделов панели разделов.

Структура подсистем определяет структуру функциональности прикладного решения. Можно сказать, что структура подсистем определяет, каким образом пользователь будет осуществлять «навигацию» по функциональности предлагаемого решения.

Подсистема является одним из объектов дерева конфигурации, отображается в узле "общие".

#### ***4. Конфигурация программного комплекса 1с: Предприятие.***

Под *конфигурацией* понимается описание структуры информационной базы, состав объектов, которые в неё входят, их особенности, поведение. Конфигурация в системе 1с Предприятие отображается в виде древовидной структуры – *дерева конфигурации*.

Объектом конфигурации является описание объекта, который, в последствии, будет содержаться в информационной базе. На основании объектов конфигурации создаются объекты рабочей среды.

Создавать объекты конфигурации можно или через контекстное меню соответствующего узла дерева, или через команду меню "Действия" дерева конфигурации.

Для просмотра свойств объекта конфигурации используется палитра свойств элемента, вызываемая через контекстное меню.

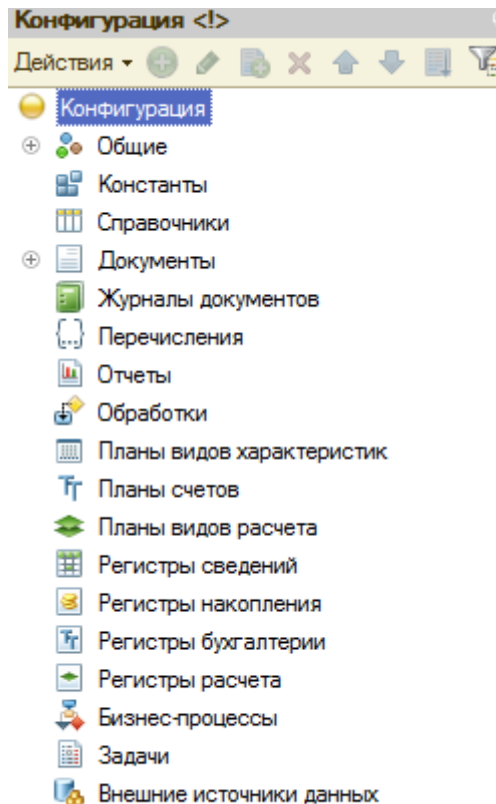


Рисунок 3.7 - Дерево конфигурации.

Для редактирования свойств элемента конфигурации используется "окно редактирования объекта конфигурации".

Для создания подсистем нужно зайти внутрь ветви общие и выполнить команду «Добавить» контекстного меню ветви «Подсистемы» (можно воспользоваться кнопкой командной панели окна дерева объектов конфигурации).

Обратите внимание на флаг «включать в командный интерфейс». Подсистемы со снятым флагом не влияют на структуру функциональности программного комплекса.

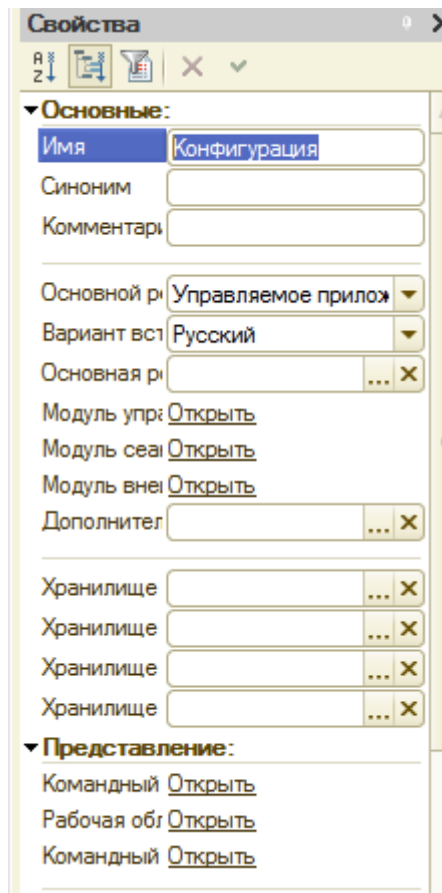


Рисунок 3.8 - Палитра свойств элемента

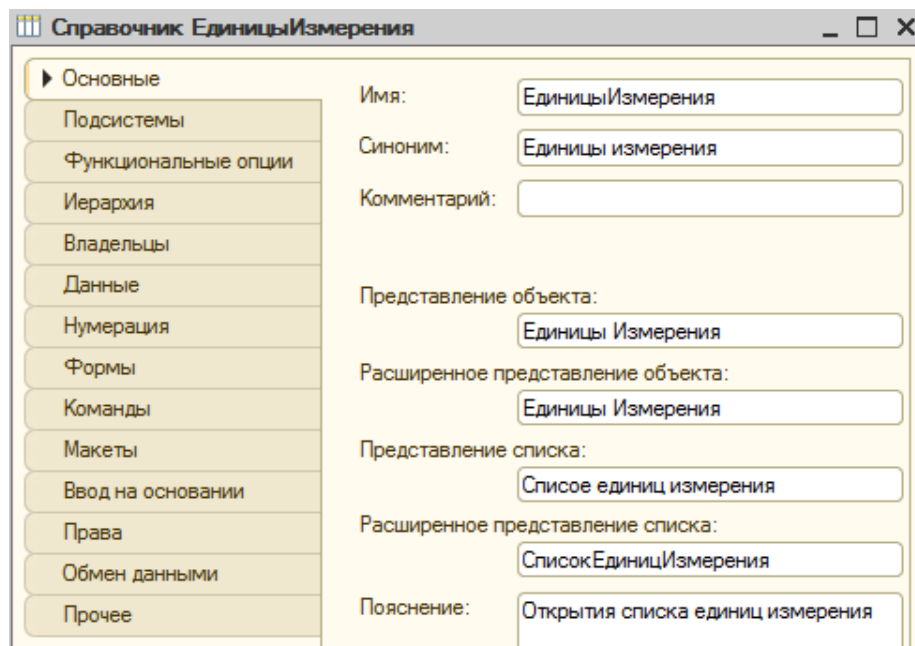


Рисунок 3.9 - Окно редактирования объекта конфигурации.

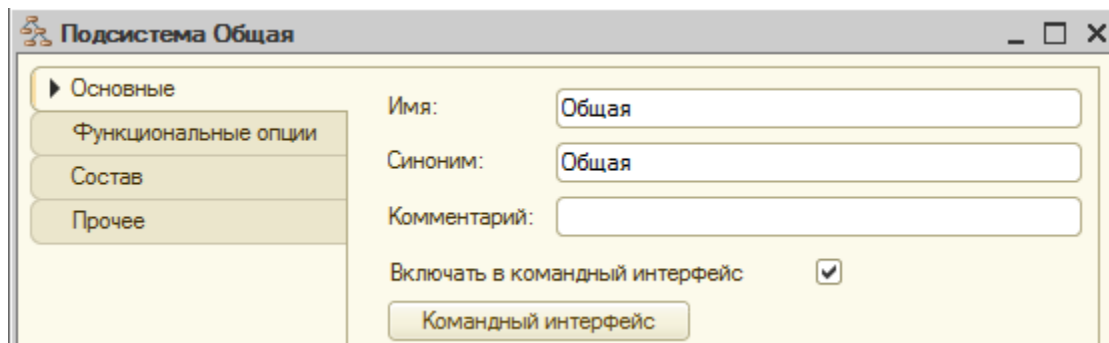


Рисунок 3.10 - Окно редактирования подсистемы

К подсистемам первого уровня можно создавать подчиненные подсистемы и так далее.

Подсистемы первого уровня определяют структуру так называемой "панели разделов" (при этом порядок разделов отличается от порядка, в котором созданы подсистемы в конфигурации. (Разделы отсортированы по алфавиту).).

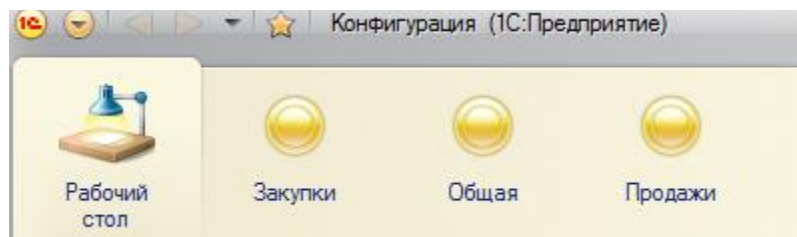


Рисунок 3.11 - Панель разделов

При создании объектов конфигурации они относятся к каким-либо подсистемам, настраиваются права доступа.

Таким образом, интерфейс системы описывается декларативно. «Внешний вид» программы формируется «на лету». Можно сказать, что интерфейс собирается как мозаика, и каждый элемент мозаики описывается в системе в определенных точках конфигурации. Для уточнения вида интерфейса контекстного меню корня дерева объектов конфигурации «Открыть командный интерфейс конфигурации» (либо в палитре свойств найти одноименное свойства и воспользоваться гиперссылкой).

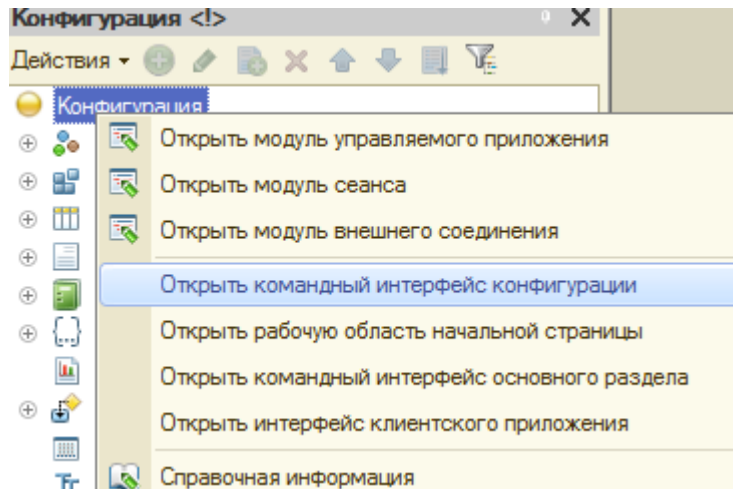


Рисунок 3.12 - Вызов команд редактирования командного интерфейса через контекстное меню.

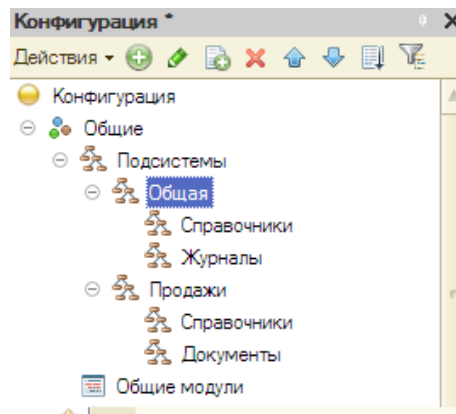


Рисунок 3.13 - Подсистемы в дереве конфигурации

## 5. Роли

С помощью ролей определяется доступность какой-либо функциональности определенной группе пользователей конфигурации. Соответственно этот состав определим исходя из возможного перечня таких групп (ролей, которые они будут выполнять при работе в данной конфигурации).

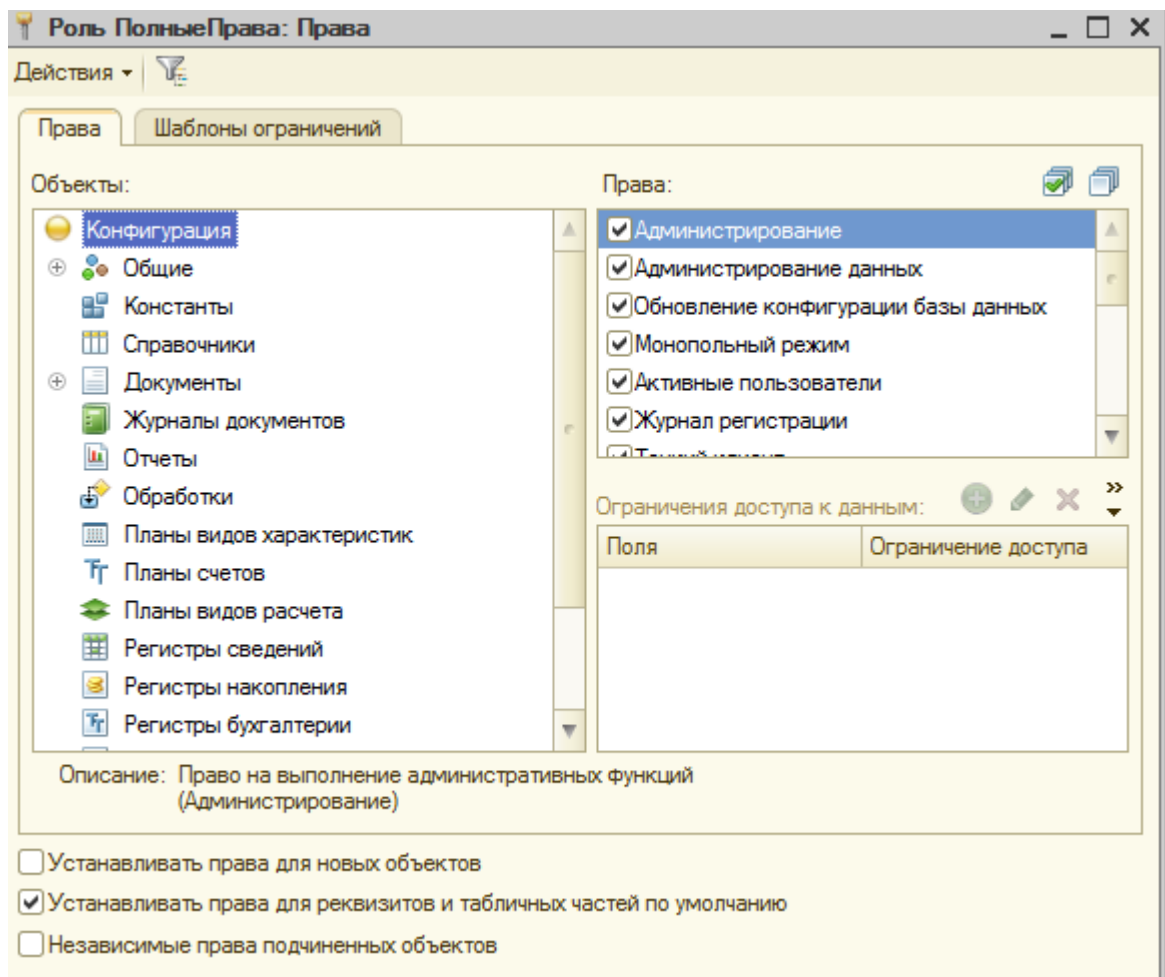


Рисунок 3.14 - Задание свойств роли

Следует отметить, что при работе с реальной конфигурацией нужно очень аккуратно относиться к такому элементу роли, как «Интерактивное удаление» (включение этого флага приводит к возможности удаления объектов без контроля ссылочной целостности)

Следующая важная составляющая командного интерфейса, это сами команды, но с ними будем знакомиться позже.

## 6. Константы

Константы используются для хранения значений, которые не меняются довольно длительное время. К ним можно отнести название организации (если учет в конфигурации ведется от имени одной организации), юридический адрес, фамилии ответственных лиц и т.д. Значение констант задаются и могут изменяться в режиме

Предприятия. Создаются константы, как и другие объекты конфигурации, на соответствующем узле дерева конфигурации.

Для отображения констант в системе по умолчанию создаются формы констант, позволяющие задавать и менять значение констант. Для каждой константы по умолчанию создаётся отдельная форма. Форму какой-либо константы можно увидеть в том разделе, к которому отнесена соответствующая константа в командной панели в группе команд "Сервис".

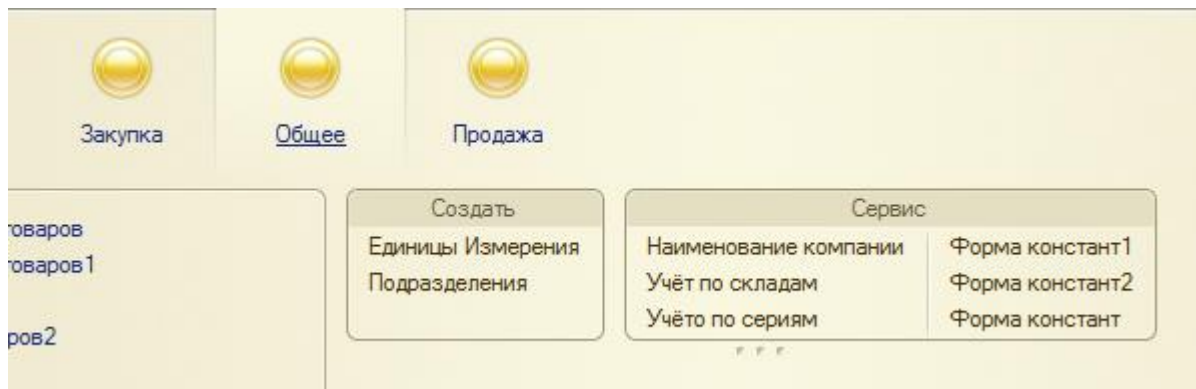


Рисунок 3.15 - Отображение на командной панели доступа к формам констант "Наименование компании", "Учёт по складам", "Учёт по сериям"

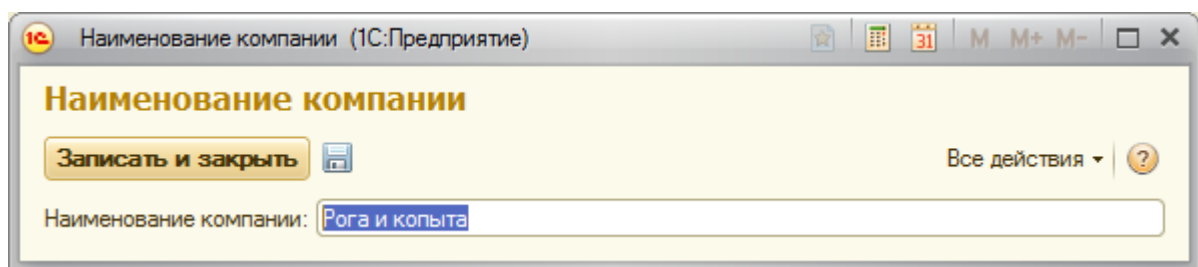


Рисунок 3.16 - Форма константы "Наименование предприятия"

### Задание

1. Создать новую информационную базу платформы 1С:Предприятие 8.2, без использования шаблонов
2. Создать подсистемы следующей структуры:
  - 2.1. Общая
    - 2.1.1. Справочники
    - 2.1.2. Журналы

- 2.2. Закупки
  - 2.2.1. Справочники
  - 2.2.2. Документы
- 2.3. Продажи
  - 2.3.1. Справочники
  - 2.3.2. Документы
- 3. Создать следующий состав ролей:
  - 3.1. ПолныеПрава (должны соответствовать своему наименованию)
  - 3.2. ОтделПродаж (предполагается, что у этой роли будет отсутствовать доступ к функциональности системы, связанной с проведением закупок).
  - 3.3. ОтделЗакупок (предполагается, что у этой роли будет отсутствовать доступ к функциональности системы, связанной с проведением продаж).
- 4. Следующие константы:
  - 4.1. Имя: «НаименованиеКомпании», тип «Строка»
  - 4.2. Имя: «УчетПоСериям», тип: «Булево»
  - 4.3. Имя: «УчетПоСкладам», тип «Булево»
- 5. Задать значение констант.

### **Контрольные вопросы**

1. Что из себя представляет архитектура 1с: Предприятия. Какие в ней выделяются составляющие.?
2. Какие виды клиента существуют в системе 1с: Предприятие? Какие режимы работы используются в каждом виде клиента?
3. Что из себя представляет отдельное решение на основе 1с: Предприятия?
4. Какие режимы запуска программного комплекса 1с: Предприятия существуют. Их особенности.
5. Какова структура интерфейса 1с: Предприятия. Как она определяется.
6. Что такое конфигурация системы 1с. Как представляется конфигурации.



7. Что такое объект конфигурации. Примеры объектов конфигурации. Как создаются объекты конфигурации.
8. Как просматриваются свойства объекта конфигурации.
9. Какие средства для редактирования свойств объекта конфигурации существуют.
10. Структура интерфейса 1сПредприятия. Как определяется структура интерфейса?
11. Для чего используется объект "подсистема".
12. Как описывать логическую структуру конфигурации при помощи объекта подсистема?
13. Что такое роль?
14. Что такое константа в среде 1с Предприятие? В каком режиме можно создать константу и задать значение константы?
15. Как можно увидеть значение константы в режиме предприятия?

## ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №4. РАБОТА С ПОЛЬЗОВАТЕЛЯМИ СИСТЕМЫ

### Общая информация

Каждый пользователь системы должен иметь свободный доступ к общей информации, такой как общие справочники, константы или перечисления.

С другой стороны, необходимо, чтобы каждый пользователь имел дело только с той информацией, которая необходима ему для работы, и никак не мог своими действиями повлиять на работу других пользователей или на работоспособность системы в целом.

Конфигуратор системы «1С: Предприятие» предоставляет разработчикам развитые средства администрирования, предназначенные для решения указанных задач.

Прежде всего, в процессе создания конфигурации создается необходимое число типовых ролей, описывающих полномочия различных категорий пользователей на доступ к информации, обрабатываемой системой. Роли могут быть заданы в достаточно широких пределах – от возможности только просмотра ограниченного числа видов документов до полного набора прав по вводу, просмотру, корректировке и удалению любых видов данных.

В системе «1С: Предприятие» различают два типа прав – основные и интерактивные. **Основные** проверяются всегда, независимо от способа обращения к объектам информационной базы. **Интерактивные** проверяются при выполнении интерактивных операций (просмотр и редактирование в форме и т. д.).

Если для объекта, данные которого представляются в форме, установлено право Просмотр, но не установлено право Редактирование, то в форме данный реквизит будет показан, но редактирование значения будет недоступно. Если снять право Просмотр, то при попытке открытия формы будет выдано предупреждение: Нарушение прав доступа! и форма не будет открыта.

В списке прав при редактировании роли следует обратить внимание на внутреннюю иерархию прав. Иерархия проявляется в виде

«старшинства» прав. При снятии «старшего» права снимаются другие права («младшие»), связанные со «старшим» правом; и наоборот, при установке «младшего» права устанавливаются снятые «старшие» права.

В общем случае права можно задавать: на всю конфигурацию в целом, объекты, реквизиты объектов, табличные части, реквизиты табличных частей, стандартные реквизиты.

При создании новой роли устанавливаются следующие права доступа на корневой объект конфигурации: ТонкийКлиент, ВебКлиент, СохранениеДанныхПользователя, Вывод.

### Ведение списка пользователей

Открыть окно «Список пользователей» можно с помощью пункта меню **Администрирование – Пользователи**.

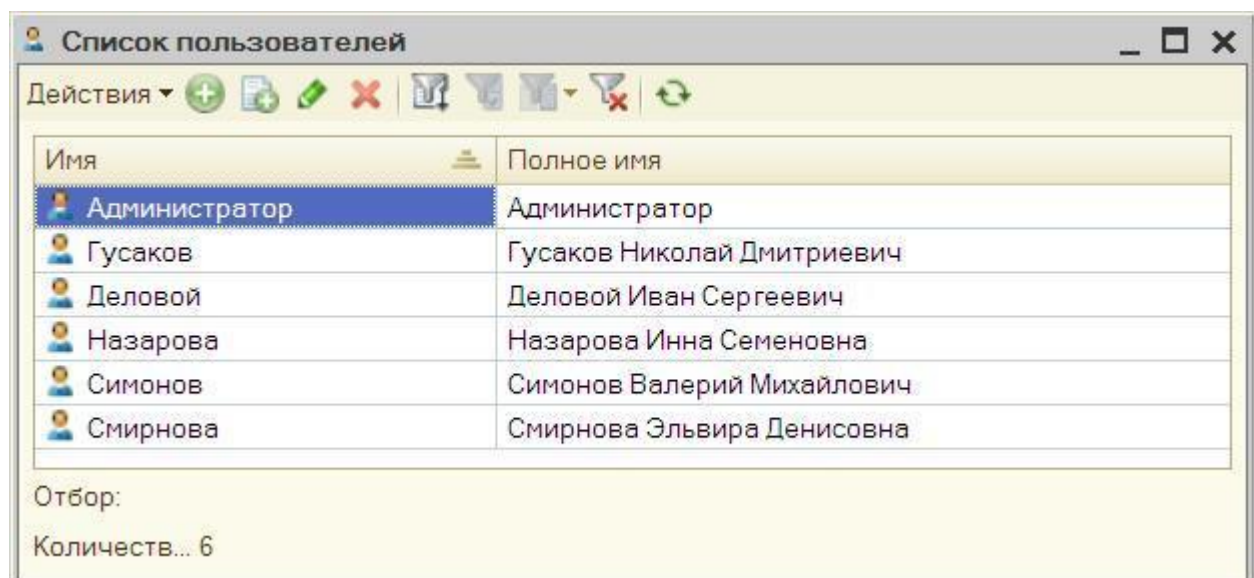


Рисунок 4 - Список пользователей

Окно со списком пользователей имеет панель инструментов и табличное поле с двумя колонками (Рисунок 4):

- в колонке **Имя** выводится список пользователей, зарегистрированных для работы с системой «1С: Предприятие 8».

- колонка **Полное имя** может содержать расшифровку имени, выданного в первой колонке.

Пользователи, для которых определен пароль доступа, отображаются пиктограммами с замочком.

Пользователи, для которых не определена роль или аутентификация, отображаются пиктограммами с вопросом.

С помощью пунктов меню **Действия** осуществляется ведение списка пользователей, настройка показа списка (отбор, состав и порядок колонок, сортировка), а также вывод списка в табличный или текстовый документы.

### *Добавление нового пользователя.*

Для добавления нового пользователя необходимо выбрать пункт **Действия – Добавить** в окне **Список пользователей**. На экран будет выдано окно для редактирования параметров пользователя (Рисунок 4.1).

На закладке **Основные** указывается имя и полное имя пользователя.

Пользователь

Основные Прочие

Имя:

Полное имя:

☒ Аутентификация 1С:Предприят...

Пароль:

Подтверждение паро...

Пользователю запрещено изменять пароль ☐

Показывать в списке выбора ☒

☐ Аутентификация операционной системы:

Пользователь:

☐ Аутентификация OpenID

OK Отмена Справка

Рисунок 4.1 - Новый пользователь

В имени пользователя не рекомендуется указывать символ ":". Уникальность пользователя информационной базы поддерживается по совокупности значений трех полей: имя, полное имя и имя пользователя операционной системы (если включена аутентификация средствами операционной системы). Для поля **Имя** уникальность поддерживается по первым 64 символам, для поля **Полное имя** – по первым 128 символам, для поля **Пользователь** (при включенной аутентификации средствами операционной системы) – по первым 128 символам. Рекомендуется не допускать для поля **Имя** превышения длины в 64 символа.

Для каждого пользователя обязательным является указание способа аутентификации.

Каждый из флажков «Аутентификация...» определяет возможность выполнения аутентификации для данного пользователя тем или иным способом. Эти флажки не влияют на порядок попыток аутентификации.

- Аутентификация «1С: Предприятия».

Пользователь может быть аутентифицирован системой «1С: Предприятие» с помощью ввода его имени и пароля (в диалоге аутентификации, в виде параметров командной строки или строки соединения с информационной базой для внешнего соединения или automation-сервера). В этом случае проверка наличия пользователя и корректности ввода его пароля выполняет система «1С: Предприятие».

- Аутентификация операционной системы.

Пользователь может быть аутентифицирован неявно средствами операционной системы. Для этого пользователю должен быть поставлен в соответствие некоторый пользователь операционной системы. При старте системы, «1С: Предприятие» запрашивает у операционной системы пользователя, который аутентифицирован в системе в данный момент. Для этого в ОС Windows используется интерфейс SSPI. Затем выполняется проверка, что данному пользователю операционной системы сопоставлен пользователь «1С: Предприятия».

Если поиск заканчивается успешно – считается, что пользователь системы «1С: Предприятие» аутентифицирован успешно, и диалог аутентификации не отображается.

Для обеспечения стабильной работы аутентификации ОС при подключении тонким клиентом через веб-сервер, необходимо внести адрес используемой информационной базы в доверенную зону веб-браузера Microsoft Internet Explorer.

**Примечание.** Клиентское приложение, работающее в ОС Linux, не поддерживает аутентификацию средствами операционной системы. Также такой вариант аутентификации не поддерживается в случае, если клиентское приложение подключается к информационной базе через веб-сервер Apache, работающий под управлением ОС Windows.

○ Аутентификация OpenID.

OpenID (<http://openid.net>) – это протокол, который позволяет пользователю использовать единую учетную запись для аутентификации на множестве не связанных друг с другом ресурсов, систем и т. д. Система «1С: Предприятие» использует протокол, созданный на основе протокола OpenID версии 2.0 по модели Direct Identity.

**Примечание.** Данный способ аутентификации не применим при обращении к веб-сервисам, опубликованным из «1С: Предприятия». В роли провайдера OpenID выступает информационная база «1С: Предприятия».

Общая схема работы выглядит следующим образом:

- Пользователь пытается выполнить вход в систему;
- Система определяет, что в информационной базе работает OpenID-аутентификация (по файлу публикации default.vrd);
- Провайдеру OpenID отправляется запрос на выполнение аутентификации;
- Если необходимо выполнить интерактивное действие (выполняется первая аутентификация для данного идентификатора или закончено время жизни признака аутентификации данного идентификатора), то провайдер сообщает системе о необходимости запросить имя и пароль

пользователя, система выполняет интерактивное действие и возвращает провайдеру OpenID запрошенные данные;

- Если провайдер аутентифицирует пользователя, то системе возвращается признак того, что пользователь аутентифицирован.

OpenID-аутентификация работает только в тех случаях, когда доступ к информационной базе осуществляется по протоколу HTTPS. Это означает, что использовать OpenID-аутентификацию могут только веб-клиент и тонкий клиент, подключенный к информационной базе через веб-сервер. При OpenID- аутентификации возможны кросс-доменные запросы при работе с помощью тонкого клиента, а также с помощью веб-браузеров Mozilla Firefox, Google Chrome, Safari и Microsoft Internet Explorer версий 8 и 9.

Если сняты все флажки, то данному пользователю запрещен доступ к прикладному решению.

Для того чтобы выполнялась попытка аутентификации с помощью протокола OpenID, необходимо, чтобы соответствующим образом была настроена публикация данной информационной базы на веб-сервере.

**Примечание.** В системе должен быть по крайней мере один пользователь, который обладает административными правами и допускает аутентификацию средствами «1С: Предприятия».

Если установлен флажок «Показывать в списке выбора», то данный пользователь будет отображаться в списке выбора при соединении с информационной базой системы «1С: Предприятие».

На закладке «**Прочие**» (Рисунок 4.2) указываются доступные роли и язык. Для одного пользователя могут быть указаны несколько ролей. Кроме того, для пользователя можно указать режим запуска «1С: Предприятия». Если используется значение **Авто**, то при запуске будет использоваться режим запуска, установленный в свойстве конфигурации **Основной режим запуска**.

В окне для редактирования свойств пользователя не обязательно заполнять сразу все поля – это можно сделать позднее путем редактирования параметров уже существующего пользователя.

Нового пользователя можно создать путем копирования существующего.

Для копирования нужно выбрать исходную строку списка пользователей и выполнить команду **Действия – Скопировать**.

Для удаления пользователя следует в списке пользователей выделить его имя и выбрать пункт **Действия – Удалить** окна **Список пользователей**.

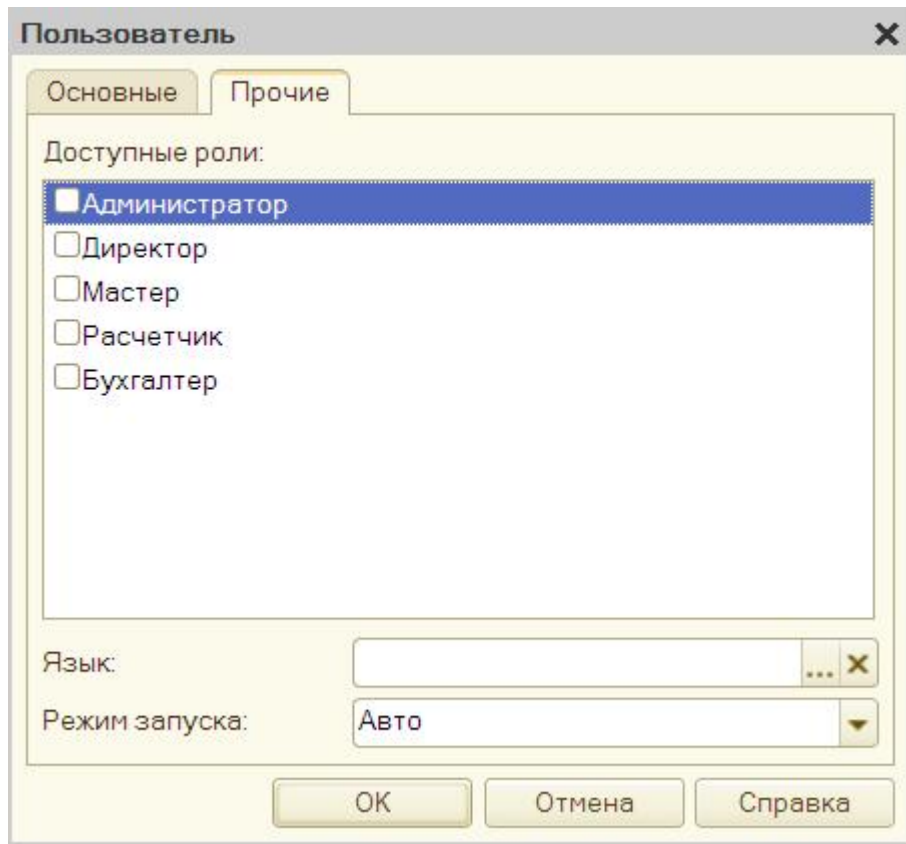


Рисунок 4.2 - Прочие параметры нового пользователя

### Список активных пользователей

В процессе работы бывает необходимо определить, какие пользователи работают в данный момент с информационной базой.

Для получения списка пользователей нужно выбрать пункт **Администрирование – Активные пользователи**. На экран выводится окно со списком пользователей, работающих в данный момент с базой данных.

С помощью пунктов меню **Действия** можно настроить показ списка, а также вывести его в табличный или текстовый документ.



Список активных пользователей можно сортировать по любой колонке.

### **Блокировка сеансов пользователей**

Система «1С: Предприятие» позволяет устанавливать блокировки сеансов пользователей с информационной базой. Можно запретить установку сеансов пользователей с информационной базой с отображением сообщения о причине запрета. Эта возможность полезна, например, когда для выполнения административных действий требуется, чтобы текущие пользователи завершили свои сеансы работы и в то же время новые пользователи не могли подключиться к информационной базе.

При работе в клиент-серверном варианте работы установка блокировки может быть выполнена с помощью утилиты администрирования кластера серверов «1С: Предприятие».

При работе в любом режиме установка блокировки также может быть выполнена средствами встроенного языка. Для этого используется объект встроенного языка *БлокировкаСеансов*, который можно создать с помощью конструктора и установить необходимые свойства блокировки установки соединений. Метод глобального контекста *УстановитьБлокировкуСеансов ()* позволяет установить, а метод *ПолучитьБлокировкуСеансов ()* – получить установленную блокировку.

### **Редактор прав доступа**

В левой части окна редактирования прав (Рисунок 4.3) выводится дерево объектов конфигурации по всем подсистемам. В правой – список прав по выбранному объекту конфигурации в дереве конфигурации. Если для действия установлен флажок, то оно разрешено.

Так, например, для пользователя с ролью *МенеджерПоПродажам* разрешен просмотр документа *ПриходТовара* и запрещено его интерактивное добавление.

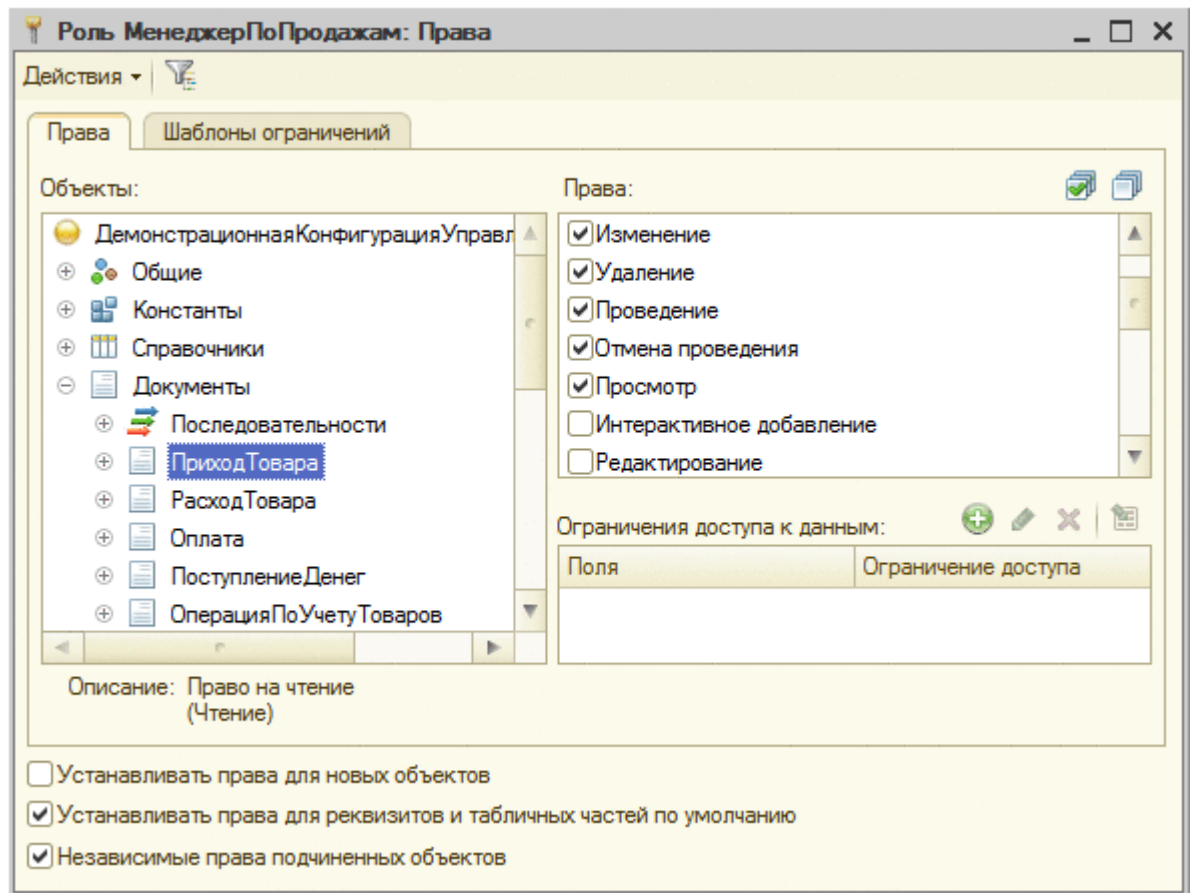


Рисунок 4.3 - Редактор прав доступа роли

Чтобы изменить право доступа, в левом списке следует выбрать объект конфигурации, а в правом изменить состояние флажка в нужной строке вида действия. Если требуется изменить доступ сразу ко всем объектам выбранной ветви, нужно указать в левой части эту ветвь и изменить установку прав доступа.

Описание каждой роли можно вывести в табличный или текстовый документ с помощью пункта Действия – Вывести список.

### Просмотр и редактирование всех ролей

Если в конфигурации используется несколько ролей, то для удобства просмотра и редактирования прав рекомендуется использовать окно «**Все роли**». Для его открытия в дереве объектов конфигурации окна Конфигурация нужно указать ветвь Роли и в контекстном меню выбрать команду «**Все роли**».

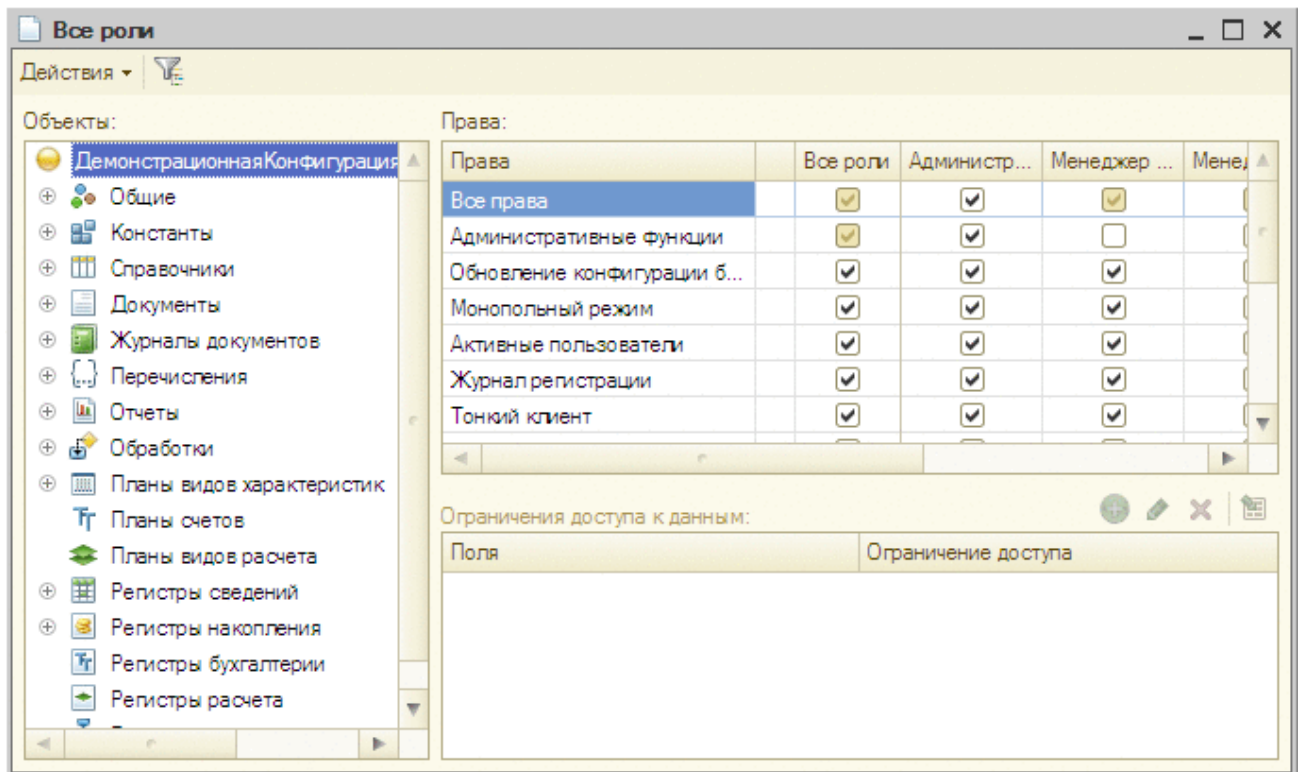


Рисунок 4.4 - Окно редактирования «Все роли»

В окне расположены три табличных поля (Рисунок 4.4). В первом (слева) производится выбор нужного объекта конфигурации. В первой колонке второго табличного поля выводится список прав по выбранному объекту. Другие колонки предназначены для указания использования каждого права для каждой существующей роли.

### Ограничение доступа к данным

Механизм ограничений доступа к данным (также известный как RLS, Row Level Security) позволяет управлять правами доступа не только на уровне объектов метаданных, но и на уровне объектов базы данных «1С: Предприятия». Для ограничения доступа к данным могут быть использованы следующие объекты «1С: Предприятия»:

- роли; Ш параметры сеанса;
- функциональные опции;
- привилегированные общие модули;
- ключевое слово РАЗРЕШЕННЫЕ в языке запросов.

Совместное использование перечисленных объектов позволяет обеспечить максимальную гибкость при необходимости разграничения прав доступа к данным между пользователями, выполняющими различные функции.

Ограничения доступа к данным могут накладываться на следующие операции с данными (права доступа): чтение (право Чтение), добавление (право Добавление), изменение (право Изменение) и удаление (право Удаление). Текущий пользователь будет иметь возможность выполнить требуемую операцию в следующих случаях:

- для операций чтения и удаления объект, находящийся в базе данных, должен соответствовать ограничению доступа к данным;
- для операции добавления ограничению доступа к данным должен соответствовать объект, который планируется записать в базу данных;
- для операции изменения ограничению доступа к данным должен соответствовать объект как до изменения (чтобы объект был прочитан), так и после изменения (чтобы объект был записан).

При наложении ограничений доступа к данным следует помнить, что для операций изменения, добавления и удаления можно задать только одно условие, а для операции чтения можно задать более одного ограничения доступа к данным. Это означает, что для чтения разных полей объекта могут быть заданы разные условия, причем при задании условия можно указать как имя конкретного поля, так и специальное поле Прочие поля.

При задании ограничения на конкретное поле, это поле будет считано в том случае, если ограничение выполняется, а при задании ограничения на Прочие поля, данные объекта будут прочитаны только в том случае, если ограничение выполняется для всех полей объекта, попавших в запрос чтения данных.

Для объектов базы данных следующих видов могут быть наложены различные ограничения на разные виды изменений (добавление, модификацию, удаление): Планы обмена, Справочники,

Документы, Планы видов характеристик, Планы счетов, Планы видов расчета, Бизнес-процессы, Задачи.

Для следующих видов объектов базы данных возможно наложение ограничений на чтение не только всего объекта целиком, но и отдельных его полей: Планы обмена, Справочники, Документы, Журналы документов, Планы видов характеристик, Планы счетов, Планы видов расчета, Регистры сведений, Бизнес- процессы, Задачи.

**Примечание.** При обращении к полям объектов базы данных посредством свойств прикладных объектов из встроенного языка «1С: Предприятие» выполняется чтение всего объекта целиком, а не только значения используемого поля.

Исключением является получение представления, когда будут прочитаны только значения полей, участвующих в формировании представления.

Ограничения доступа содержатся в ролях, они могут быть указаны для большинства объектов метаданных и записываются на специальном языке, являющимся подмножеством языка запросов.

### ***Механизм наложения ограничений.***

Любая операция над данными, хранимыми в базе данных, в «1С: Предприятии» в конечном счете приводит к обращению к базе данных с некоторым запросом на чтение или изменение данных. В процессе исполнения запросов к базе данных внутренние механизмы «1С: Предприятия» выполняют наложение ограничений доступа. При этом:

- Формируется список прав (чтение, добавление, изменение, удаление), список таблиц базы данных и список полей, используемых этим запросом.
- Из всех ролей текущего пользователя выбираются ограничения доступа к данным для всех прав, таблиц и полей, задействованных в запросе. При этом если какая-нибудь роль не содержит ограничений доступа к данным какой-нибудь таблицы или поля, то это значит, что в данной таблице доступны значения требуемых полей из любой записи.

- Получаются текущие значения всех параметров сеанса и функциональных опций, участвующих в выбранных ограничениях.
- Ограничения, полученные из одной роли, объединяются операцией И. Ш Ограничения, полученные из разных ролей, объединяются операцией ИЛИ.
- Построенные условия добавляются к SQL-запросам, с которыми «1С: Предприятие» обращается к СУБД. При обращении к данным со стороны условий ограничения доступа проверка прав не выполняется (ни к объектам метаданных, ни к объектам базы данных). Причем механизм добавления условий зависит от выбранного способа действия ограничений «все» или «разрешенные».

В системе «1С: Предприятие» предусмотрен режим группового редактирования ограничений прав доступа и шаблонов, который вызывается командой «Все ограничения доступа» контекстного меню ветки Роли (Рисунок 4.5).

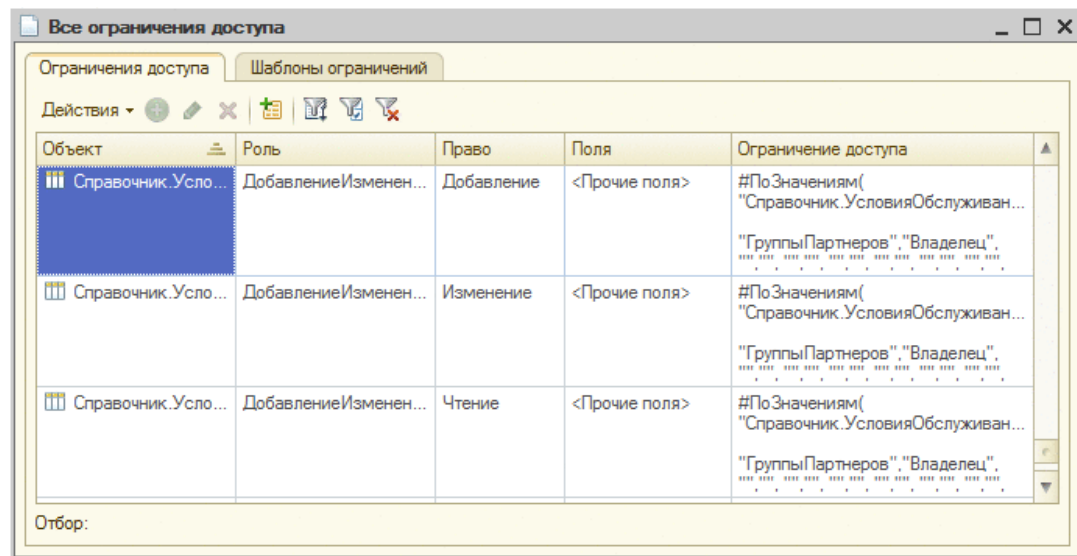


Рисунок 4.5 - Все ограничения прав доступа и шаблоны

На закладке Ограничения доступа можно просматривать все введенные ограничения доступа в общем списке (по всем ролям, объектам, правам, комбинациям полей).

Существует возможность добавлять ограничение доступа сразу для нескольких ролей, объектов, прав и комбинаций ролей.

Сформировать условия ограничения доступа к данным можно с помощью соответствующего конструктора. Для его вызова в табличном поле Ограничения доступа к данным в колонке Ограничение доступа нужно перейти в режим редактирования и нажать кнопку выбора, а в открывшейся форме нажать кнопку «Конструктор запроса».

### ***Общие рекомендации по ограничению прав.***

Чтобы гибко управлять доступом пользователей к данным в соответствии с функциями при установке ограничений доступа к данным, рекомендуется придерживаться следующих принципов:

1. нужно выбрать совокупность информации (может быть зависимой от текущего пользователя), для которой целесообразна предварительная подготовка, распределить ее по параметрам сеанса;
2. установить значения параметров сеанса в обработчике *УстановкаПараметровСеанса ()* модуля сеанса;
3. задать ограничения доступа к тем данным, для которых это оправданно (данные являются секретными или наиболее важными для сохранения целостности системы);
4. при необходимости обеспечить выполнение некоторого ограниченного количества операций над данными со стороны пользователя, которому полный доступ к этим данным давать нецелесообразно, вынести эти действия в привилегированные модули или явно включать и выключать привилегированный режим в соответствующих местах программного кода;
5. доступ к данным при различных проверках, выполняемых системой при записи объектов, выполняется в привилегированном режиме, что позволяет не отключать ограничения в правах на уровне записей для соответствующих полей, если работа конфигурации с этими данными планируется только в управляемом режиме:
  - для справочников при проверке родителя, владельца и уникальности кода;

- для документов, бизнес-процессов и задач при проверке уникальности номера;
- для планов обмена отключена при проверке уникальности кода;
- для планов счетов и планов видов характеристик при проверке родителя и уникальности кода.

При создании запроса ограничения к данным следует помнить о некоторых ограничениях и особенностях:

- 1) Если для объектной таблицы заданы ограничения доступа к данным и в запросе к данным используется объединение с такой таблицей, то в условии соединения (секция запроса ПО) не допускается использование табличной части объекта с заданным ограничением доступа.
- 2) Если в запросе указана таблица, у которой в запросе не используется ни одного поля, то на эту таблицу накладываются все ограничения доступа к данным.
- 3) Если для каких-то полей не задано условий, то запрос будет выполнен для всех записей таблицы.
- 4) Если в запросе используется таблица верхнего уровня, то ограничения, заданные для колонок вложенных таблиц, не накладываются.
- 5) Если в запросе используется вложенная таблица, то накладываются ограничения как для вложенной таблицы, так и для таблицы верхнего уровня.
- 6) Если доступ к полям, необходимым для получения представления ссылочного объекта метаданных, запрещен с помощью ограничений доступа к данным или доступ к объекту запрещен на уровне прав доступа, то получение представления такого объекта не влияет на ход текущей транзакции.

### **Задание**

1. Создать 5 пользователей системы с различными правами доступа.
2. Просмотреть список активных пользователей.
3. Произвести блокировку сеансов пользователей.



4. Изменить права доступа пользователей.
5. Произвести просмотр и редактирование всех ролей.
6. Ограничить доступ к данным некоторым пользователям.

### **Контрольные вопросы**

1. Как просмотреть список зарегистрированных в системе пользователей?
2. Как осуществляется выгрузка информационной базы в файл? В каких случаях это может потребоваться?
3. Как осуществляется загрузка информационной базы из файла?
4. Чем отличается процесс создания резервной копии информационной базы от процесса ее выгрузки в файл?
5. Перечислите функции, относящиеся к стандартным функциям администрирования.
6. Что такое роль? Как создать роль? Как назначить пользователю определенную роль?
7. Что такое набор прав? Как создать набор прав? Как назначить пользователю определенный набор прав?

## ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №5. РАБОТА С ПРИКЛАДНЫМИ ОБЪЕКТАМИ ССЫЛОЧНОГО ВИДА

### Создание внешних обработок

Для того чтобы начать приводить примеры по тем или иным возможностям языка программирования 1С, первоначально мы будем использовать внешние обработки. Что такое обработка? **Обработка** – это, объект конфигурации, предназначенный для реализации различных механизмов обработок информации и сервисных функции. Более подробно вопросы работы с обработками и иными объектами конфигурации мы будем проходить в четвертой главе. В основном мы будем их использовать для изучения языка программирования.

В этой части изучим, как создаются и сохраняются новые обработки, в которых Вы будете делать все Ваши тренировки. В дальнейшем рекомендую Вам на каждую часть главы (и даже, возможно, на каждый пример) создавать отдельную обработку и самостоятельно прописывать все те примеры, которые даются в книге. Если Вы будете прописывать все приведенные мною примеры и выполнять все те рекомендации, которые я дам в главах книги, то усвоите нужный вам материал гораздо лучше, чем просто пролистывая книгу.

Итак, как создать новую обработку? Для этого перейдите в меню: «Файл» - «Новый».

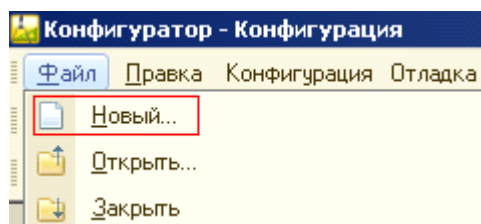


Рисунок 5 – Создание новой обработки

Выбираем в имеющемся списке «Внешняя обработка» и нажимаем кнопку «ОК».

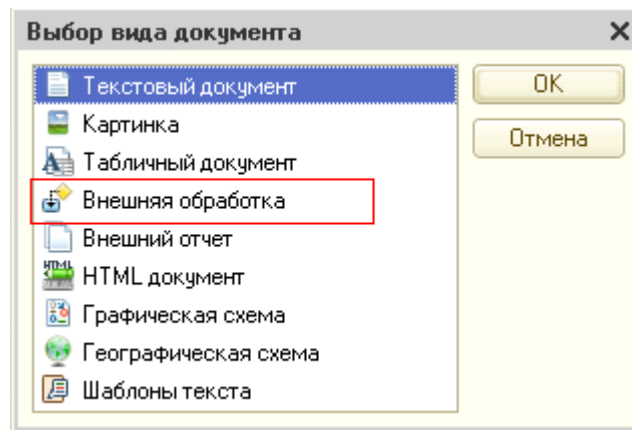


Рисунок 5.1 – Выбор внешней обработки

Открылась вновь созданная внешняя обработка.

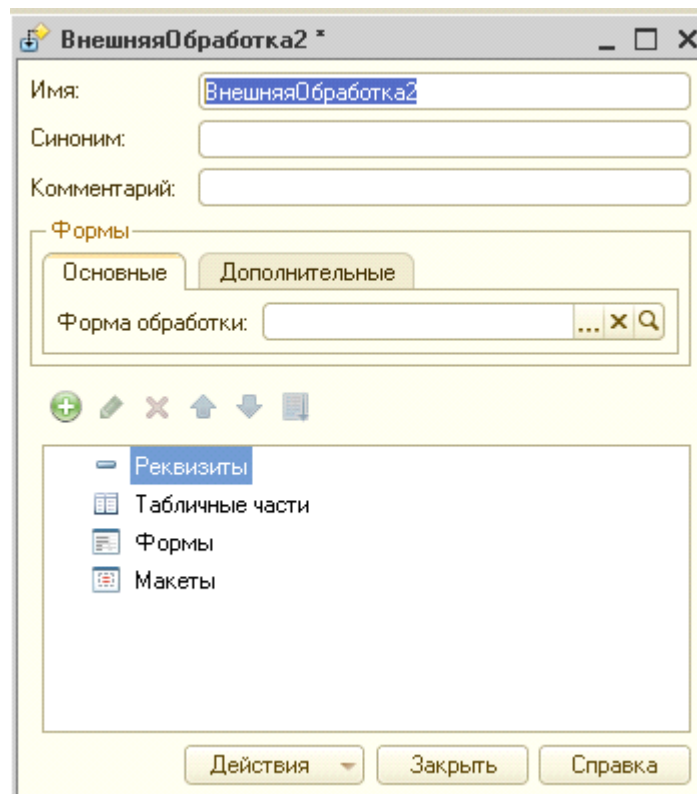


Рисунок 5.2 – Свойства внешней обработки

Назовите как-нибудь Вашу обработку. Рекомендуется в дальнейшем давать имена им по номеру главы, части и примера.

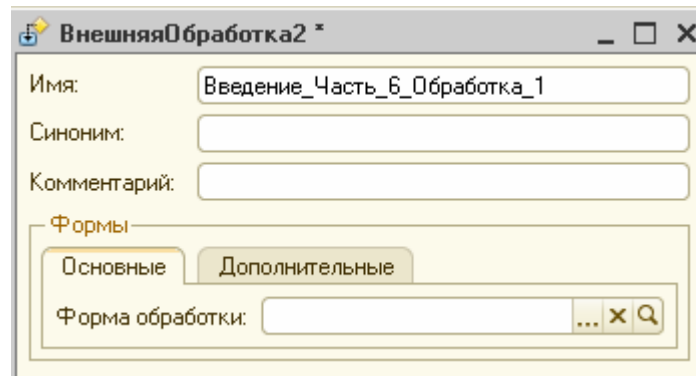


Рисунок 5.3 – Свойства внешней обработки

Сохраните ее в любое место на жестком диске.

Каким образом Вы будете работать с новой обработкой? Делать это мы будем используя форму. На данном этапе обучения мы не будем подробно затрагивать работу с формами, все интересующие нас вопросы на эту тему мы изучим в четвертой главе этой книги.

Создайте форму. Для этого наведите курсор на элемент «Формы» в списке и выделите этот элемент.

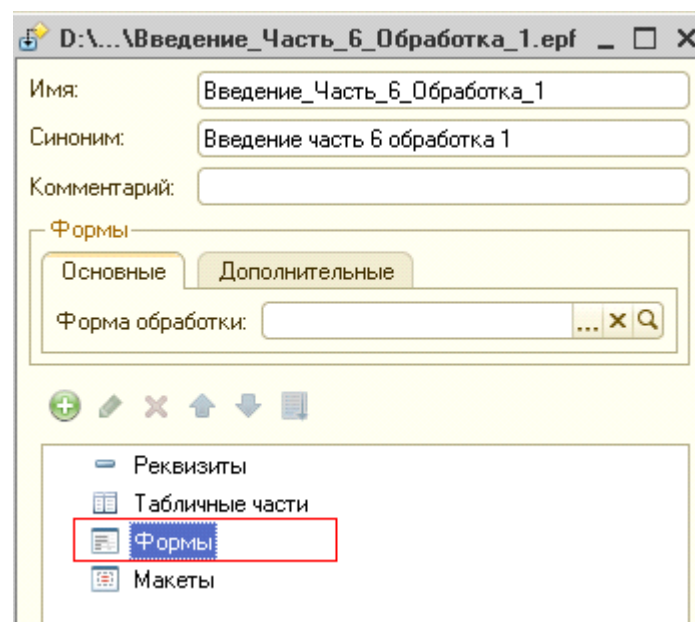


Рисунок 5.4 – Свойства внешней обработки выбор Формы

Нажмите правую кнопку мышки и выберите пункт «Добавить».

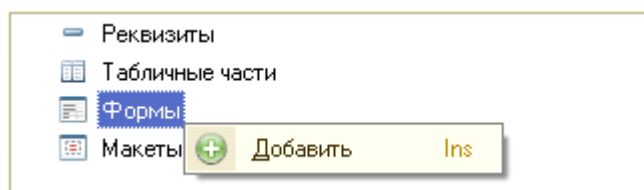


Рисунок 5.5 – Добавление Формы

В открывшемся окне ничего не меняйте и нажмите кнопку «Готово».

Рисунок 5.6 –Конструктор Формы обработки

Открывалась Ваша вновь созданная форма.

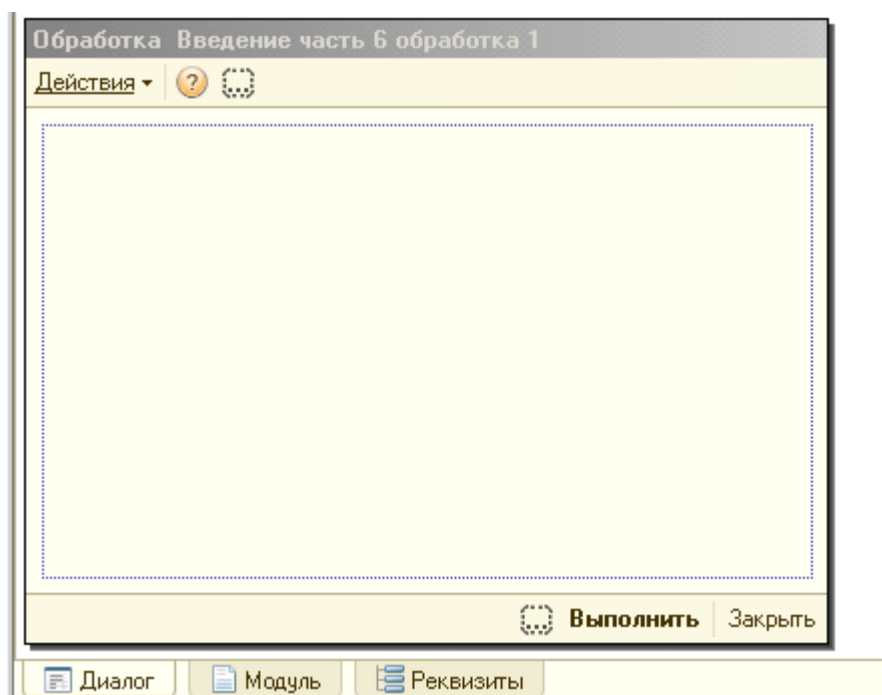


Рисунок 5.7 – Обработка Диалог

Зайдите в модуль формы.

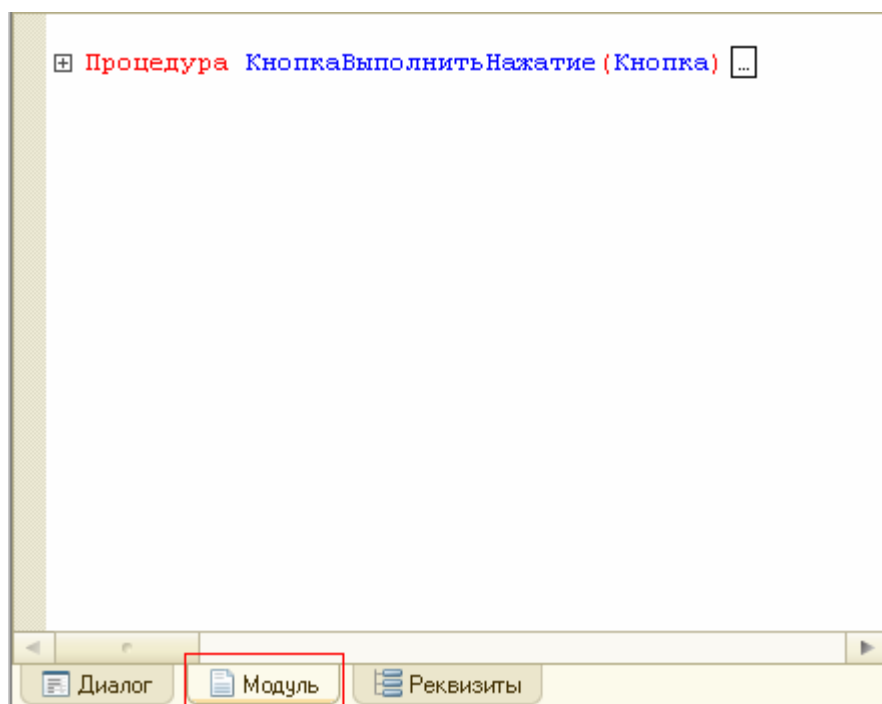


Рисунок 5.8 – Модуль обработки

Раскройте процедуру *«КнопкаВыполнитьНажатие»*. Внутри этой процедуры и будем работать. Напишите в ней следующий код:

```

Процедура КнопкаВыполнитьНажатие (Кнопка)
    Предупреждение ("Привет, Мир!");
КонецПроцедуры

```

Рисунок 5.9 – Код процедуры

Это процедура предупреждение и в ней текст *«Привет, Мир!»*.

А теперь Вам надо посмотреть, как Ваша новая обработка выполнится. Запустите отладку прямо из конфигуратора, нажав кнопку *«Начать отладку»*.

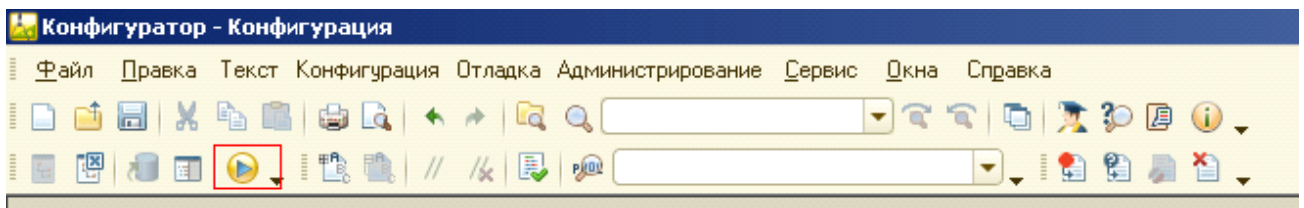


Рисунок 5.10 – Отладка

Вы увидите, что сразу выйдет окно 1С: Предприятия. В этом окне Вы откройте Вашу только что созданную обработку. Идем: *«Файл»* - *«Открыть»*, находим файл, куда Вы ее сохранили. Открываем. Нажимаем кнопку *«Выполнить»* - и выйдет предупреждение.

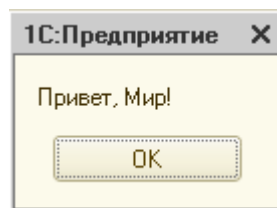


Рисунок 5.11 – Предупреждение

**Задание**

1. Создать обработку
2. Создать форму обработки
3. Создать процедуру
4. С помощью обработки вывести предупреждение пользователю

**Контрольные вопросы**

1. Что такое обработка?
2. Как создать обработку?
3. Что такое процедура?
4. Что такое форма?



## **ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №6. ЗАПРОСЫ К ТАБЛИЦАМ-ИСТОЧНИКАМ ДАННЫХ ПРИКЛАДНЫХ ОБЪЕКТОВ. УСТАНОВКА ОТБОРОВ**

### **Табличная модель доступа к данным. Запросы.**

Наряду с рассмотренной выше объектной моделью доступа к данным, в системе «1С: Предприятие» реализована табличная модель. В основе ее – использование запросов к таблицам базы данных. В результате запроса возвращается набор данных, таблиц, которые можно анализировать и использовать. При помощи запроса программист никоим образом не может изменить данные. В этом отчасти преимущество запросов в системе «1С»: пользователь, программист не может случайно испортить информацию. В основе языка запросов лежит язык SQL с рядом ограничений (и, естественно, на русском языке).

Т.к. запрос производит чтение информации из таблиц базы данных, то, очевидно, что использовать запросы мы можем лишь на стороне сервера: либо в модуле формы при серверных вызовах, либо в модуле объекта (чаще всего). Текст запроса, его обработку и выполнение можно написать вручную, используя специальные конструкции, полный перечень которых приведен в Синтаксис-помощнике. Однако в среде «1С: Предприятие» реализован более удобный инструмент для визуального создания запросов – конструктор запроса. Для вызова конструктора запроса в любой точке программного кода в модуле (формы, объекта) необходимо в контекстном меню выбрать либо «Конструктор запроса», либо «Конструктор запроса с обработкой результата». Конструктор запроса с обработкой результата отличается тем, что при его вызове автоматически формируется код специальных инструкций по выполнению запроса, передаче в него нужных параметров и обход результатов с целью их обработки и анализа; при вызове конструктора запроса без обработки результата производится формирование только текста запроса.

Решим следующую задачу. При поступлении товаров оказывается, что пользователь в табличную часть один и тот же товар может ввести несколько раз, при том даже по разным ценам. В результате при проведении документа в регистре накопления **«Остатки товаров»** будет сформировано несколько похожих записей. Необходимо эти записи сгруппировать, чтобы для одного товара было одно движение. А для регистра сведений **«Цены поставщиков»** установить цену, максимальную для данного товара.

Отрываем процедуру обработки проведения в модуле объекта документа **«Приходная накладная»**, и в начале процедуры запускаем «конструктор запроса с обработкой результата». На предложение создать новый запрос даем положительный ответ. В результате откроется следующее окно (Рисунок 6).

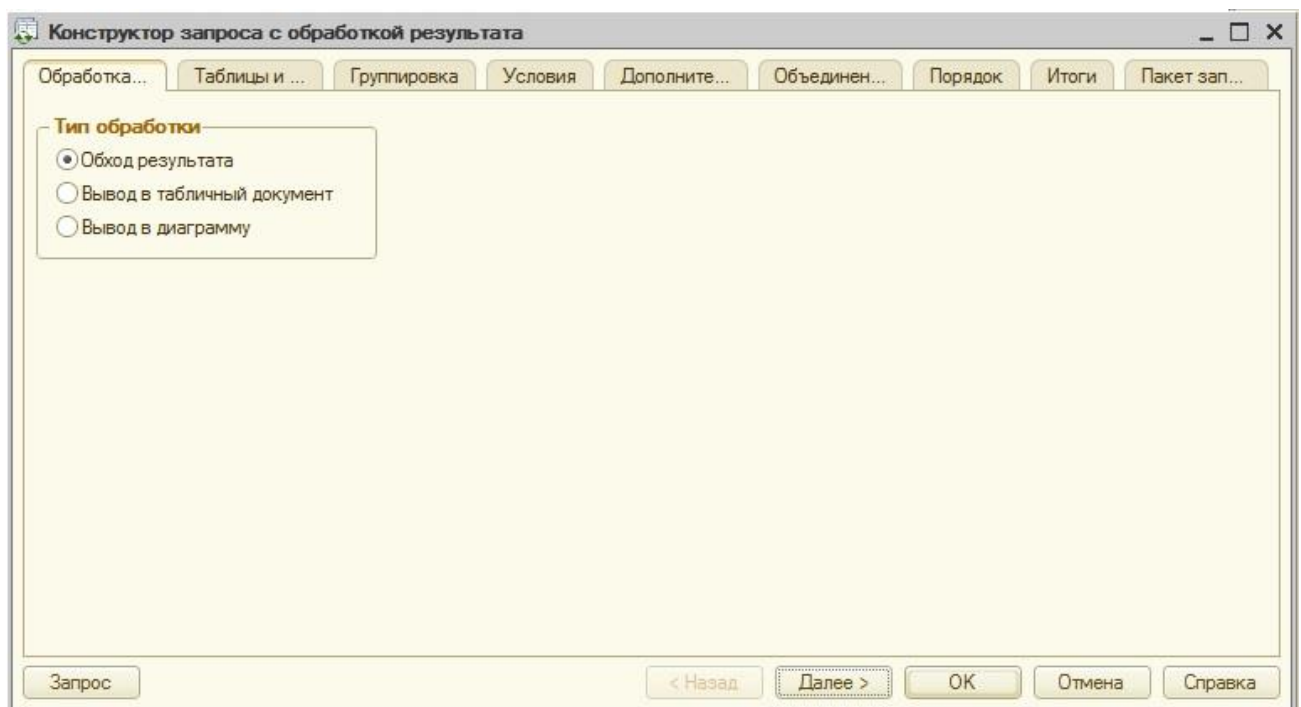


Рисунок 6 - Конструктор запроса с обработкой результата

Выбираем тип обработки «Обход результата» и переходим к следующей вкладке. На вкладке «Таблицы и поля» (Рисунок 6.1) слева представлена структура разрабатываемого приложения в виде таблиц базы данных и виртуальных таблиц. Это – источники данных.

В нашей задаче источником данных будет таблица, представляющая табличную часть документа «**Приходная накладная**». Дело в том, что физически в базе данных реквизиты шапки (справочника, документа) хранятся в одной таблице, а под табличные части элементов создается дополнительная таблица, в которой указывается ссылка на «элемент-владелец» (справочника, документа), к которому данные записи относятся. Из выбранной таблицы необходимо выбрать поля: «**Номенклатура**», «**Цена**», «**Количество**», «**Сумма**» (Рисунок 6.1).

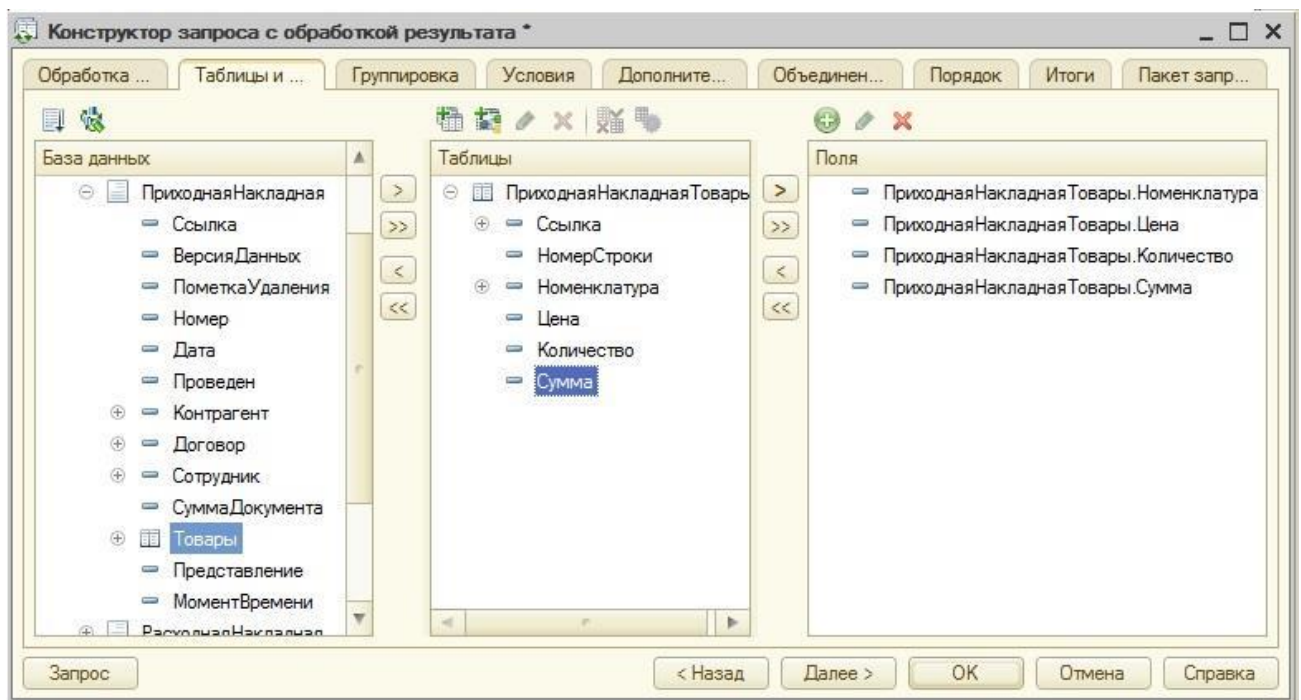


Рисунок 6.1 - Выбор источников данных и полей

На вкладке группировка (Рисунок 6.2) накладываются условия группировки: все поля разносятся по двум категориям: поля группировки и группируемые поля. Не должно возникать ситуации «висящих» полей, не принадлежащих ни той, ни другой группе.

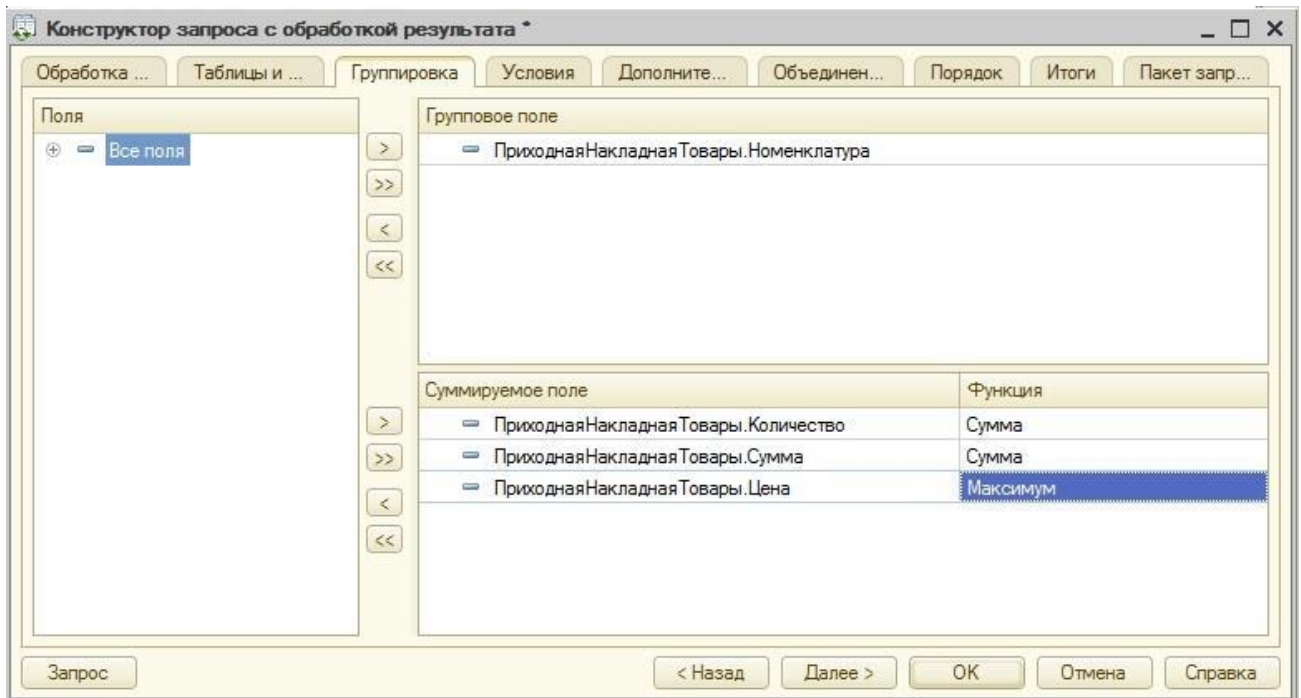


Рисунок 6.2 - Группировка полей в запросе

В нашей задаче полем группировки будет «**Номенклатура**» (одинаковые товары будут фигурировать в движениях один раз), все остальные поля – группируемые. К группируемым полям применяются агрегатные функции («Сумма», «Сумма различных», «Максимум» и т.д.). Для полей «**Количество**» и «**Сумма**» применяется агрегатная функция «Сумма» (для простого суммирования одинаковой номенклатуры, находящейся в различных строчках), а к полю «**Цена**» - функция «Максимум», для выбора наибольшей установленной цены для данной номенклатурной позиции.

На закладке «Условия» (Рисунок 6.3) можно наложить условия на поля запроса, тем самым ограничить число записей, возвращаемых в результате запроса. В нашей задаче мы хотим получить сгруппированные записи не по всем документам, а только по текущему. Для этого нужно наложить условие на реквизит «Ссылка», конкретную ссылку на документ будем передавать через параметр запроса (Рисунок 6.3).

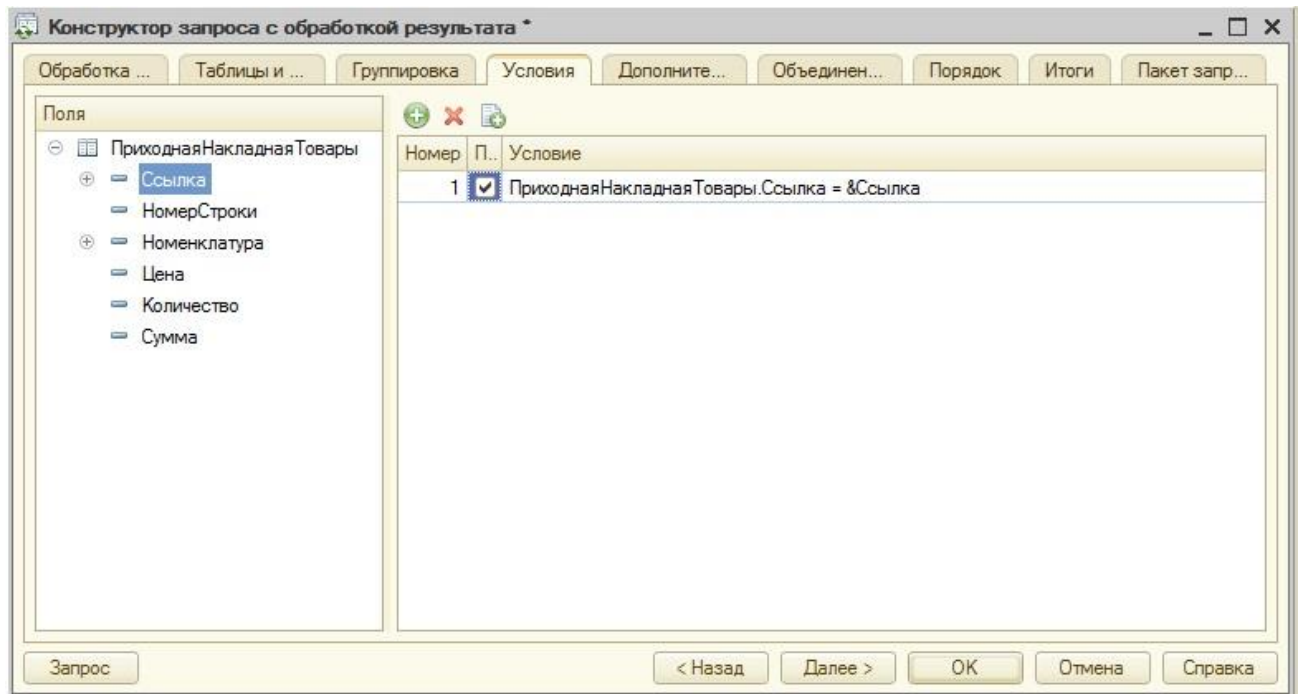


Рисунок 6.3 - Наложение условий в запросе

На вкладке «Объединения/Псевдонимы» можно задать псевдонимы для всех полей, возвращаемых в результате запроса. В нашем случае изменять ничего не нужно.

Для просмотра текста сформированного запроса можно нажать кнопку «Запрос». Для завершения формирования запроса нажимаем «ОК». В результате в модуле объекта сформировался следующий код:

Запрос = Новый Запрос;

Запрос.Текст =

"ВЫБРАТЬ

| ПриходнаяНакладнаяТовары.Номенклатура,

| МАКСИМУМ (ПриходнаяНакладнаяТовары.Цена) КАК Цена,

| СУММА (ПриходнаяНакладнаяТовары.Количество) КАК

Количество,

| СУММА (ПриходнаяНакладнаяТовары.Сумма) КАК Сумма

| ИЗ

| Документ.ПриходнаяНакладная.Товары КАК

ПриходнаяНакладнаяТовары

| ГДЕ

| ПриходнаяНакладнаяТовары.Ссылка = &Ссылка

|

|СГРУППИРОВАТЬ ПО

| ПриходнаяНакладнаяТовары.Номенклатура";

Запрос.УстановитьПараметр("Ссылка", Ссылка);

РезультатЗапроса = Запрос.Выполнить();

ВыборкаДетальныеЗаписи = РезультатЗапроса.Выбрать();

Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

// Вставить обработку выборки ВыборкаДетальныеЗаписи

КонецЦикла;

В данном тексте вначале создается специальный объект типа «Запрос», после чего пишется сам текст запроса. Далее идет раздел задания параметров (УстановитьПараметр), в котором конструктор автоматически определяет параметры, передаваемые в запрос: слева указывается имя параметра в строковом представлении, справа – его значение. Если значение параметра, сформированное автоматически, оказалось неправильным, то его нужно исправить. Далее идет инструкция, выполняющая запрос, после чего следует передача результата запроса: в таблицу, диаграмму, или выборку. Выборка используется для обхода каждой записи в результате запроса. Используя метод **Выбрать()** система позиционируется на начале выборки. Для перехода к каждому следующему элементу выборки и считыванию значений конкретных полей используется метод **Следующий()**.

Соответственно, представленный цикл осуществляет переход к каждому следующему элементу выборки до конца. Внутри данного цикла мы можем обращаться к полям запроса как к полям выборки, например, **ВыборкаДетальныеЗаписи.Номенклатура**.

В нашей задаче параметр, передаваемый в запрос верный: мы передаем ссылку на текущий обрабатываемый документ. Цикл обхода элементов выборки будет выглядеть следующим образом (жирным выделены изменения в формировании движений, по сравнению с предыдущим вариантом):

Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

Движение = Движения.ОстаткиТоваров.Добавить ();

Движение.ВидДвижения = ВидДвиженияНакопления.Приход;

Движение.Период = Дата;

Движение.Номенклатура =  
**ВыборкаДетальныеЗаписи.Номенклатура;**  
 Движение.Количество = **ВыборкаДетальныеЗаписи.Количество;**  
 Движение.Стоимость = **ВыборкаДетальныеЗаписи.Сумма;**  
 Движение = Движения.ЦеныПоставщиков.Добавить ();  
 Движение.Период = Дата;  
 Движение.Контрагент = Контрагент;  
 Движение.Номенклатура =  
**ВыборкаДетальныеЗаписи.Номенклатура;**  
 Движение.Цена = **ВыборкаДетальныеЗаписи.Цена;**  
 КонецЦикла;

После цикла, не забываем оставить возможность записи движений в регистры:

- Движения.ОстаткиТоваров.Записывать = Истина;
- Движения.ЦеныПоставщиков.Записывать = Истина;

Весь оставшийся предыдущий код процедуры обработки проведения, удаляем.

Запускаем систему и проверяем данный механизм на практике. Обратите внимание, что запрос используется лишь для получения, группировки и обработки данных. При этом запрос работает намного быстрее, чем обращение через точку (объектная модель). Но для записи данных все равно используется объектная модель.

### Задание

1. Выполните группировку строк в документе «**Расходная накладная**» перед записью в регистры накопления.

2. Измените процедуру проведения для документа «**Расходная накладная**». Учтите тот факт, что по табличной части «**Товары**» формируются движения в регистрах «**Остатки товаров**» и «**Продажи**», а по табличной части «**Услуги**» – только по регистру «**Продажи**».

**Контрольные вопросы**

1. Что такое запрос?
2. Как создать запрос?
3. Как выполнить группировку?
4. Как проверить результат выполнения запроса?



## ЛАБОРАТОРНАЯ РАБОТА №1. УПОРЯДОЧИВАНИЕ. ПОЛУЧЕНИЕ СВОДНОЙ ИНФОРМАЦИИ.

### Методы документов

Существует несколько методов для работы с документами в режиме разработки «Конфигуратор»:

#### 1. НайтиДокумент ().

Метод НайтиДокумент (Документ) предназначен для нахождения документа по значению (или позиционировать документ). Возвращает:

- 1 — если действие выполнено (документ найден);
- 0 — если действие не выполнено.

Параметр Документ — выражение со значением типа "Документ".

#### 2. НайтиДокументПоНомеру ().

Метод НайтиДокументПоНомеру (Номер, Дата, ИдентификаторВида) предназначен для поиска документа по номеру. Он может возвращать значения:

- 1 — если действие выполнено (документ найден);
- 0 — если действие не выполнено.

Параметры:

- Номер — строка с номером искомого документа;
- Дата — дата из диапазона, в котором нужно искать документ.
- ИдентификаторВида — необязательный параметр (строковое выражение, содержащее ИдентификаторВида документа или идентификатор Нумератора).

Параметр ИдентификаторВида задается для объекта "Документ" общего вида.

#### 3. ВыбратьДокументы ().

Метод ВыбратьДокументы (Дата1, Дата2) открывает выборку документов в интервале дат.

Возвращает значения:

- 1 — если действие выполнено и в выборке есть хотя бы один документ;

- 0 — если действие не выполнено или в выборке нет ни одного документа.

Параметры:

- Дата1 — дата, документ или позиция начала выборки документов (если данный параметр опущен, то выборка начинается с самого первого существующего в системе документа);
- Дата2 — дата, документ или позиция конца выборки документов (если данный параметр опущен, то выборка заканчивается самым последним существующим в системе документом).

Снимем пометки на удаление с документов за период с 1 января 2023 г. по 31 марта 2023 г.:

Док = СоздатьОбъект («Документ»);

// Зададим выборку документов

Док.ВыбратьДокументы ("01.01.2004", "31.03.2004");

Пока Док.ПолучитьДокумент () = 1 Цикл

Если Док.ПометкаУдаления () = 1 Тогда

Док.СнятьПометкуУдаления ();

КонецЕсли;

КонецЦикла;

4. ВыбратьПодчиненныеДокументы ().

Подчиненными документами являются документы, которые были введены на основании другого документа, который в свою очередь называется документом-основанием.

Метод ВыбратьПодчиненныеДокументы (Дата1, Дата2, Документ) открывает выборку документов, подчиненных заданному документу-основанию, в интервале дат.

Возвращает значения:

- 1 — если действие выполнено и в выборке есть хотя бы один документ;
- 0 — если действие не выполнено или в выборке нет ни одного документа.

### Параметры:

- Дата1 — дата, документ или позиция начала выборки документов (если данный параметр опущен, то выборка начинается с самого первого существующего в системе документа);
- Дата2 — дата, документ или позиция конца выборки документов (если данный параметр опущен, то выборка заканчивается самым последним существующим в системе документом);
- Документ — документ-основание.

Рассмотрим использование метода на примере. Для этого создадим в журнале "Ремонт" графу Оплата с типом "Текст", без привязки к реквизиту документов. Для отображения суммы оплаты документа определим для этой графы функцию Оплата ():

Функция Оплата ()

Сумма=0;

Если (ТекущийДокумент.Вид() = "Ремонт") Тогда

ТекущийДокумент.Операция.ВыбратьПроводки ();

Пока ТекущийДокумент.Операция.ПолучитьПроводку () = 1 Цикл

Если ТекущийДокумент.Операция.Дебет.Счет = СчетПоКоду ("62.2")

Тогда

Сумма = Сумма + ТекущийДокумент.Операция.Сумма;

КонецЕсли;

КонецЦикла;

Если Сумма=0 Тогда

Док=СоздатьОбъект("Документ");

Док.ВыбратьПодчиненныеДокументы (,ТекущийДокумент);

Пока Док.ПолучитьДокумент() = 1 Цикл

Если Док.Вид () = "Выписка" Тогда

Док.ВыбратьСтроки();

Пока Док.ПолучитьСтроку() = 1 Цикл

Если Док.ДокументПоставки = ТекущийДокумент Тогда

Сумма = Док.Приход;

КонецЕсли;  
 КонецЦикла;  
 ИначеЕсли Док.Вид() = "ПриходныйОрдер" Тогда  
 Сумма = Док.Сумма;  
 КонецЕсли;  
 КонецЦикла;

#### 5. ВыбратьПоЗначению ().

Метод ВыбратьПоЗначению (Дата1, Дата2, ИмяОтбора, Значение) открывает выборку документов в интервале дат с заданным значением реквизита отбора.

Возвращает:

- 1 — если действие выполнено и в выборке есть хотя бы один документ;
- 0 — если действие не выполнено или в выборке нет ни одного документа.

Параметры:

- Дата1 — дата, документ или позиция начала выборки документов (если данный параметр опущен, то выборка начинается с самого первого существующего в системе документа);
- Дата2 — дата, документ или позиция конца выборки документов (если данный параметр опущен, то выборка заканчивается самым последним существующим в системе документом);
- ИмяОтбора — строка с названием общего реквизита документов либо названием графы отбора журналов;
- Значение — значение отбора, по которому строится выборка документов.

Изменим процедуру ПоступлениеПоДокументам (), приведенную для иллюстрации метода Выбрать(). Для этого используем метод ВыбратьПоЗначению () для сокращения выборки до перечня документов только заданного контрагента.

Процедура ПоступлениеПоДокументам ()

```

Меню = СоздатьОбъект("СписокЗначений");
Меню.ДобавитьЗначение("Счет", "Счет");
Меню.ДобавитьЗначение("СчетФактура", "Счет-фактура");
Меню.ДобавитьЗначение("РасходнаяНакладная", "Отгрузка товаров,
продукции");
Меню.ДобавитьЗначение("РеализацияОтгруженнойПродукции",
"Реализация отгруженной продукции");
Меню.ДобавитьЗначение("ОказаниеУслуг", "Оказание услуг");
Меню.ДобавитьЗначение("ВыполнениеЭтапаРабот", "Выполнение
этапа работ");
Меню.ДобавитьЗначение("ОтпускМатериаловНаСторону", "Отгрузка
материалов на сторону");
Меню.ДобавитьЗначение("ПередачаОС", "Передача ОС");
Меню.ДобавитьЗначение("ПередачаНМА", "Передача НМА");
ВидДок = ""; Если Меню.ВыбратьЗначение (ВидДок,,,1) = 1 Тогда
Док = СоздатьОбъект("Документ."+ВидДок);
// Контрагент задан в поле Субконто1, ограничим выборку по
Субконто1 Док.ВыбратьПоЗначению(ДатаДок, «Контрагент»,
Субконто1);
// Предложим диалоговое окно для выбора документа из перечня //
документов заданного контрагента
Если Док.Выбрать("Выберите документ, в оплату которого поступили
денежные средства") = 1 Тогда
СформироватьСтрокиПоДокументу (Док);
КонецЕсли;
КонецЕсли;
КонецПроцедуры // ПоступлениеПоДокументам
6. ВыбратьПоНомеру ().
Метод ВыбратьПоНомеру (Номер, Дата, ИдентВида)
открывает выборку документов в интервале дат по номеру.
Возвращает значения:


- 1 — если действие выполнено и в выборке есть хотя бы один
документ;

```

- 0 — если действие не выполнено или в выборке нет ни одного документа.

Параметры:

- Номер — строка, содержащая номер искомых документов;
- Дата — любая дата из диапазона, в котором нужно искать документ с данным номером. Поиск зависит от выбранного в Конфигураторе способа уникальности номеров (по месяцу, году и др.);
- ИдентВида — необязательный параметр (строковое выражение, содержащее идентификатор вида документа или идентификатор "Нумератора").

#### 7. ВыбратьПоПоследовательности ().

Метод ВыбратьПоПоследовательности (Дата1, Дата2, КодПоследовательности) открывает выборку документов в интервале дат по заданной последовательности.

Возвращает значения:

- 1 — если действие выполнено и в выборке есть хотя бы один документ;
- 0 — если действие не выполнено или в выборке нет ни одного документа.

Параметры:

- Дата1 — необязательный параметр, представляющий дату, документ или позицию начала выборки документов (если данный параметр опущен, то выборка начинается с самого первого существующего в системе документа);
- Дата2 — необязательный параметр, представляющий дату, документ или позицию конца выборки документов (если данный параметр опущен, то выборка заканчивается самым последним существующим в системе документом);
- КодПоследовательности — строка с названием используемой последовательности.

#### 8. УстановитьФильтр ().

Метод УстановитьФильтр (Проведенные, Непроведенные, НеИмеющиесяПризнаковУчета, Оперативные, Расчетные, Бухгалтерские) назначает фильтр выборки документов.

Параметры:

- Проведенные — число, принимающее значения:
  - 0 — не включать в выборку проведенные документы;
  - 1 — включать;
- Непроведенные — число, принимающее значения:
  - 0 — не включать в выборку непроведенные документы;
  - 1 — включать;
- НеИмеющиесяПризнаковУчета — число, принимающее значения:
  - 0 — не включать в выборку документы, не имеющие признаков учета;
  - 1 — включать;
- Оперативные — число, принимающее значения:
  - 0 — не включать в выборку оперативные документы;
  - 1 — данный флаг не влияет на выборку;
  - 2 — если оперативный документ, то включается в выборку;
- Расчетные — число, принимающее значения:
  - 0 — не включать в выборку расчетные документы;
  - 1 — данный флаг не влияет на выборку;
  - 2 — если расчетный документ, то включается в выборку;
- Бухгалтерские — число, принимающее значения:
  - 0 — не включать в выборку бухгалтерские документы;
  - 1 — данный флаг не влияет на выборку;
  - 2 — если бухгалтерский документ, то включается в выборку.

#### 9. ПолучитьДокумент ().

Метод ПолучитьДокумент () получает из выборки следующий документ. Возвращает значения:

- 1 — если документ выбран;
- 0 — в противном случае.

Метод не имеет параметров.

При иллюстрации метода ВыбратьДокументы () мы использовали данный метод для перебора всех документов, заданных в выборке:

```
Док = СоздатьОбъект("Документ");
```

```
// Зададим выборку документов Док.ВыбратьДокументы  
("01.01.2004","31.03.2004");
```

```
// Выбираем следующий документ, пока метод возвращает 1
```

```
Пока Док.ПолучитьДокумент() = 1 Цикл
```

```
Если Док.ПометкаУдаления() = 1 Тогда
```

```
Док.СнятьПометкуУдаления();
```

```
КонецЕсли;
```

```
КонецЦикла;
```

### **Задание**

1. Создать документ с параметрами
2. Заполнить документ данными
3. Выполнить с 1 по 9 метод для работы с документами
4. Результаты представить в виде отчета

### **Контрольные вопросы**

1. Какие существуют методы для работы с документами
2. Что представляет метод НайтиДокумент ().
3. Что представляет метод НайтиДокументПоНомеру ().
4. Что представляет метод ВыбратьДокументы ().
5. Что представляет метод ВыбратьПодчиненныеДокументы ().
6. Что представляет метод ВыбратьПоЗначению ().
7. Что представляет метод ВыбратьПоНомеру ().
8. Что представляет метод ВыбратьПоПоследовательности ().
9. Что представляет метод УстановитьФильтр ().
10. Что представляет метод ПолучитьДокумент ().



## ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №7. РАБОТА С ОБЩИМИ ФОРМАМИ

### Форма констант

Форма констант, как видно из названия позволяет работать с несколькими константами, задавать и менять их значение. В отличие от *формы константы*, создаваемой сразу при создании константы, в которой можно работать с одной константой

Форма констант – одна из общих форм. Как и другие общие объекты конфигурации, общие формы находятся в узле "общие" дерева конфигурации. Как и другие объекты конфигурации, общую форму можно создать, как через контекстное меню узла конфигурации, так и через команду "действия" окна дерева конфигурации. Кроме того, форму констант можно создать через контекстное меню соответствующей константы.

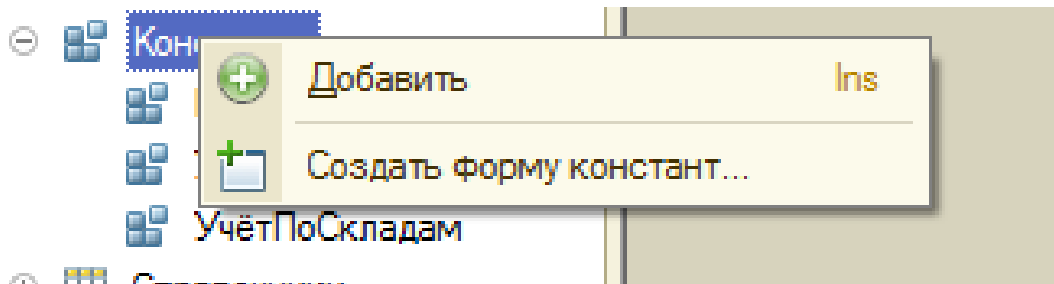


Рисунок 7 – Создание формы констант через контекстное меню узла константы

Результатом выполнения команды будет запуск окна редактирования объекта конфигурации (в данном случае формы). Созданную форму констант можно найти в узле "Общие дерева конфигурации".

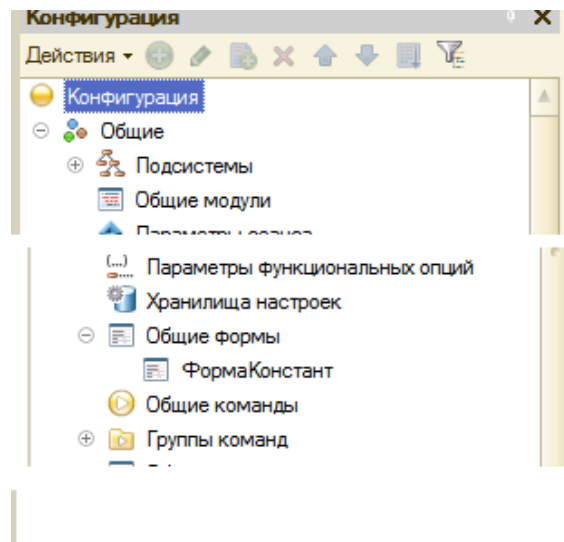


Рисунок 7.1 - Форма констант в дереве конфигурации

При создании формы констант из узла "Общие формы" предварительно открывается окно конструктора общих форм, в котором необходимо выбрать тип создаваемой общей формы.

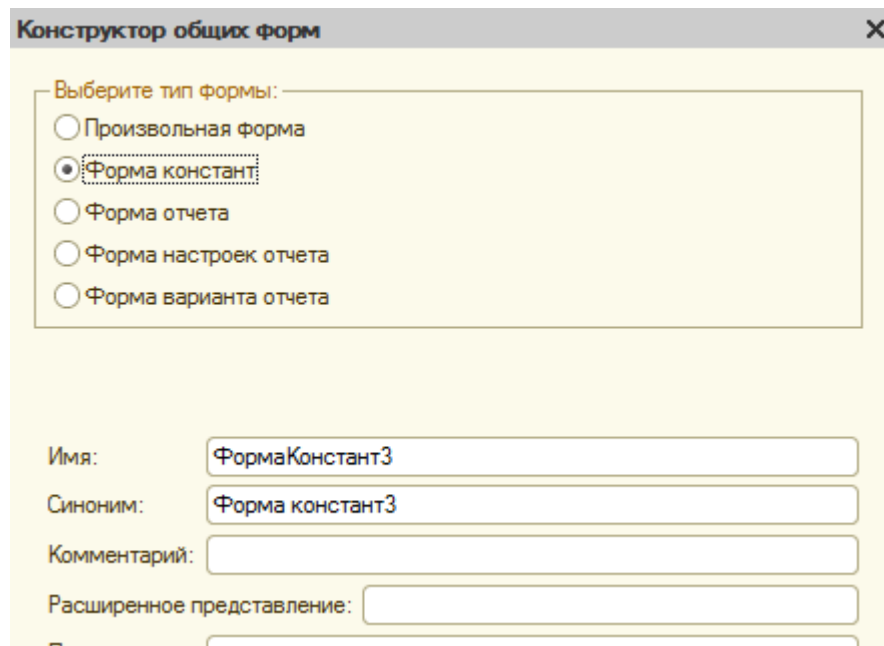


Рисунок 7.2 - Окно конструктора общих форм

Как видно из окна конструктора форм возможны следующие типы общих форм:

- произвольная форма,
- форма констант,

- форма отчёта,
- форма настроек отчёта,
- форма варианта отчёта.

Отличие типов форм заключается в предварительно определённых для них реквизитах.

Форму констант можно рассматривать как частный случай формы среды 1С8.X

Общие принципы работы с конструктором форм одинаковы и не зависят от того, для какого объекта вы создаете форму. Это несколько этапов:

- Выбор типа формы (один из ВАЖНЕЙШИХ моментов). Выбор типа напрямую влияет на функциональность формы, предоставляемой по умолчанию.
- Определение имени формы
- Определения значения флага «Использовать стандартные команды» (установка этого флага приводит к появлению стандартных команд в интерфейсе в тех подсистемах, к которым относится данный объект, в нашем случае это форма)
- После нажатия на кнопку «Далее» можно определить состав отображаемых (в нашем случае) констант.

Рисунок 7.3 - Конструктор общих форм

Для того что бы форму можно было увидеть в интерфейсе пользователя, форму, как и другие объекты нужно отнести к одной из подсистем.

После нажатия на кнопку готово, откроется конструктора формы.

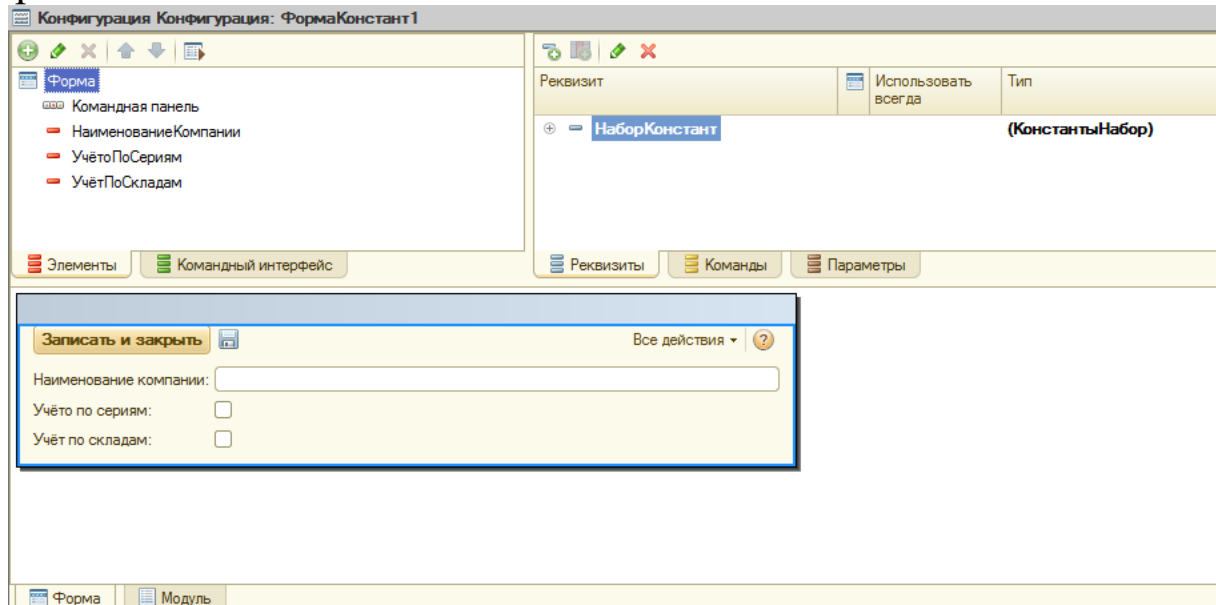


Рисунок 7.4 - Окно конструктора формы.

Основные элементы окна конструктора формы.

В окне конструктора формы две вкладки - вкладка "форма" и вкладка "Модуль". На вкладке "форма" происходит работа с формой в режиме конструктора. На вкладке модуль отображается программный модуль формы.

В правой верхней части окна определяется список реквизитов формы (закладка "Форма"), то есть того, что входит в понятие «Данные формы». Реквизиты также можно назвать источниками функциональности формы. На форму можно добавить реквизиты соответствующие существующим объектам конфигурации. Для обеспечения доступа к данным базы и их сохранения у реквизита в панели его свойств должны быть установлены флаги "Основной реквизит", "Сохраняемые данные".

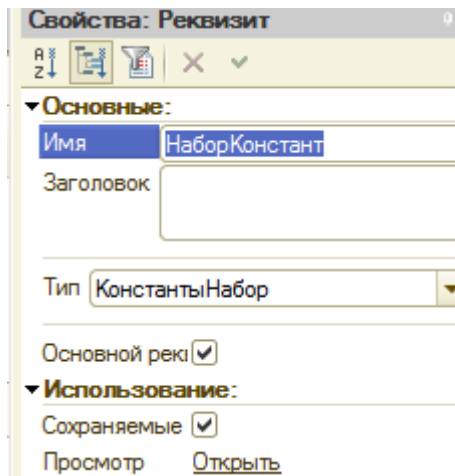


Рисунок 7.5 - Установка свойств реквизита формы

В той же части, в которой определены реквизиты формы, на другой закладке определяются особые элементы формы "Команды формы". Под командой формы понимается особый объект, с которым может быть связан некоторая функция или процедура программного модуля. Связь команды с процедурой или функцией задаётся через свойство команды "Действие".

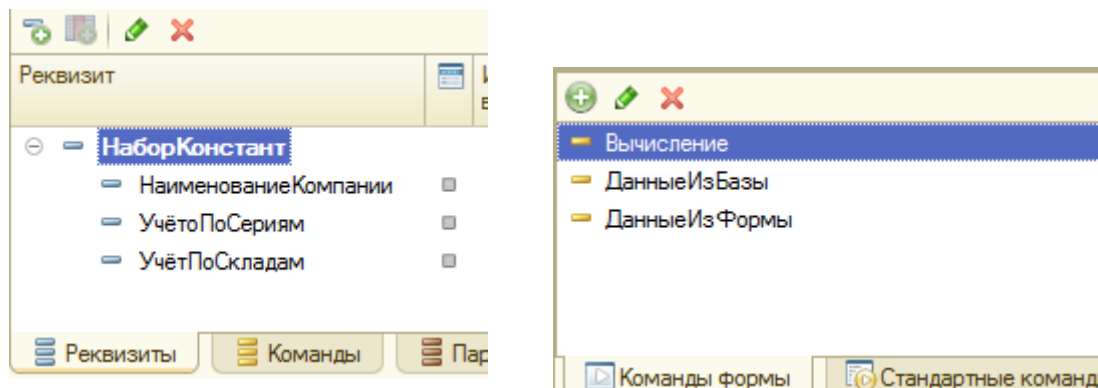


Рисунок 7.6 - Реквизиты и команды формы

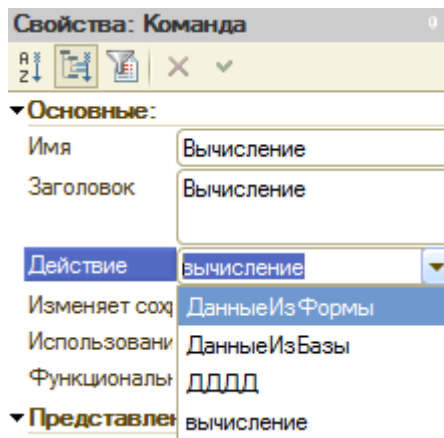


Рисунок 7.7 - Связь команды с процедурой обработчиком

В левой верхней части формы находится дерево элементов управления.

С помощью дерева элементов управления можно изменить какие-либо свойства элементов, определить обработчики событий. Корень данного дерева определяет саму форму (для изменения свойств формы в целом, нужно работать со свойствами именно корневого объекта «Форма»). По умолчанию в состав элементов управления входит командная панель. Настройка команд командной панели происходит отдельно.

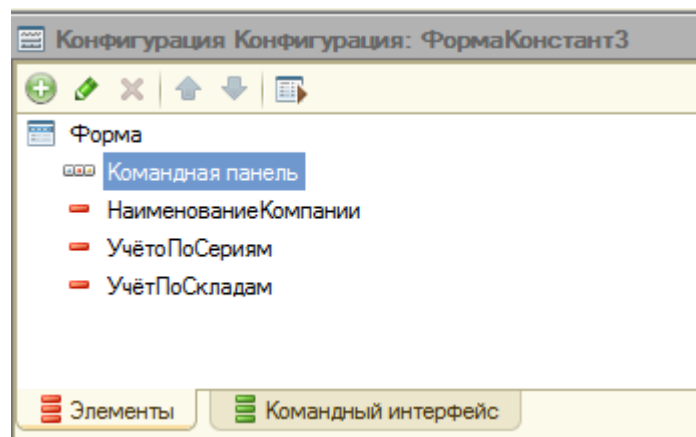


Рисунок 7.8 - Дерево управляющих элементов формы

В нижней части находится область предпросмотра (какой внешний вид имеет форма в результате всех ваших действий).

*Когда в рабочем режиме выполняется открытие формы – система выполняет следующие действия.*

- создает экземпляр формы,
- производит чтение объектов, связанных с реквизитами формы (для формы констант, это будут константы). Прочитанные значения «копируются» в соответствующие реквизит формы.
- Форма отправляется клиентскому приложению.
- Элементы управления отображают значения реквизитов формы, связанных с ними. Связь элемента управления со значением свойства реквизита осуществляется через свойство «Данные» элемента.

При настройке логики работы формы принят следующий подход:

- элементы управления выступают источниками событий;
- элементы управления активизируют связанные с ними команды;
- в качестве обработчиков событий вызываются программные модули, связанные с командами;
- программные модули в качестве данных используют реквизиты формы, при изменении реквизитов формы автоматически могут измениться свойства связанных с ними элементов управления.

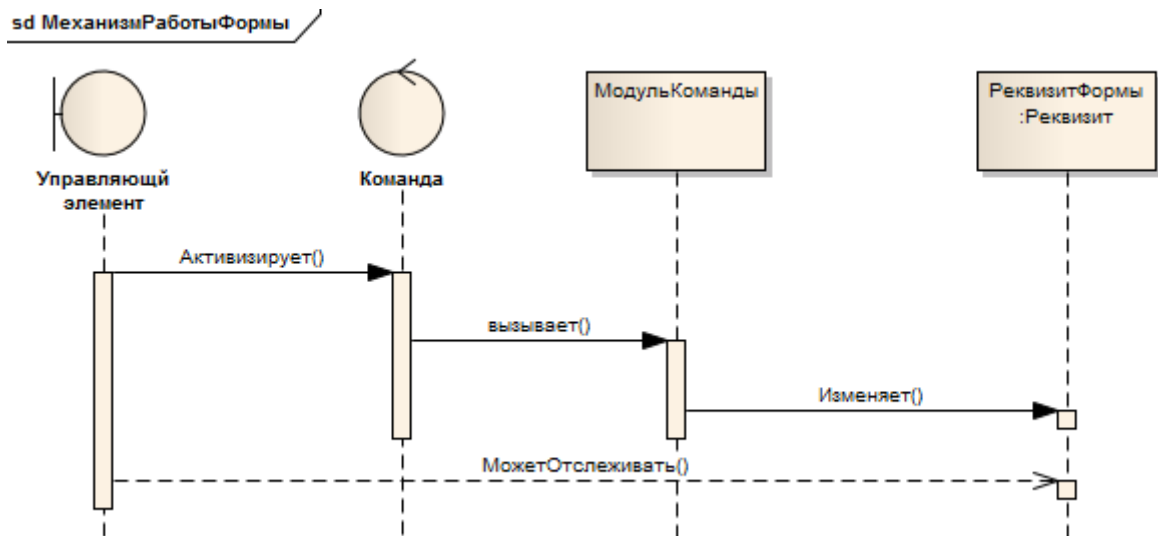


Рисунок 7.9 - Механизм работы формы

Для того что бы рассмотреть разработку программных модулей 1с необходимо рассмотреть базовые принципы программирования на языке 1с и особенности построения программных модулей.

### **Разработка программных модулей 1с.**

Формат описания процедур и функций:

```
&ДирективаКомпиляции
Процедура ДДДД(Параметры)
// Вставить содержимое обработчика.
КонецПроцедуры

&ДирективаКомпиляции
Функция ИмяФункции(Параметры)

Возврат (ПозвращаемыйПараметр);
КонецФункции
```

Следует отметить, что каждая процедура, функция или объявление переменной модуля формы должны предваряться одной из следующих директив компиляции:

&НаКлиенте - означает, что процедура/функция выполняется на стороне клиента, а переменная существует все время жизни клиентской части управляемой формы. Из клиентского метода допустимыми являются вызовы клиентских, серверных и серверных внеконтекстных методов.

&НаСервере - означает, что процедура/функция выполняется на стороне сервера, а переменная существует только во время вызова выполнения серверного или серверного внеконтекстного вызова. Для серверных методов допустимыми являются вызовы серверных и серверных внеконтекстных методов.

&НаСервереБезКонтекста означает, что процедура/функция исполняется на сервере вне контекста формы. Переменные не могут быть внеконтекстными. В таких методах недоступен контекст формы (включая данные формы). Допустимыми являются вызовы только



других внеконтекстных методов. При вызове этих методов не выполняется передача данных формы на сервер и обратно. Применение внеконтекстных методов позволяет существенно уменьшить объем передаваемых данных при вызове серверной процедуры из среды клиентского приложения.

`&НаКлиентеНаСервереБезКонтекста` - процедура/функция может исполняться в управляемом клиенте или на сервере, при этом контекст формы не доступен.

Отсутствие директивы компиляции перед процедурой означает использование директивы `«&НаСервере»`.

В качестве процедуры, связанной с командой (используемой в качестве "Действия" команды, может использоваться только процедура с директивой компиляции `"НаКлиенте"`

Рассмотрим пример создание логики работы формы. Пусть требуется обеспечить чтение данных из реквизитов формы.

Последовательность создания логики работы формы:

1. Создаются необходимые реквизиты, обеспечивающие доступ к требуемым данным.
2. Создаются команды на окне РеквизитыКоманды закладка – команды.
3. Создаются необходимые управляющие элементы на форме, с управляющими элементами связываются команды.
4. Создаётся программный модуль (процедура или функция)– связанный с данной командой. Программный модуль создаётся на вкладке формы.
5. Модуль связывается с командой. свойство "действие" в панели свойств команды. Если в качестве свойства "действие" задать имя не существующего программного модуля, то в модуле формы будет автоматически воздан шаблон процедуры с директивой `"НаКлиенте"`.
6. Создаются дополнительные процедуры и функции, реализующие требуемую логику работы.

### Пример создания логики работы формы.

1. Создайте новую команду – [ДанныеИзФормы](#).
2. На палитре свойств команды для свойства "Действие" зададим новое действие (введём имя) или выберем из существующих. В качестве имя действия подразумевается имя программного модуля.

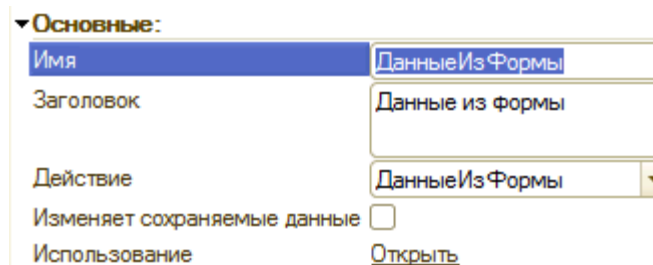


Рисунок 7.10 - Задание имени программного модуля – обработчика команды

Так как программного модуля с именем "ДанныеИзФормы" не существует, то в модуле формы автоматически создана следующая процедура -

[&НаКлиенте](#)

[Процедура ДДДД\(Команда\)](#)

// Вставить содержимое обработчика.

[КонецПроцедуры](#)

Обратите внимание, у процедуры – обработчика – стоит опция [&НаКлиенте](#). Как уже отмечалось, только такие процедуры можно назначать в качестве обработчика команд, кроме того у такой процедуры, должен быть определён формальный параметр – [Команда](#).

Пусть обработчик команды будет определён следующим образом:

[&НаКлиенте](#)

[Процедура ДанныеИзФормы \(Команда\)](#)

```

ЗначениеКонстанты = НаборКонстант.НаименованиеКомпании;
ПоказатьОповещениеПользователя (ЗначениеКонстанты);
КонецПроцедуры

```

В данной процедуре используется объект "НаборКонстант" это автоматически созданный реквизит на форме констант, обеспечивающий доступ к константам из программного модуля, выполняемого на клиенте, определённым в прикладном решении. Можно такой реквизит создать вручную установив для него тип "НаборКонстант".

Добавьте на форму управляющий элемент Кнопка и задайте в качестве команды, связанной с ним команду ДанныеИзФормы.

Создайте новую команду ДанныеИзБазы и определите для неё обработчик с именем ДанныеИзБазы. Текст обработчика команды задайте следующий:

```

&НаКлиенте
Процедура ДанныеИзБазы (Команда)
ЗначениеКонстанты=ДанныеИзБазыСервер();
ПоказатьОповещениеПользователя(ЗначениеКонстанты);
КонецПроцедуры

```

В процедуре используется дополнительная процедура "ДанныеИзБазыСервер" она осуществляет чтение данных из базы данных системы.

```

&НаСервереБезКонтекста
Функция ДанныеИзБазыСервер()
ЗначениеКонстанты=Константы.НаименованиеКомпании.Получ
ить();
Возврат(ЗначениеКонстанты);
КонецФункции

```

Константы – специальный объект, обеспечивающий доступ к значениям констант, хранимых в базе данных.

Таким образом, на форме реализовано чтение данных из константы "Наименование компании" с помощью клиентского объекта "НаборКонстан" и с помощью объекта, обеспечивающего чтение данных из базы данных – "Константы". Разница в двух этих подходах состоит в организации логики работы. Предпочтительным является использование данных базы данных, что позволяет строить полноценного тонкого клиента.

***Используемые в программных модулях системные функции, константы, типы данных.***

#### **Функция**

**ПоказатьОповещениеПользователя**((<Текст>, <НавигационнаяСсылка>, <Пояснение>, <Картинка>)

*Показывает окно, по умолчанию располагающееся внизу экрана, которое постепенно затухает и исчезает с экрана. В окне может располагаться ссылка, выбор которой вызывает определенные для ссылки действия (например, открытие формы).*

*Функция сообщить выводит данные в строку сообщений.*

*Тип данных **НаборКонстант** используется для задания типа данных значений реквизитов формы. Обеспечивает доступ к значениям констант. Реквизит данного типа доступен в модулях с директивой компиляции &НаКлиенте.*

**Константы** – специальный объект (коллекция объектов метаданных), обеспечивающий доступ к значениям констант, хранимых в базе данных. Используются методы

**Константы. <ИмяКонст>.Получить()** – возвращает значение указанной константы (<ИмяКонст>)

**Константы. <ИмяКонст>.Установить(НовоеЗначение)**

## Задание

### 1. Выполнение общего задания

#### 1.1. Создать константы

1.2. "Наименование компании" – строка, "уставной капитал фирмы" – число, "Результат" – число.

1.3. Создать форму констант. (форма должна быть отнесена к одной из подсистем).

1.4. Задать функциональность формы в соответствии с теоретическими положениями. Форма должна отображать значения констант. Создайте на форме команды для вывода значений констант средствами клиентской части и на основе данных информационной базы.

1.5. Создайте на форме управляющие элементы, с которыми свяжите команды.

### 2. Индивидуальное задание

2.1. Задать константу в соответствии с заданием.

2.2. Создать общую форму (Произвольную форму для того, чтобы произвольная форма отображалась в интерфейсе необходимо установить флаг "Использовать" в панели свойств (группа свойств "Представление").

2.3. Создать программный код для вывода в виде сообщения пользователю данных о заданной константе средствами клиентской части и выполнения вычислений.

2.4. Создать программный код для вывода в виде сообщения пользователю данных о заданной константе средствами серверной части и выполнения вычислений.

2.5. Создать *произвольную* форму задать в ней вычисление суммы двух значений. Или другое математическое выражение по заданию преподавателя.

2.6. Заданием является создание констант и общих форм для работы с ними.

Таблица 8 – Варианты заданий

| Номер варианта | Константа                | Функциональность общей формы  |
|----------------|--------------------------|---|
| 1              | ИмяБухгалтера-строка.    | форма должна заносить в константу Результат текст включающий ИмяБухгалтера и Наименование компании. |
| 2              | Год основания фирмы дата | форма должна рассчитывать срок существования фирмы и вводить в константу Результат.                 |
| 3              | Курс доллара             | Расчёт уставного капитала фирмы в долларах и выводить в константу Результат.                        |
| 4              | Произвольное число       | Выполняется заданное арифметическое выражение и выводится в константу – результат.                  |

### Контрольные вопросы

1. Чем отличается "Форма константы" от "Формы констант"? Как можно создать форму констант.
2. Какие типы общих форм выделяются в среде конфигуратора 1с 8.2?
3. Какие части выделяются на окне конструктора формы?
4. Какие элементы участвуют в логике работы формы? Каково их назначение?
5. Какова последовательность их взаимного обращения?
6. Каким образом на форме осуществляется доступ к данным информационной базы. Средствами программных модулей, выполняемых на клиентской части.
7. Какие используются реквизиты?
8. Каким образом на форме осуществляется доступ к данным информационной базы средствами программных модулей, выполняемых на серверной части.
9. Какие используются реквизиты и объекты в том и другом случае

## ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №8. РАБОТА С УПРАВЛЯЕМЫМИ ФОРМАМИ

### Конструктор управляемой формы. Модуль формы

Заполняем табличную часть в режиме исполнения: выбираем номенклатуру, вводим цену, количество. Т.е. все действия, которые выполняются – выполняются интерактивно. Соответственно, обработку необходимо применить к интерактивным действиям пользователя, т.е. соответствующий программный код должен располагаться в модуле формы. Поэтому в окне редактирования объекта переходим на вкладку «Формы» (Рисунок 9).

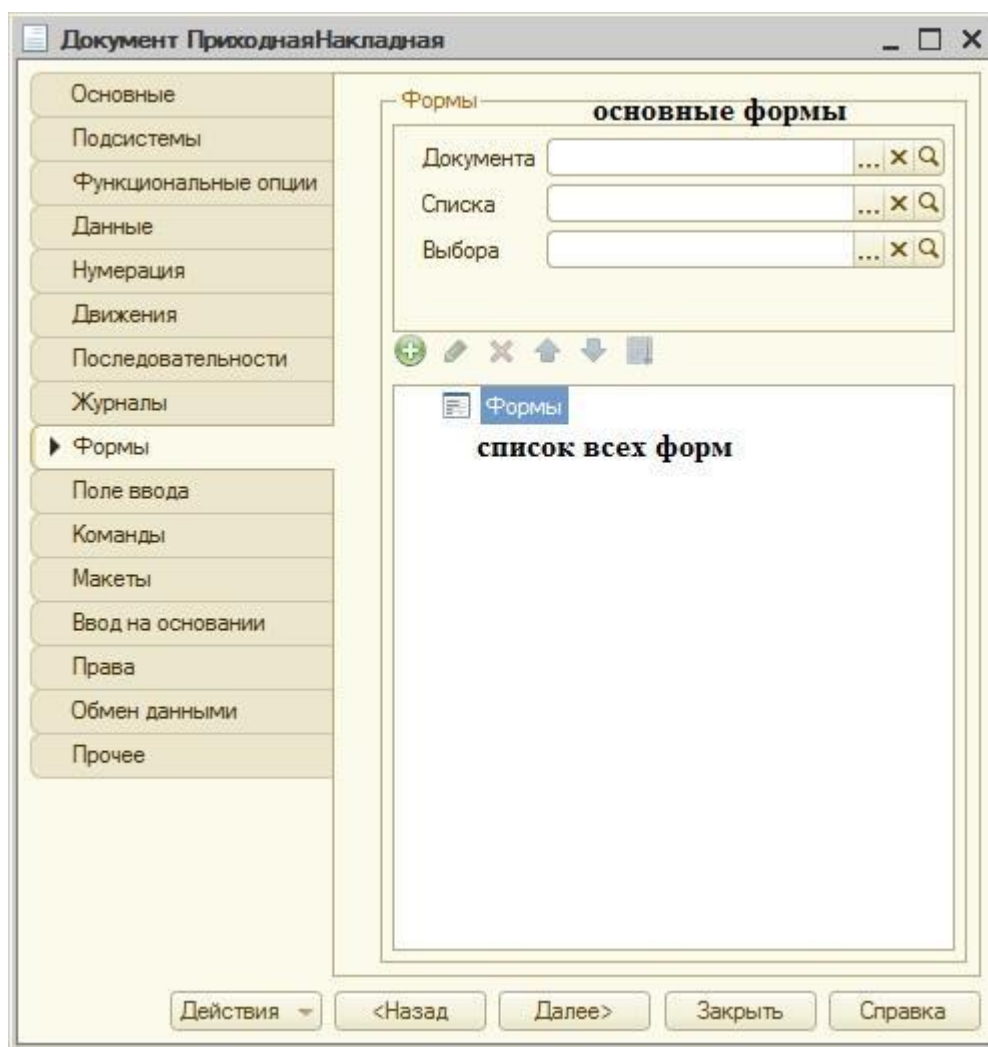





Рисунок 8 - Настройка форм объекта

В верхней части окна «Формы» располагается перечень возможных типов форм (элемента, списка и т.д.) для объекта данного вида. Для каждого типа назначается форма, являющаяся основной (Рисунок 9). Программист может создать сколько угодно большое число форм одного типа, отличающихся функциональностью. Список всех форм располагается ниже панели инструментов (Рисунок 9). Для объекта, из всех форм определенного типа, только одна является основной. Основная форма – форма, загружаемая по умолчанию, при интерактивном обращении к ней в режиме исполнения. Все остальные формы могут загружаться программно в результате работы различных алгоритмов. Для создания основной формы необходимо выбрать требуемый тип формы и нажать  (Рисунок 8), либо выбрать нужную форму из списка всех форм при помощи .

При создании/редактировании экземпляра документа открывается форма элемента (форма документа). Сейчас система ее создает автоматически, нам же данный механизм необходимо изменить. Нажимаем  для создания основной формы документа, в результате запускается специальный мастер. Оставляем все параметры по умолчанию. По окончании работы мастера запускается конструктор формы документа (Рисунок 8.1).

В верхней правой части конструктора формы (Рисунок 8.1) располагаются элементы, отвечающие за данные формы. Сюда относятся, в первую очередь, реквизиты формы, команды формы, которые могут быть расположены в виде кнопок или гиперссылок. Это те данные, которые отвечают за функциональность формы. Также здесь можно добавить параметры формы, которые либо будут передаваться в другие открываемые формы, либо использоваться в различных программных алгоритмах.

В левой верхней части конструктора (Рисунок 8.1) располагается перечень интерактивных элементов – т.е. тех элементов, которые будут расположены на форме, и которые будут доступны для редактирования, использования конечным пользователем. Здесь производится настройка внешнего интерфейса: как элементы будут располагаться, как группироваться и т.д. Интерактивные элементы, выполняющие какие-либо действия, а не являющиеся просто декорациями, связаны с реквизитами и командами формы (через



специальное свойство «ПутьКДанным»). Также здесь настраивается командный интерфейс: панель навигации для перехода к определенным разделам приложения и командная панель.

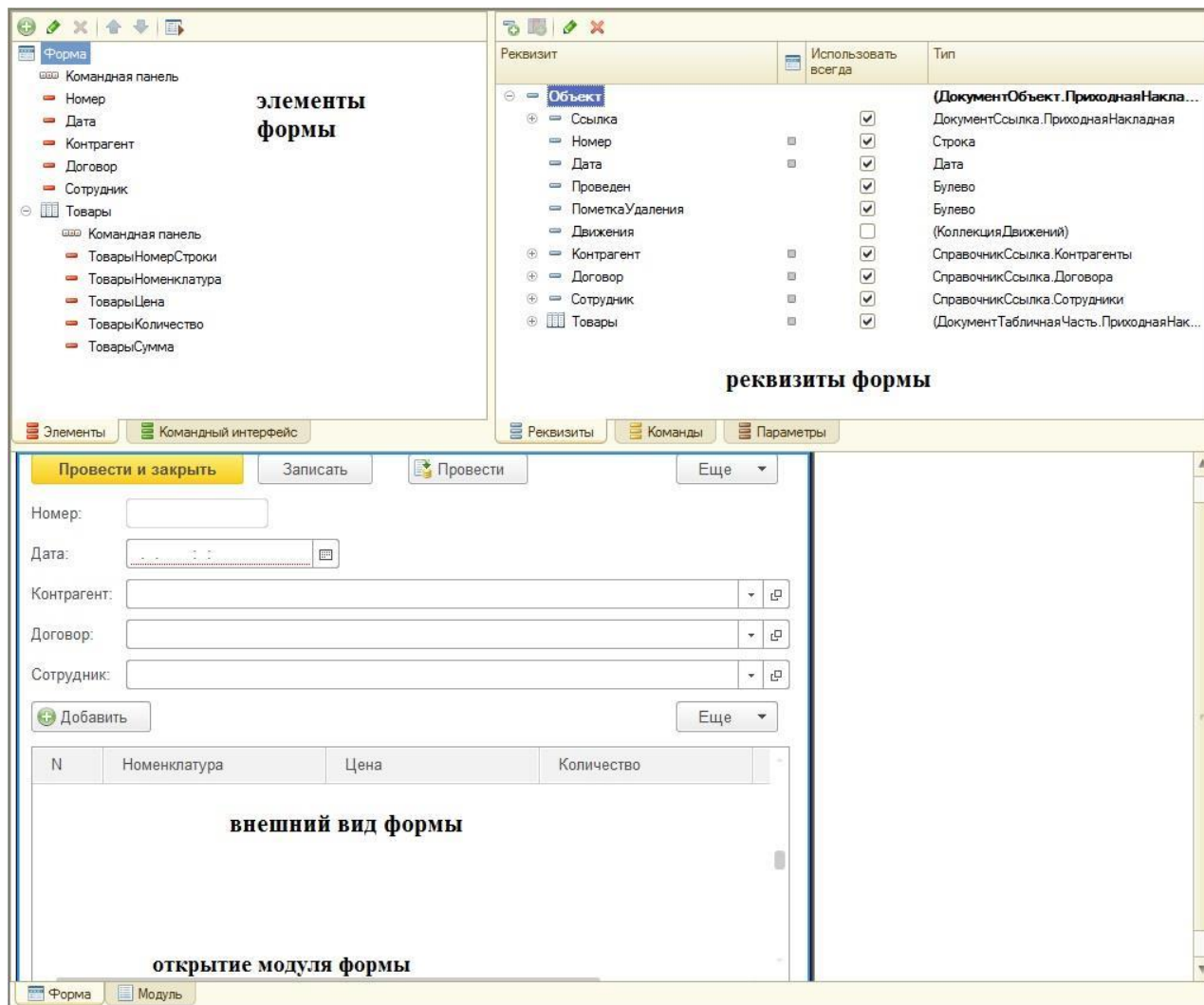


Рисунок 8.1 - Конструктор формы

Внизу (Рисунок 8.1) отображается внешний вид – как создаваемая форма будет выглядеть в режиме исполнения. Также имеется закладка для перехода к модулю формы – это то место, где располагается программный код, описание обработчиков событий. В модуле формы описываются как некоторые служебные (вспомогательные) процедуры и функции, так и обработчики, связанные с элементами формы. Каждый элемент формы (и сама форма в целом) имеет ряд событий, которые могут возникать в

результате работы системы. Для просмотра списка всех событий необходимо открыть свойства конкретного элемента в дереве элементов (см. Рисунок 8.1), и затем перейти к вкладке «События» (Рисунок 8.2).

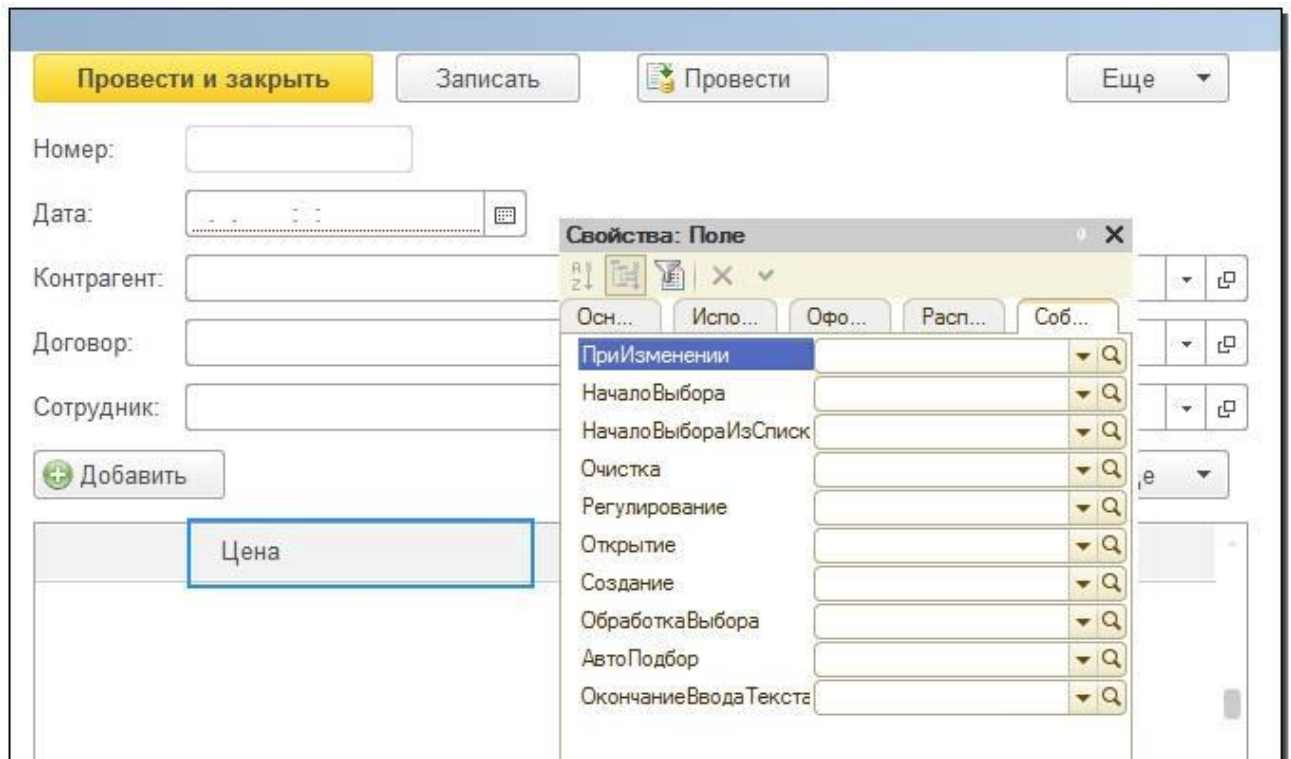




Рисунок 8.2 - События элементов формы

Каждое событие интерактивного элемента формы может быть связано лишь с одной процедурой – обработчиком события. Имя процедуры может быть произвольным. Сами процедуры-обработчики описываются в модуле формы. Для того, чтобы связать событие с процедурой, которая станет обработчиком данного события, используется выпадающий список , в котором из списка всех процедур указывается та, которую следует назначить в качестве обработчика. Для автоматического создания процедуры-обработчика определенного события и связи события и его обработчика, можно использовать кнопку  около требуемого события элемента. Следует обратить внимание, что часть событий выполняется на стороне клиента, а часть – на стороне сервере. Поэтому в модуле формы любую процедуру, функцию (не только обработчики событий)

необходимо предварять директивой, указывающей, где происходит выполнение программного кода.

Для реализации автоматического расчета Суммы, необходимо для элемента «**Цена**» описать обработчик события «**При изменении**». Система задает вопрос, какие процедуры необходимо создать:

- обработчик на клиенте;
- обработчик на клиенте и бесконтекстную процедуру на сервере (для случая, когда необходимо получать какие-то данные с сервера);
- обработчик на клиенте и контекстную серверную процедуры (аналогично предыдущему случаю, но передается весь контекст формы);

Для нас будет достаточно обработчика на клиенте т.к. никакие данные с сервера (из базы данных нам не потребуются). В результате создается шаблон процедуры- обработчика.

Теперь необходимо определить, как обратиться к данным, которые сейчас находятся в табличной части документа? Для этого воспользуемся отладкой: введем в процедуре заглушку  $A=1$ ; и установим в этой строке точку останова. Запустим систему в режиме отладки и попытаемся создать один документ «**Приходная накладная**».

Добавим строку табличной части и введем в поле «**Цена**», допустим, «**5**». Система перебрасывает нас в модуль объекта в точку останова. Используем инструмент «Вычислить выражение». Мы можем попытаться обратиться к значениям реквизитов или элементов формы.

Все реквизиты хранятся в основном реквизите – «**Объект**». В поле «Вычислить» вводим «Объект», в результате получаем значения реквизитов (Рисунок 8.3).

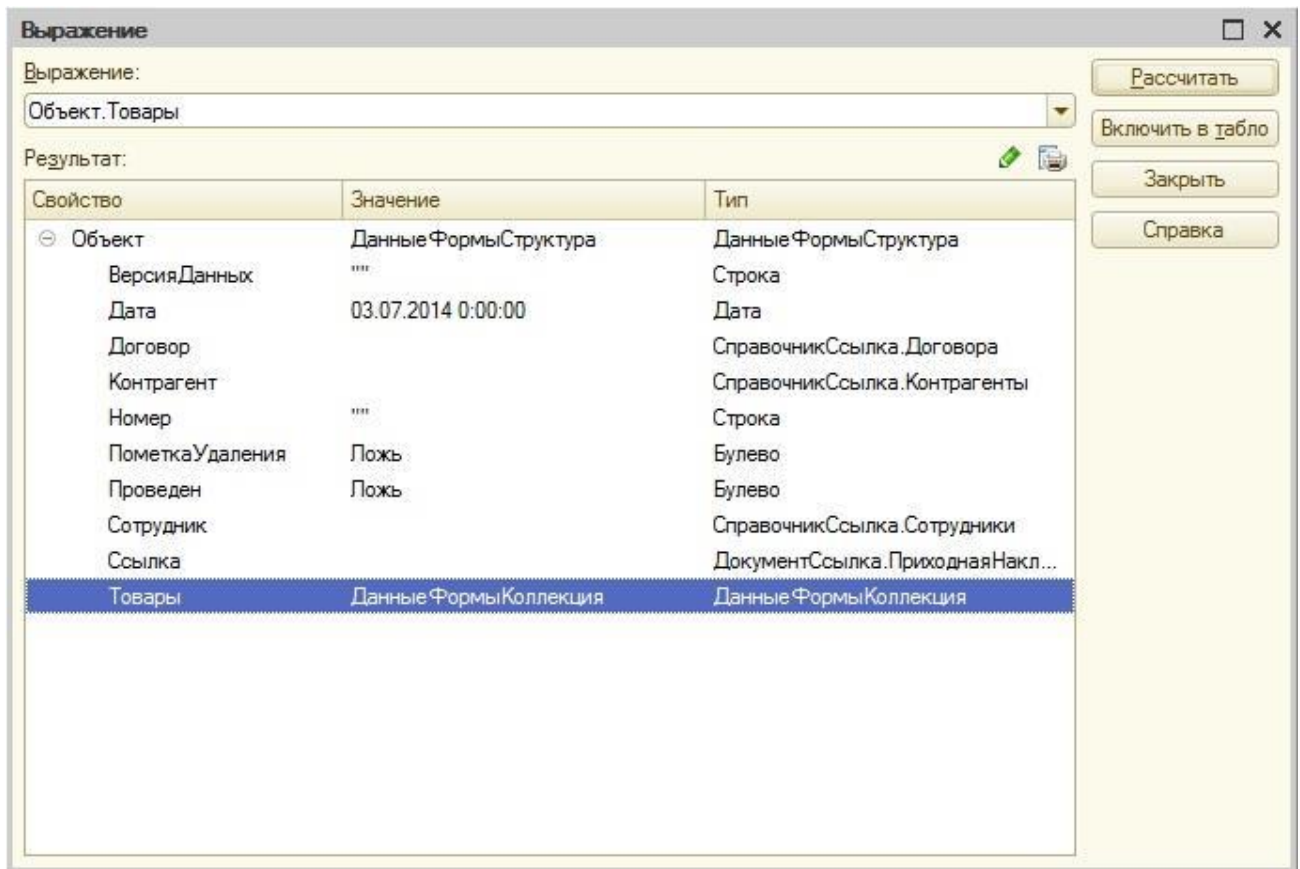


Рисунок 8.3 - Содержимое реквизитов документа


Однако тем самым мы не получили желаемого: мы не видим строку табличной части с ценой. Известно, что табличная часть называется «Товары», поэтому находим ее среди всех полученных реквизитов (Рисунок 8.3). Данный реквизит имеет тип ДанныеФормыКоллекция. Коллекцию значений мы можем посмотреть в отдельном окне при помощи кнопки , в результате чего открывается табличная часть «Товары» (Рисунок 8.4).



Рисунок 8.4 - Просмотр табличной части в отдельном окне

Здесь мы видим наш товар с заданной ценой, но опять – нет ничего, чтобы указывало на то, как к этому полю обратиться. Пробуем просмотреть соответствующую запись также в отдельном окне (Рисунок 8.5).

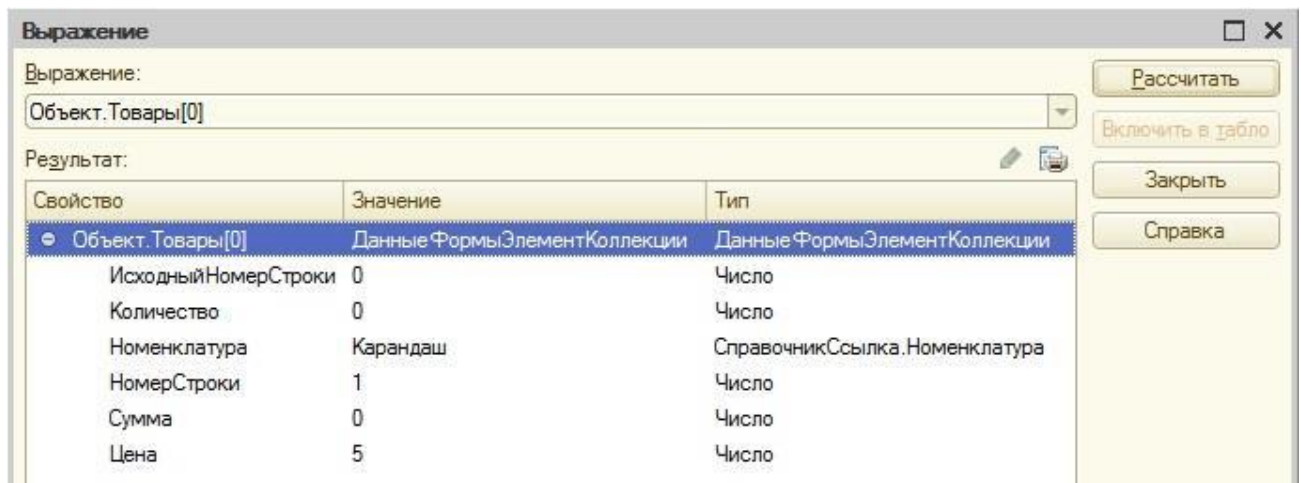


Рисунок 8.5 - Просмотр выделенной строки табличной части в отдельном окне

В результате срабатывает инструмент «Вычислить выражение» для конкретной строки табличной части: **Объект.Товары[0]**. Данный вариант невозможен к использованию, т.к. строк может быть великое множество, и нам будет неизвестно, в какой конкретно строке мы находимся в данный момент. Таким образом, поиск решения среди реквизитов объекта не дал желаемого результата.

Для обращения к элементам формы следует обратиться к коллекции «Элементы». В результате мы получим окно, содержащее гораздо больше элементов, доступных для просмотра (Рисунок 8.6).

Это на самом деле ожидаемо т.к. элементов пользовательского интерфейса достаточно много, настроек, параметров и характеристик – тоже. Главная проблема – как среди всего этого множества отыскать то, что требуется именно нам.

В первую очередь поиск необходимо начинать по основному наименованию элемента.

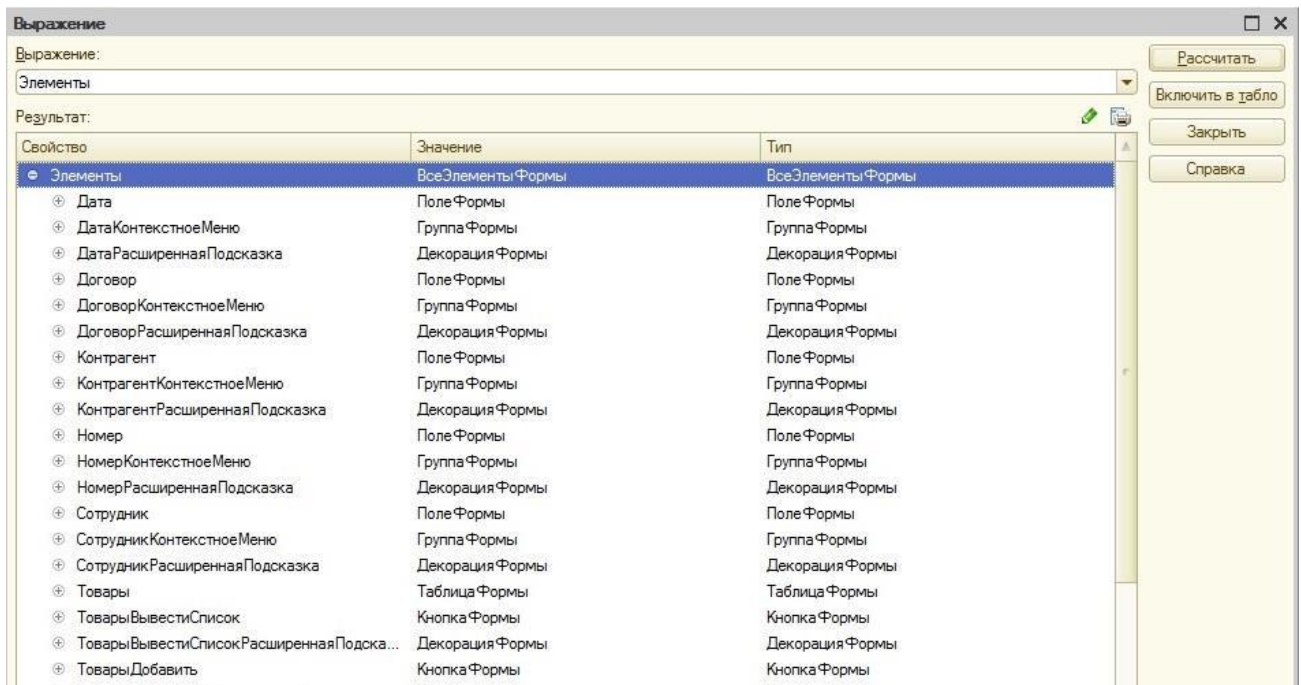


Рисунок 8.6 - Список интерактивных элементов, доступных для просмотра при отладке

Табличная часть называется «**Товары**», поэтому в первую очередь и ищем – **Элементы.Товары**. Раскроем найденную группу «**Товары**». Табличная часть имеет тип – **ДанныеФормыКоллекция** (см. Рисунок 8.6), а конкретная строка табличной части – **ДанныеФормыЭлементКоллекции**. Среди перечня всех элементов найдем элемент соответствующего типа. Это элемент под названием **ТекущиеДанные**. Раскрыв данный элемент, мы увидим данные в строке, в которой мы в данный момент находимся. Более того, выделив поле «**Цена**», в поле «Выражение» инструмента «Вычислить выражение» мы увидим полный путь к этому полю: **Элементы.Товары.ТекущиеДанные.Цена**, что нам и требовалось получить. Аналогичным образом, осуществляется доступ к содержимому полей «**Количество**» и «**Сумма**».

Закрываем отладку приложения, возвращаемся к редактированию модуля формы документа. В созданной процедуре-обработчике убираем точку останова и текст-заглушку. Пишем следующий программный код:


СтрокаРасчета = Элементы.Товары.ТекущиеДанные;



СтрокаРасчета.Сумма = СтрокаРасчета.Цена \* СтрокаРасчета.Количество;

Первая строчка содержит в себе строку-ссылку на текущие данные (введена для большей читабельности кода). Вторая строка производит требуемые действия с элементами, входящими в указанную строку-ссылку.

Аналогичным образом создадим обработчик события «ПриИзменении» для поля «Количество» и добавим в него точно такие же строчки программного кода. Запустим систему в режиме отладки и попробуем добавить приходную накладную теперь. В конце нажимаем кнопку «Записать» (без проведения) после чего закрываем текущую приходную.

Создадим документ «**Расходная накладная**» копированием. В системе «1С: Предприятие» очень сильно развит инструмент создания объектов (прикладных и подчиненных) «на основании» других. Для того чтобы просто создать точную копию объекта, необходимо в дереве конфигурации выделить требуемый объект и нажать кнопку «Добавить копированием» , либо использовать функциональную клавишу F9. После этого необходимо изменить имя прикладного объекта, и выполнить требуемую настройку реквизитов и поведения объекта. Реквизиты шапки, табличные части и их реквизиты объекта также можно добавлять копированием.

Также имеется возможность создания на основе прикладного объекта одного вида, прикладного объекта другой сущности (не стоит путать с инструментом «Ввод на основании», далее в пособии). К примеру, из объекта типа «Справочник» создать объект типа «Документ». Для этого необходимо в дереве конфигурации выделить исходный объект и перетащить его на класс объектов, экземпляр которого необходимо создать. При этом скопируются только те настройки, реквизиты, табличные части (и их реквизиты), которые могут существовать у объектов различных классов и имеют одинаковый тип данных.

В нашем примере мы не будем создавать копированием объекты различных сущностей. Выделим документ «**Приходная накладная**» и при помощи «Добавить копированием» создадим новый документ «**Расходная накладная**». Включим документ в подсистему «Отдел

**продаж**». Для реквизита **«Контрагент»** изменим настройку **«Параметры выбора»**: теперь мы должны выбирать контрагентов из списка **«Покупателей»**. Для реквизита **«Номенклатура»** табличной части **«Товары»** сбросим настройку **«Параметры выбора»** – мы продаем не только товары, но и оказываем услуги, и все это делаем в рамках одного документа **«Расходная накладная»**.

После того, как мы внесли все необходимые изменения, можно запустить систему в режиме отладки и посмотреть на результат. Если вдруг оказалось, что при создании документа копированием, система сохранила условия отбора, накладываемые на формы выбора, как у исходного документа (к примеру, можно выбрать только контрагентов из списка поставщиков) – тогда необходимо удалить форму документа **«Расходная накладная»** и просто заново ее создать с описанием требуемого функционала.

Обратим внимание на следующее: в модуле формы документа **«Приходная накладная»** для события ПриИзменении полей **«Цена»** и **«Количество»** был прописан одинаковый программный код. Создав копированием документ **«Расходная накладная»**, данный код также оказался дважды написан в модуле документа. Таким образом, одни и те же действия несколько раз были описаны в различных частях конфигурации. Это не совсем рационально. Правильнее этот программный код описать в одном единственном месте (в общем модуле), а в остальных частях конфигурации делать вызов соответствующей процедуры/функции.

Создадим общий модуль **«Работа с документами»**, который будет выполняться исключительно на клиенте (установите соответствующую галочку в настройках модуля).

Добавим в модуль следующую процедуру:

```
Процедура ПересчитатьСумму (СтрокаРасчета) Экспорт
СтрокаРасчета.Сумма = СтрокаРасчета.Цена * СтрокаРасчета.
Количество;
КонецПроцедуры
```

СтрокаРасчета – параметр процедуры, передаваемый ей в точке вызова. В модуле объекта для документов **«Приходная накладная»**,



**«Расходная накладная»** для имеющихся обработчиков событий изменим программный код на следующий:

СтрокаРасчета = Элементы.Товары.ТекущиеДанные;

РаботаСДокументами.ПересчитатьСумму(СтрокаРасчета);

В первой строке кода мы получаем текущие данные, для которых нужен обработчик.

Во второй строке мы вызываем процедуру из общего не глобального модуля, и передаем эти данные через параметр процедуры.

При написании программного кода обратите внимание на то, что система автоматически выдает список конструкций, которые могут быть. Также при описании прикладных объектов, при нажатии на точку система показывает список возможных реквизитов объекта, из которых можно выбрать. Вы всегда можете ввести первые символы конструкции (или прикладного объекта, реквизита и т.д.), которую хотите использовать, после чего нажать комбинацию клавиш «Ctrl+Пробел». В результате система выполнит синтаксический поиск по первым введенным символам и выведет список возможных методов, событий, объектов, реквизитов и т.д., которые можно использовать; вам будет достаточно в появившемся списке выбрать подходящий вариант.

Это позволяет, во-первых, разрабатывать код более быстро и эффективно, а, во-вторых, снижает вероятность возникновения ошибок из-за опечаток при наборе. Однако это будет действовать не всегда: например, вы присвоили переменной имя объекта. Не факт, что система всегда сможет определить реквизиты данной переменной, как и объекта; у исходного объекта она обязана это делать, но если было переименование, то это не совсем обязательно. В данном случае вам необходимо выполнять контроль корректности ввода данных самостоятельно.

### Задание

1. Создать форму констант.
2. Настроить форму добавив соответствующие элементы.
3. Убедиться, что данная форма работает корректно.

**Контрольные вопросы**

1. Что такое форма?
2. Как создать форму?
3. Какие элементы нужны для работы формы?
4. Как связать элементы формы для их корректной работы?

## ЛАБОРАТОРНАЯ РАБОТА №2. ИСПОЛЬЗОВАНИЕ МАКЕТА ПЕЧАТНОЙ ФОРМЫ

### Макет печатной формы

Практически всегда в прикладных задачах есть необходимость привязать к определенному документу *Печатную форму*. Форма может представлять из себя некоторый бланк, который пользователь будет распечатывать после записи или проведения документа. В данной части мы научимся создавать печатные формы.

#### Макет.

Вывод любой печатной формы выполняется с помощью макетов. Всего существует девять типов макетов, но мы с Вами в этой книге рассмотрим только один тип – табличный документ.

Сейчас Вы создадите какой-нибудь макет к документу *Прибытие в гараж* нашей конфигурации.

Откройте конфигурации и раскройте в дереве документ *Прибытие в гараж* и обратите внимание на пункт «*Макеты*».

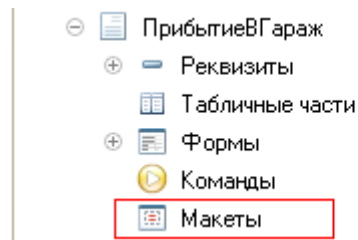


Рисунок 9 – Дерево документа

Кликните правой кнопкой мышки и выберите «*Добавить*».



Рисунок 9.1 – Добавить Макет

Открывается конструктор макетов. В данном конструкторе мы ничего не меняем, но обращаем внимание, что тип макета выбран «*Табличный документ*».

**Конструктор макета**

Имя:

Синоним:

Комментарий:

Выберите тип макета:

☒ Табличный документ

☐ Текстовый документ

☐ Двоичные данные

☐ Active document

☐ HTML документ

☐ Географическая схема

☐ Графическая схема

☐ Схема компоновки данных

☐ Макет оформления компоновки данных

Загрузить из файла:

Рисунок 9.2 –Конструктор Макета

Нажмите кнопку «Готово», и Вы создали макет под названием «Макет».

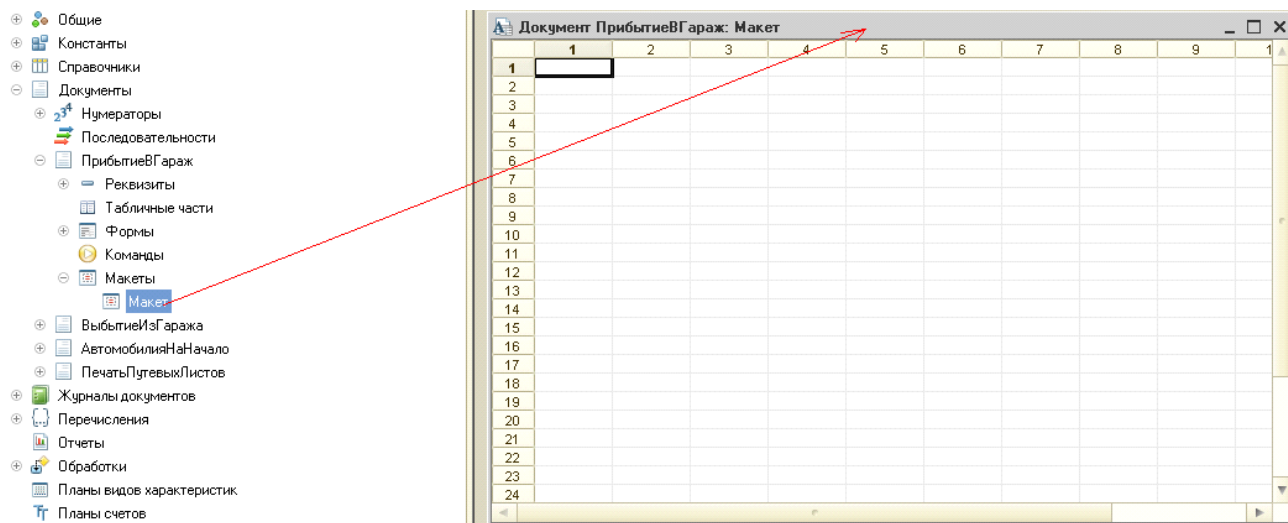


Рисунок 9.3 – Макет

Как видите, макет табличного документа представляет собой форму наподобие экселевской. Она также состоит из строк, столбцов и ячеек.

Каждая ячейка имеет свои уникальные свойства (кликните правой кнопкой мышки и в выпадающем меню выберите пункт *Свойства*)

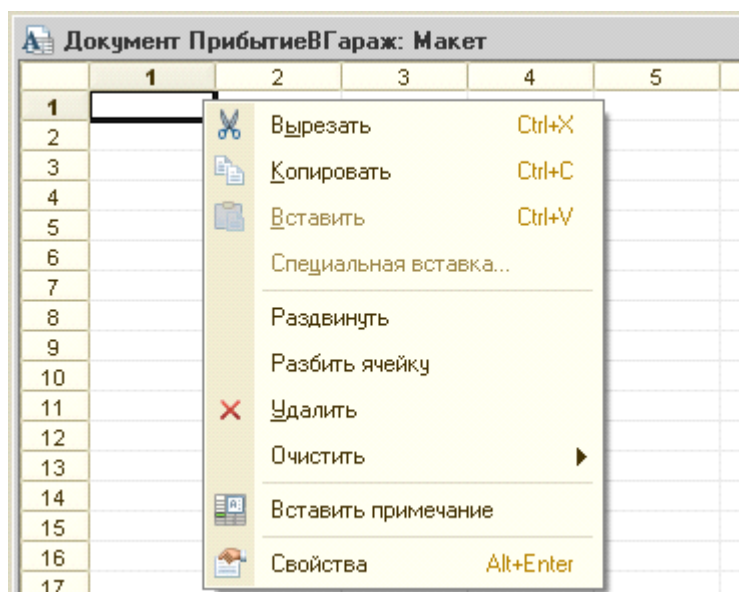


Рисунок 9.4 – Макет табличного документа

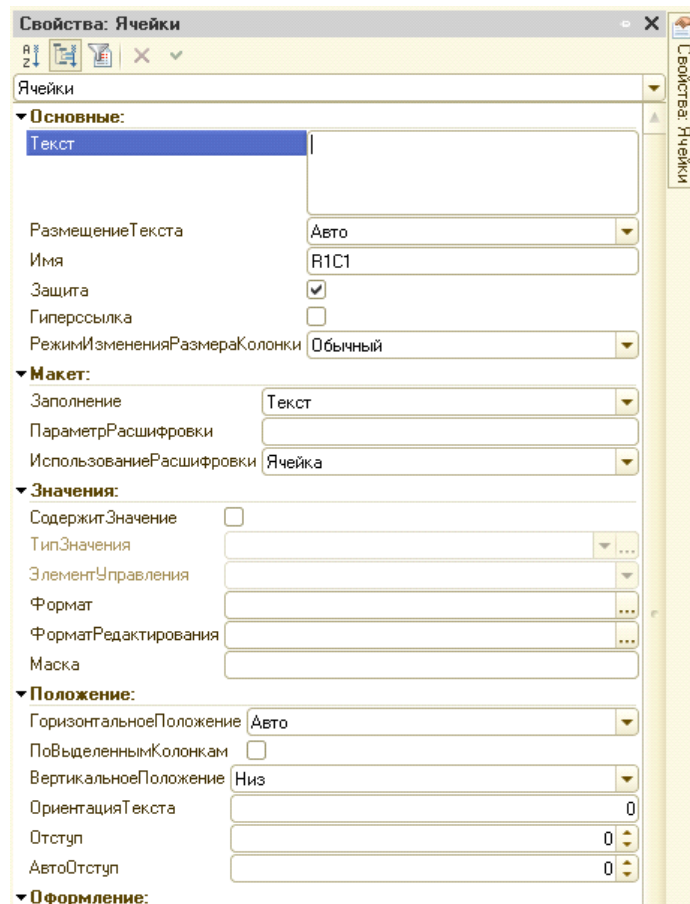


Рисунок 9.5 – Свойства ячейки

Внутри любой ячейки можно разместить текст, разместите в ячейке 2 колонки, 2-го столбца текст: Прибытие автомобиля №.

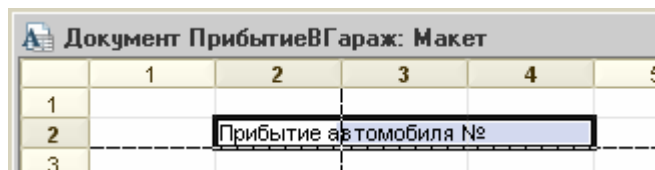
|   | 1 | 2                     | 3 | 4 |
|---|---|-----------------------|---|---|
| 1 |   |                       |   |   |
| 2 |   | Прибытие автомобиля № |   |   |
| 3 |   |                       |   |   |

Рисунок 9.6 – Разметка

Вы видите, что наш текст не уместился на одной ячейке, а сразу перешел на вторую, и появились две линии по краям ячейки – горизонтальная и вертикальная. Эти линии ограничивают область, которая будет выводиться на экран.

Понятно, что строка «Прибытие автомобиля» будет выведена не полностью. Чтобы такого не произошло, объединим ячейки, и

увеличим шрифт. Для этого, необходимо их выделить (нажать левую кнопку мышки и вести по ячейкам курсор с нажатой кнопкой).



|   | 1 | 2                     | 3 | 4 | 5 |
|---|---|-----------------------|---|---|---|
| 1 |   |                       |   |   |   |
| 2 |   | Прибытие автомобиля № |   |   |   |
| 3 |   |                       |   |   |   |

Рисунок 9.7 – Разметка

После перейдите в меню «Таблица» и нажмите на кнопку «Объединить».

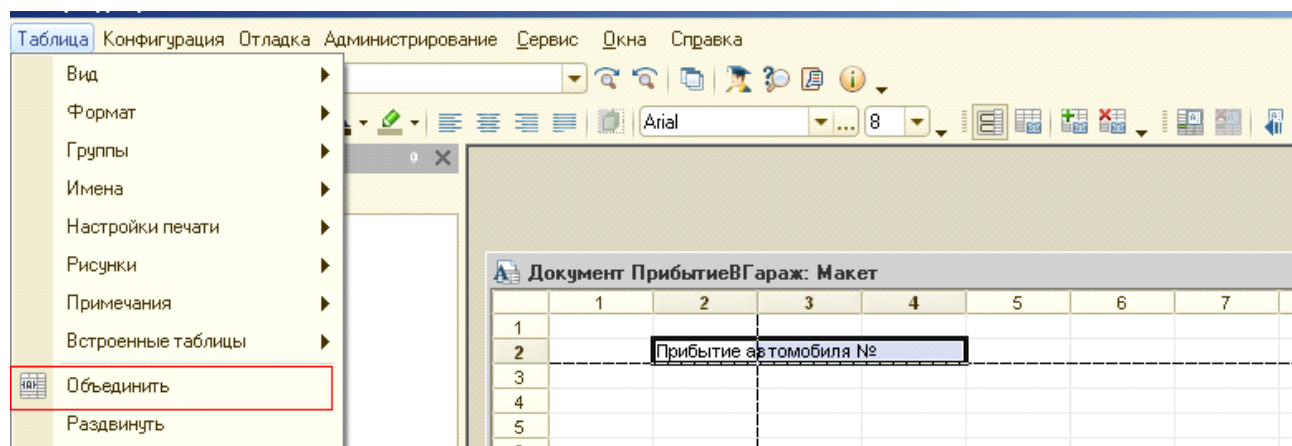


Рисунок 9.8 – Разметка

Видите, ячейка стала одна. И границы сдвинулись.

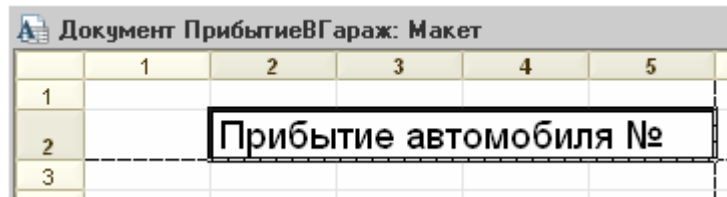
|                                |   |                       |   |   |
|--------------------------------|---|-----------------------|---|---|
| Документ ПрибытиеВГараж: Макет |   |                       |   |   |
|                                | 1 | 2                     | 3 | 4 |
| 1                              |   |                       |   |   |
| 2                              |   | Прибытие автомобиля № |   |   |
| 3                              |   |                       |   |   |

Рисунок 9.9 – Разметка

Теперь измените величину шрифта на 14.

Ячейка увеличилась, и текст опять не помещается. Снова объедините ячейки с двумя соседними. Для этого нужно снять имеющееся объединение, так же выделив ячейки и нажав на кнопку

«Объединить» в меню «Таблица». И заново объединить в этот раз четыре ячейки.



|   | 1 | 2                     | 3 | 4 | 5 |
|---|---|-----------------------|---|---|---|
| 1 |   |                       |   |   |   |
| 2 |   | Прибытие автомобиля № |   |   |   |
| 3 |   |                       |   |   |   |

Рисунок 9.10 – Результат изменения шрифта

Помните: все, что Вы напишите в данной таблице, будет, при определенных условиях, выведено в печатную форму.

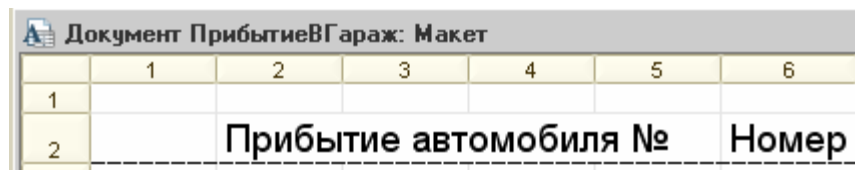
Теперь нам необходимо передать номер и дату документа прибытия в макет.

Как это сделать? Делается это достаточно просто: необходимо изменить свойство

«Заполнение ячейки».

Внесите текст в следующую ячейку после объединенной, и измените шифр на 14.

Внесите текст в следующую ячейку после объединенной, и измените шифр на 14.



|   | 1 | 2                     | 3 | 4 | 5     | 6 |
|---|---|-----------------------|---|---|-------|---|
| 1 |   |                       |   |   |       |   |
| 2 |   | Прибытие автомобиля № |   |   | Номер |   |

Рисунок 9.11 – Результат изменения ячейки

Далее Вам нужно сделать так, чтобы в ячейку можно было передавать номер документа (как это делается, мы изучим немного позже). А пока настроим ячейку на такую возможность. Для этого откройте свойства ячейки: выделите ее и кликните правой кнопкой мышки, выйдет контекстное меню, где выберите пункт *Свойства*.



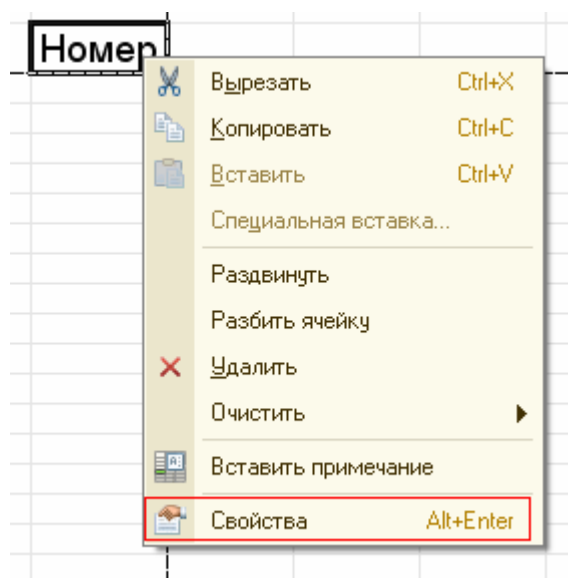


Рисунок 9.12 – Свойства для ячейки

Откроются свойства этой ячейки, и в них обратите внимание на свойство «Заполнение».

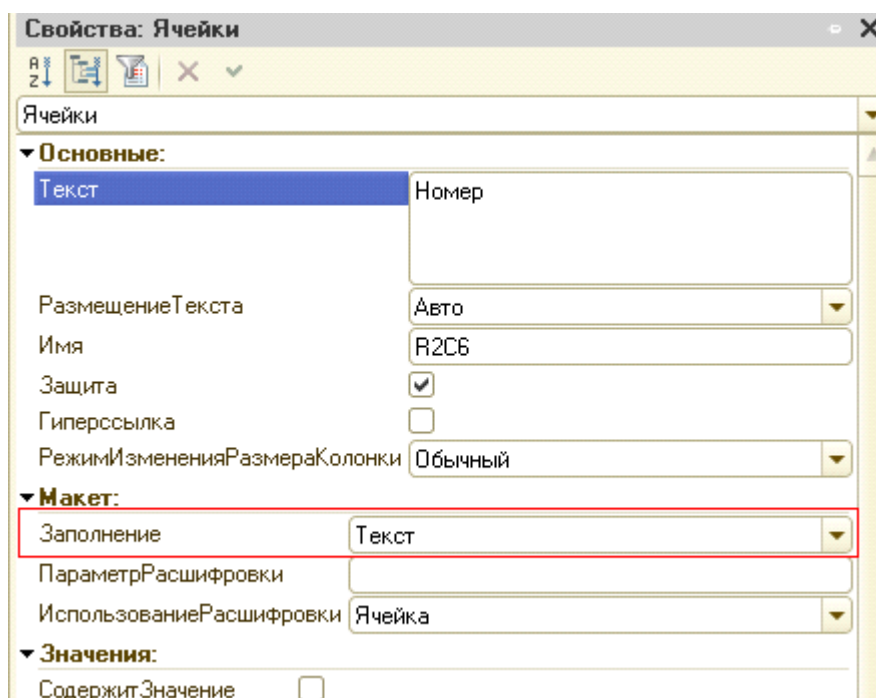


Рисунок 9.13 – Свойства ячейки

Измените его значение в «*Параметр*».

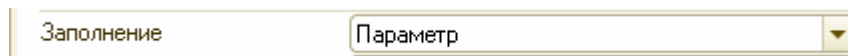


Рисунок 9.14 – Свойство ячейки Заполнение

После этого, как Вы видите, слово «*Номер*» закрылось своеобразными скобками.

|   | 1 | 2 | 3                     | 4 | 5 | 6       |
|---|---|---|-----------------------|---|---|---------|
| 1 |   |   |                       |   |   |         |
| 2 |   |   | Прибытие автомобиля № |   |   | <Номер> |
| 3 |   |   |                       |   |   |         |

Рисунок 9.15 – Результат применения Параметра

Теперь добавьте еще и дату. Ячейке со словом *Дата* также поставьте свойство «Заполнение» в значение «Параметр».

|   | 1 | 2 | 3                     | 4 | 5       | 6 | 7  | 8      |
|---|---|---|-----------------------|---|---------|---|----|--------|
| 1 |   |   |                       |   |         |   |    |        |
| 2 |   |   | Прибытие автомобиля № |   | <Номер> |   | от | <Дата> |
| 3 |   |   |                       |   |         |   |    |        |

Рисунок 9.16 – Результат добавления даты

Сразу же установите формат даты в удобный для восприятия, чтобы потом не возвращаться к этому в коде.

Для этого зайдите также в свойства и перейдите в свойство «*Формат*».

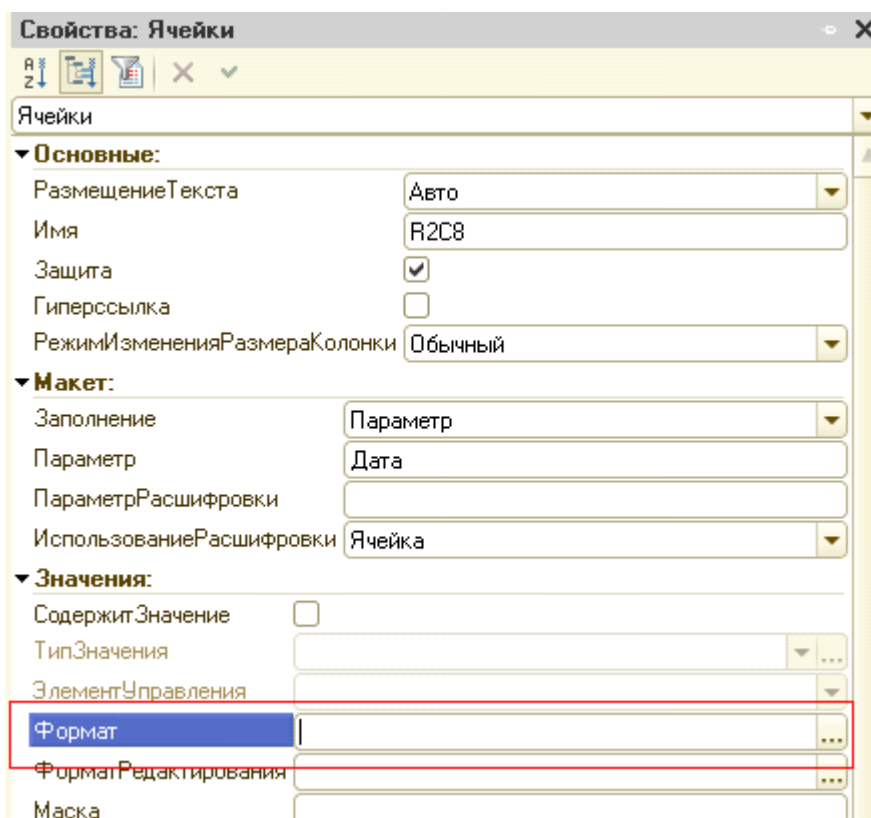


Рисунок 9.17 – Свойство ячейки Формат

Выбираем нужный Вам формат даты.

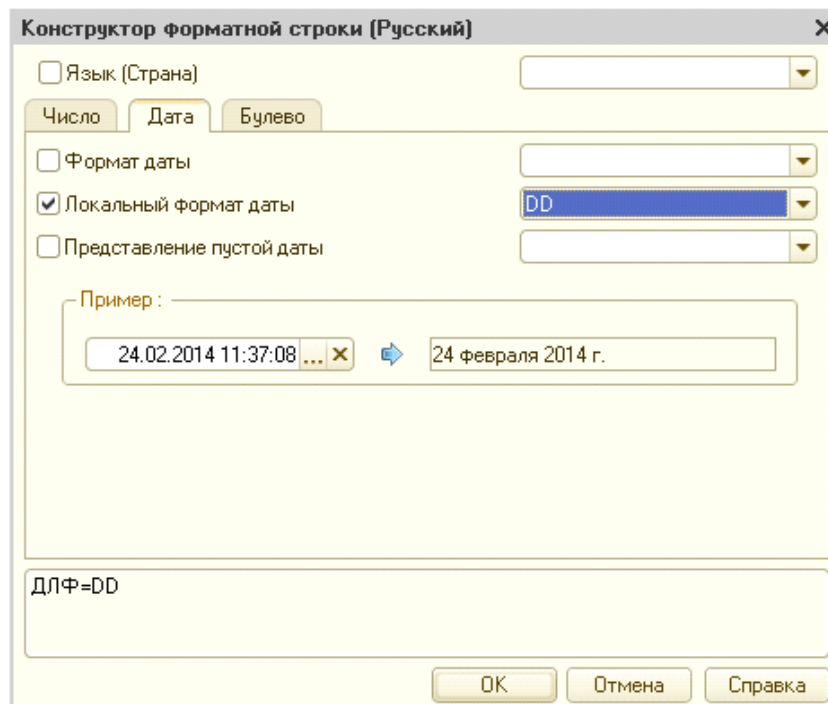


Рисунок 9.18 – Конструктор форматной строки



Рисунок 9.19 – Конструктор форматной строки

Теперь у многих из вас может возникнуть вопрос: а можно ли передавать параметр внутрь текста? Да можно, для этого необходимо устанавливать свойство «Заполнение» в значение «Шаблон».

Напишите через строку следующий текст. Поставьте 12 шрифт и объедините ячейки.

| Документ ПрибытиеВГараж: Макет |   |  |   |   |         |   |           |   |
|--------------------------------|---|--|---|---|---------|---|-----------|---|
|                                | 1 | 2  | 3 | 4 | 5       | 6 | 7         | 8 |
| 1                              |   |  |   |   |         |   |           |   |
| 2                              |   | Прибытие автомобиля №                              |   |   | <Номер> |   | от <Дата> |   |
| 3                              |   |  |   |   |         |   |           |   |
| 4                              |   | [Автомобиль] прибыл в гараж [Гараж] [ДатаПрибытия] |   |   |         |   |           |   |
| 5                              |   |  |   |   |         |   |           |   |

Рисунок 9.20 – Результат выполнения объединения

Перейдите в свойства объединенной ячейки и установим свойство «Заполнить» в «Шаблон».

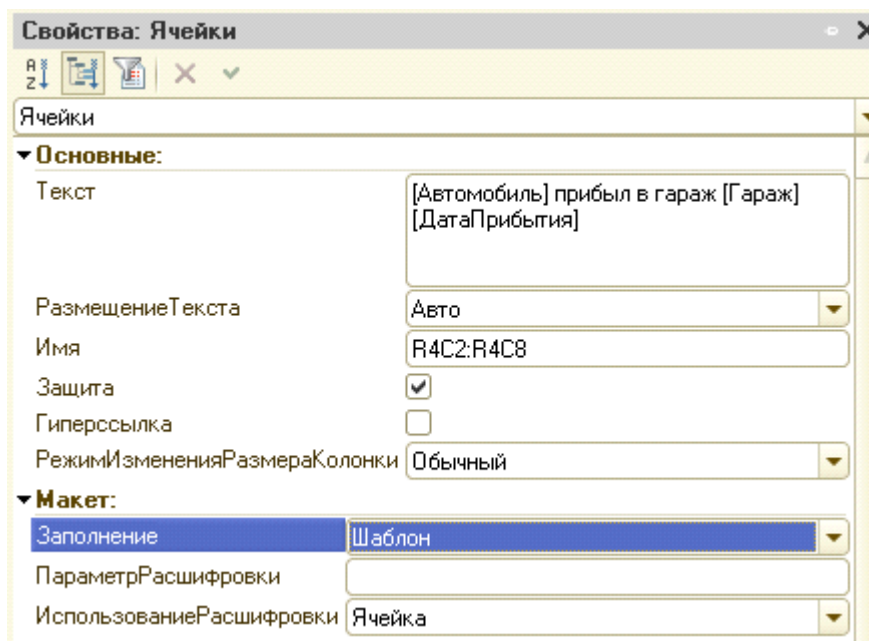


Рисунок 9.21 – Свойство ячейки «Заполнить» в «Шаблон».

Все слова, которые находятся в квадратных скобках, будут передаваться в виде параметров. Для ячеек можно устанавливать границы в виде различных линий. Делается это с помощью свойств «Граница слева», «Граница сверху», «Граница справа», «Граница снизу».

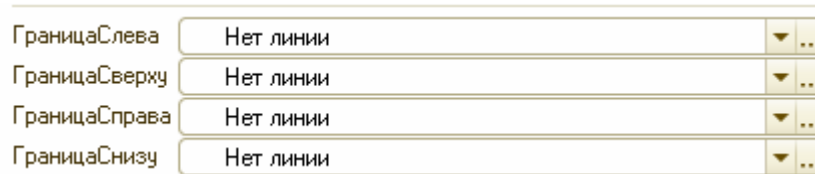


Рисунок 9.22 – Свойства Границы ячейки.

Поставьте на несколько нижних ячеек границы снизу, напомним текст под ними:

| Документ ПрибытиеВГараж: Макет |   |  |   |   |         |   |    |        |
|--------------------------------|---|--|---|---|---------|---|----|--------|
|                                | 1 | 2  | 3 | 4 | 5       | 6 | 7  | 8      |
| 1                              |   |  |   |   |         |   |    |        |
| 2                              |   | Прибытие автомобиля №                                |   |   | <Номер> |   | от | <Дата> |
| 3                              |   |  |   |   |         |   |    |        |
| 4                              |   | <[Автомобиль] прибыл в гараж [Гараж] [ДатаПрибытия]> |   |   |         |   |    |        |
| 5                              |   |  |   |   |         |   |    |        |
| 6                              |   |  |   |   |         |   |    |        |
| 7                              |   | Ответственный  |   |   | Подпись |   |    |        |
| 8                              |   |  |   |   |         |   |    |        |

Рисунок 9.23 – Граница ячейки.

В целом макет готов, но это еще не все. Для того, чтобы было удобнее выводить макет, принято использовать области. Зададим одну область и назовем ее *Основная*.

Для этого выделите те ячейки, которые мы хотим поместить в область.

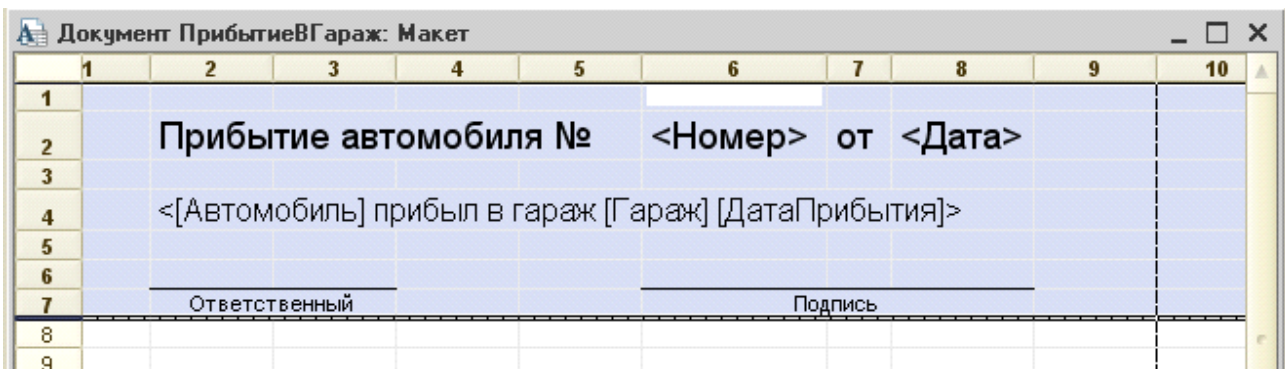


Рисунок 9.24 – Область выделения.

После этого перейдите в меню (область должна быть выделена) «Таблица»– «Имена»– «Назначить имя».

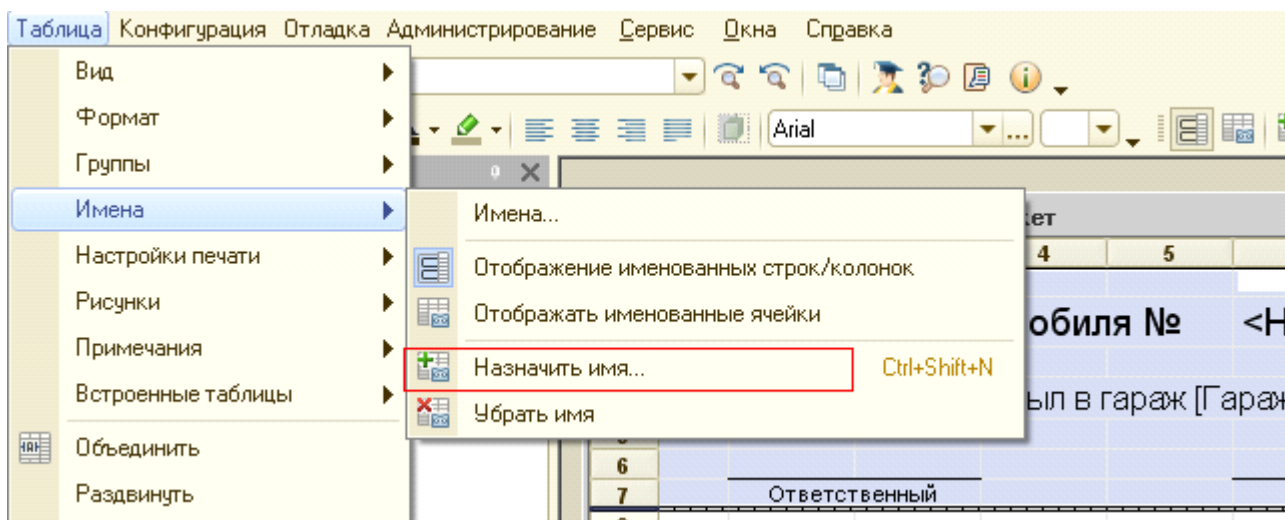


Рисунок 9.25 – Назначение имени.

Создайте область и назовите ее *Основная*.



Рисунок 9.26 – Назначение имени.

Макет должен иметь следующий вид:

|          | 1 | 2  | 3 | 4 | 5 | 6       | 7 | 8  | 9 | 10     |
|----------|---|--|---|---|---|---------|---|----|---|--------|
| Основная | 1 |  |   |   |   |         |   |    |   |        |
|          | 2 | Прибытие автомобиля №                                |   |   |   | <Номер> |   | от |   | <Дата> |
|          | 3 |  |   |   |   |         |   |    |   |        |
|          | 4 | <[Автомобиль] прибыл в гараж [Гараж] [ДатаПрибытия]> |   |   |   |         |   |    |   |        |
|          | 5 |  |   |   |   |         |   |    |   |        |
|          | 6 |  |   |   |   |         |   |    |   |        |
|          | 7 | Ответственный  |   |   |   | Подпись |   |    |   |        |
|          | 8 |  |   |   |   |         |   |    |   |        |

Рисунок 9.27 – Результат после назначения имени.

Все, мы создали самый примитивнейший макет. Теперь необходимо вывести эту информацию на экран при нажатии на какую-нибудь кнопку на форме.

Откройте форму документа *Прибытие в гараж*.

Рисунок 9.28 – Форма Документа

Создайте в нижней командной панели кнопку и назовите ее «*Печать*».

Рисунок 9.29 – Создание кнопки «*Печать*»

Теперь в обработчике нажатия кнопки «*Печать*» напишем следующий код:

```

Процедура ОсновныеДействияФормыПечать (Кнопка)
    Макет = ПолучитьМакет ("Макет");
    ОбластьОсновная = Макет.ПолучитьОбласть ("Основная");

    ОбластьОсновная.Параметры.Номер = Номер;
    ОбластьОсновная.Параметры.Дата = Дата;
    ОбластьОсновная.Параметры.Автомобиль = Автомобиль;
    ОбластьОсновная.Параметры.Гараж = Гараж;
    ОбластьОсновная.Параметры.ДатаПрибытия = формат (ДатаПрибытия, "дд.мм.гг");
    ТабДокумент = Новый ТабличныйДокумент;
    ТабДокумент.Вывести (ОбластьОсновная);
    ТабДокумент.Показать ();

КонецПроцедуры

```

Рисунок 9.30 – Код кнопки «Печать»

Разберем данный код.

В первой строке мы получили макет с помощью метода объекта документа *ПолучитьМакет*. В качестве параметра мы передали название макета, которое указано в конфигураторе.

Во второй строке мы получаем область, которую задали в макете, указав ее название.

Обращаю Ваше внимание, что можно и не задавать имя области, а просто указать диапазон

ячеек. Но на начальном этапе, да и в дальнейшем, лучше все делать через имена областей, т.к. это гораздо более удобно для доработок и чтения кода.

Теперь нам нужно заполнить параметры области, которые мы задали в макете.

Делается это с помощью свойства области «*Параметры*».

Данное свойство представляет объект *Параметры макета табличного документа*, который является коллекцией параметров макета.

К параметрам макета можно обращаться через квадратные скобки, а проще напрямую, как в данном примере.

После того, как мы заполнили параметры области, нам необходимо вывести ее.

Делается это с помощью объекта *Табличный документ*. Данный объект предназначен для вывода табличных форм на экран. Он создается с помощью оператора *Новый*.



С помощью метода *Вывести* табличного документа добавляем основную область в общий документ.

Прежде чем вывести общий табличный документ, его необходимо скомпоновать из имеющихся в макете областей. Дело в том, что областей может быть несколько, некоторые из них должны будут повторяться, а некоторые, наоборот, отсутствовать при определенных условиях. Поэтому с помощью данного метода и формируется результирующий документ.

После этого мы с помощью метода *Показать* выводим его на экран.

Сохраните конфигурацию, и посмотрим, как выходит печатная форма.

|    | 1 | 2   | 3 | 4 | 5 | 6                   | 7 | 8 | 9 | 10 | 11 |
|----|---|---|---|---|---|---------------------|---|---|---|----|----|
| 1  |   |   |   |   |   |                     |   |   |   |    |    |
| 2  |   | Прибытие автомобиля № 000000000                             |   |   |   | от 3 января 2013 г. |   |   |   |    |    |
| 3  |   |   |   |   |   |                     |   |   |   |    |    |
| 4  |   | Автомобиль гл. буха прибыл в гараж Главный 3 января 2013 г. |   |   |   |                     |   |   |   |    |    |
| 5  |   |   |   |   |   |                     |   |   |   |    |    |
| 6  |   |   |   |   |   |                     |   |   |   |    |    |
| 7  |   | Ответственный   |   |   |   | Подпись             |   |   |   |    |    |
| 8  |   |   |   |   |   |                     |   |   |   |    |    |
| 9  |   |   |   |   |   |                     |   |   |   |    |    |
| 10 |   |   |   |   |   |                     |   |   |   |    |    |

Рисунок 9.31 – Печатная форма

### Задание

1. Разработать макет печатной формы
2. Убедиться, что данный макет отображает все элементы, которые в него передаются.

### Контрольные вопросы

1. Что такое макет печатной формы?
2. Для чего нужен данный вид макета?
3. Какие области есть при создании макета?

## ЛАБОРАТОРНАЯ РАБОТА №3. РАБОТА С СИСТЕМОЙ КОМПОНОВКИ ДАННЫХ

### Понятие выборки

*Выборка* - это специализированный способ перебора чего-либо. В языке программирования 1С существует много видов выборки, но в этой главе мы рассмотрим два из них, это: *Выборка документов* и *Выборка справочников*.

### Выборка справочников

*Выборка справочников* - это объект, который представляет собой специализированный способ перебора элементов справочника. *Выборка* является динамическим объектом, то есть она не получает все элементы справочника сразу, а получает их порциями, что позволяет достаточно быстро обходить справочники, состоящие из большого количества элементов.

Данный объект возвращается двумя методами менеджера справочника, это: *Выбрать* и *Выбрать иерархически*.

Рассмотрим оба этих метода. И посредством работы с данными методами перейдем к выборке ***Выбрать***. Метод *Выбрать* объекта «Справочник менеджера» формирует некоторую выборку данных по заданным условиям. Этот метод является функцией и возвращает объект *Выборка справочника*.

Рассмотрим синтаксис данного метода.

**Выбрать(<Родитель>, <Владелец>, <Отбор>, <Порядок>)**

Все параметры данного метода являются необязательными.

«*Родитель*» - применим для иерархических справочников. Если он указан, то будут выбираться только те элементы справочника, у которых в свойстве *Родитель* стоит данный элемент. Если параметр не заполнен, то выберутся все элементы справочника. Чтобы выбрать элементы только верхнего уровня, необходимо указать в качестве *Родителя* пустую ссылку. «*Владелец*» - необходим если мы работаем с подчиненным справочником. Когда мы его укажем, то будет

осуществлена выборка всех элементов, которые подчинены данному владельцу.

«Отбор» -. Вот на этом параметре мы остановимся поподробнее. Данный параметр представляет собой структуру, в которой мы можем задать критерий, по которому будет произведена выборка данных. Ключом элемента структуры будет название поля, по которому будет действовать отбор, а значение структуры – непосредственно то значение, по которому должен будет производиться отбор. Структура должна содержать только один элемент! Т.е. отбор можно осуществить только по одному реквизиту.

Обращаю Ваше внимание, что в качестве полей для отбора могут быть заданы только те поля, у которых признак индексирования установлен либо в «Индексировать», либо в «Индексировать с дополнительным упорядочиванием». Рассмотрим, где этот признак находится. Для этого зайдите в справочник *Автомобили* и откройте свойства элемента *Коробка передач*. Обратите внимание на параметр *Индексирование*.

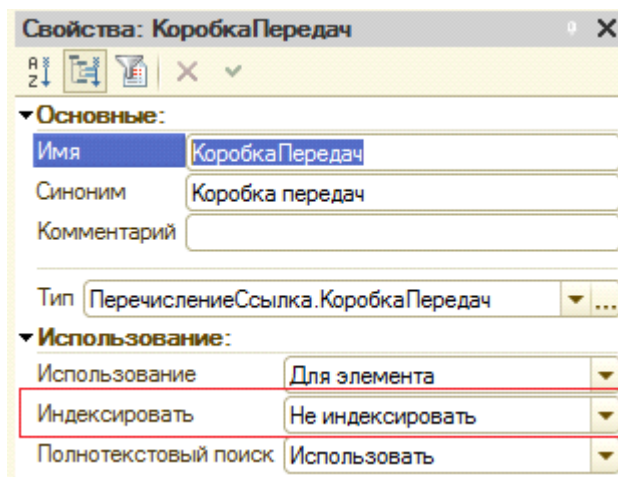


Рисунок 10 – Свойства элемента Справочника

У данного поля этот параметр установлен в значение «Не индексировать». Поменяйте его на значение «Индексировать».

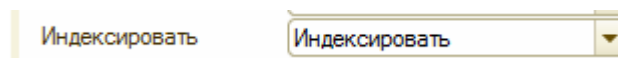


Рисунок 10.1 – Свойства элемента «Индексировать».

Все, теперь мы его сможем использовать в отборах для выборки. Существуют всего два поля, которые могут свободно быть использованы в отборе, это поля *Код* и *Наименование*.

Поэтому прежде, чем делать какую-нибудь выборку с отбором, обратите внимание на данный параметр для отбираемого поля. И только после принимайте решение об использовании отбора. Если данное поле не является индексируемым, а Вам все равно необходимо получить выборку с отбором по этому полю, то необходимо использовать язык запросов. О нем мы поговорим во второй части текущей главы.

И последний, тоже интересный параметр - это *Порядок*, данным параметром является строка с именем реквизита, по которому будет упорядочена выборка. В данный параметр свободно могут быть указаны поля *Наименование* и *Код*, а также поля примитивных типов (*число, строка, дата, булево*), для которых признак *Индексирование* установлен либо в «*Индексировать*», либо в «*Индексировать с дополнительным упорядочиванием*». Сортировка данных будет осуществлена по возрастанию, но можно провести и сортировку по убыванию, написав в конце строки слово «*Убыв*».

Например так:

### **"Код Убыв"**

Также можно и задать сортировку по возрастанию, написав в конце строки слово «*Возр*».

### **"Код Возр"**

Итак, все параметры мы изучили, теперь осталось применить это на практике. Для этого сначала сделаем простую выборку справочника *Склады*, а потом выборку с иерархией.

Результаты выборки будем выводить в таблицу значений.

На основе данных примеров Вы научитесь работать с выборками.

Создайте новую обработку, разместите на ее форме табличное поле, в которое добавьте одну колонку – *Склад* (тип *ссылка на справочник Склад*).

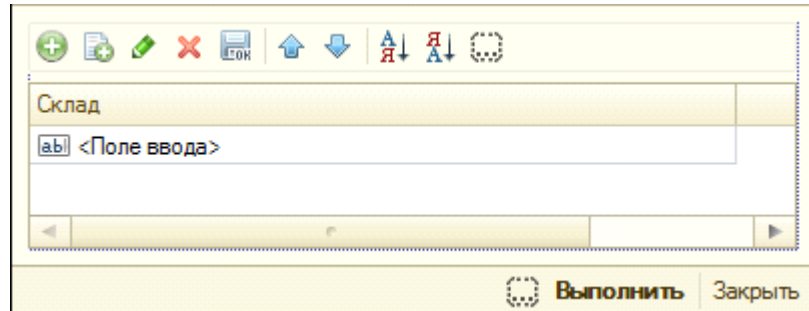


Рисунок 10.2 – Создание обработки.

В обработчике кнопки «*Выполнить*» напомним следующий код.

```

Процедура КнопкаВыполнитьНажатие (Кнопка)
    ВыборкаСкладов = Справочники.Склады.Выбрать ();
    Пока ВыборкаСкладов.Следующий () Цикл
        НоваяСтрока = Склады.Добавить ();
        НоваяСтрока.Склад = ВыборкаСкладов.Ссылка;
    КонечЦикла;
КонечПроцедуры

```

Рисунок 10.3 –Код обработки.

Разберем данный код. В первой строке мы получаем выборку элементов справочника *Склады*.

Метод, использующийся во второй строке, Вам незнаком. Это метод *Следующий*.

Данный метод получает следующий элемент выборки. Дело в том, что любая выборка представляет собой совокупность элементов, которые были выбраны.

Обращаться к этим элементам можно непосредственно через ту переменную, в которую данная выборка записана. В нашем случае это переменная *ВыборкаСкладов*. Но когда мы ее только получили с помощью метода *Выбрать*, она еще не позиционирована ни на каком элементе. Поэтому для того, чтобы пройти по всем элементам

выборки, нам понадобится метод *Следующий*, с помощью которого осуществляется переход на следующий элемент выборки. Данный метод является функцией и возвращает значение *Истина*, если элемент выбран, и *Ложь* - в том случае, когда достигнут конец выборки. Если нам необходим просто первый элемент выборки, достаточно написать вот так:

**ВыборкаСкладов = Справочники.Склады.Выбрать ();**  
**ВыборкаСкладов.Следующий ();**

После того как элемент выборки выбран, то можно обращаться к нему как к элементу справочника. Поэтому мы получаем доступ ко всем реквизитам справочника и можем их использовать в дальнейшей работе. Сама переменная *ВыборкаСкладов* имеет тип *СправочникВыборка.Склады*:  
**СправочникВыборка.Склады.**

Поэтому в данном примере мы получили ссылку на элемент справочника, обратившись к выборке.

Если бы нам надо было получить наименование элемента справочника или код, то мы тоже легко смогли бы это сделать:

**Наименование = ВыборкаСкладов.Наименование;**  
**Код = ВыборкаСкладов.Код;**

Если Вы уже самостоятельно прописывали вышеприведенный код, то заметили, что когда мы пишем точку после переменной *ВыборкаСкладов*, то возникает выпадающий список, в котором перечислены все реквизиты справочника.

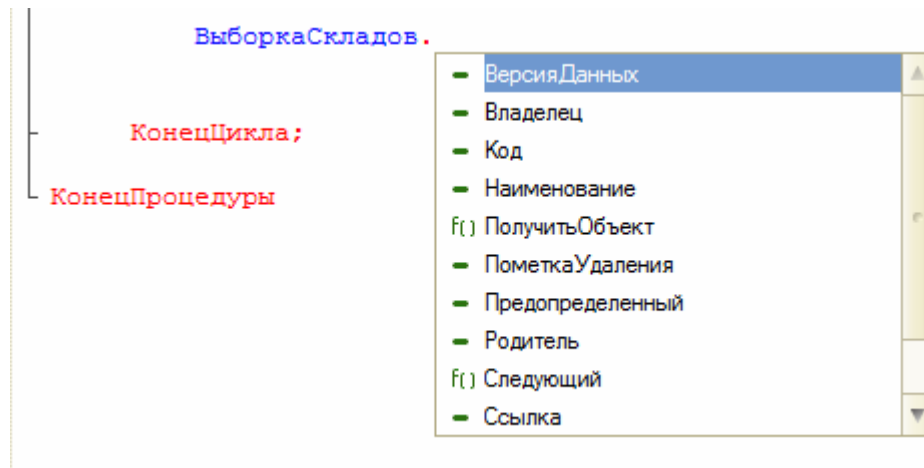


Рисунок 10.4 – Свойства реквизитов справочника.

У выборки справочника есть еще один метод - *ПолучитьОбъект*. С помощью данного метода мы получаем объект элемента справочника.

### **ВыборкаСкладов.ПолучитьОбъект();**

С этим методом Вам все должно быть понятно, и каких-либо дополнительных пояснений не нужно.

Теперь, чтобы разобрать более подробно метод *Выбрать*, доработайте Ваш справочник складов. Добавьте в него два реквизита, это реквизит *Склад автозапчастей* (тип *булево*, неиндексируемый) и реквизит *Время начала работы склада* (тип *дата*, состав даты - время, индексируемый).

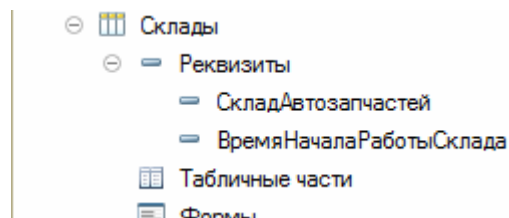


Рисунок 10.5 –Реквизиты склада.

И переделайте форму созданной обработки: добавьте в табличное поле новые колонки: *Время открытия*, и *Склад автозапчастей*. У колонки *Склад автозапчастей* сделаем элемент управления – флажок.

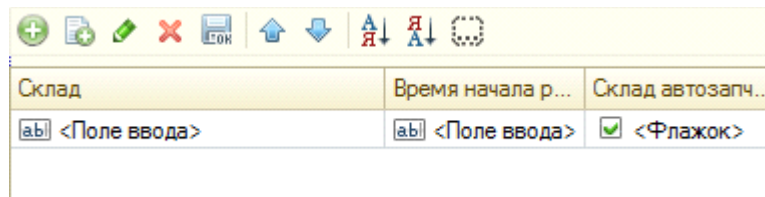


Рисунок 10.6 – Добавление в табличное поле новой колонки.

Допишите в конце цикла следующие строки:

```
НоваяСтрока.ВремяНачалаРаботы = ВыборкаСкладов.ВремяНачалаРаботыСклада;
НоваяСтрока.СкладАвтозапчастей = ВыборкаСкладов.СкладАвтозапчастей;
КонецЦикла;
```

Рисунок 10.7 – Код выборки

Создайте самостоятельно разветвленную структуру складов (с группами и элементами).

Сейчас мы научимся сначала выбирать элементы только верхнего уровня, а потом элементы одной конкретной группы.

Выберем элементы только первого уровня.

Для этого в первом параметре метода *Выбрать* напишем пустую ссылку на элемент справочника.

```
ВыборкаСкладов = Справочники.Склады.Выбрать(Справочники.Склады.ПустаяСсылка(), , );
Склады.Очистить();
Пока ВыборкаСкладов.Следующий() Цикл
```

Рисунок 10.8 – Код выборки

Сохраните обработку и запустите, как Вы увидели, выведены все элементы верхнего уровня, включая группы. К сожалению, мы не можем отобрать с помощью выборки только элементы справочника, поэтому если Вам не нужно появление групп в табличном поле, то можно дописать следующий код:

```
Пока ВыборкаСкладов.Следующий() Цикл
    Если ВыборкаСкладов.ЭтоГруппа тогда
        Продолжить;
    КонецЕсли;
    НоваяСтрока = Склады.Добавить();
    НоваяСтрока.Склад = ВыборкаСкладов.Ссылка;
```

Рисунок 10.9 – Код выборки

Сохраните обработку и перезапустите.



И, как Вы должны увидеть, вышли только элементы справочника. Теперь научимся делать выборки для конкретного родителя. Для этого разместим на форме поле ввода для справочника *Склады* и поставим у него признак «*Выбор групп и элементов*».

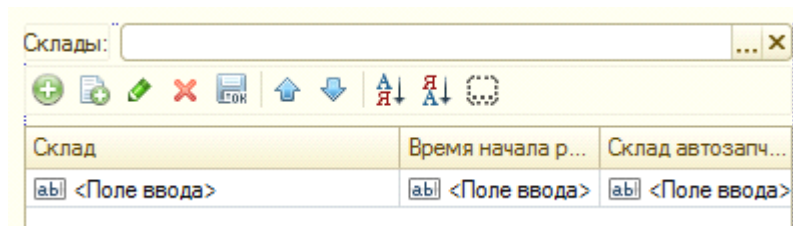


Рисунок 10.10 – поле ввода для справочника *Склады*

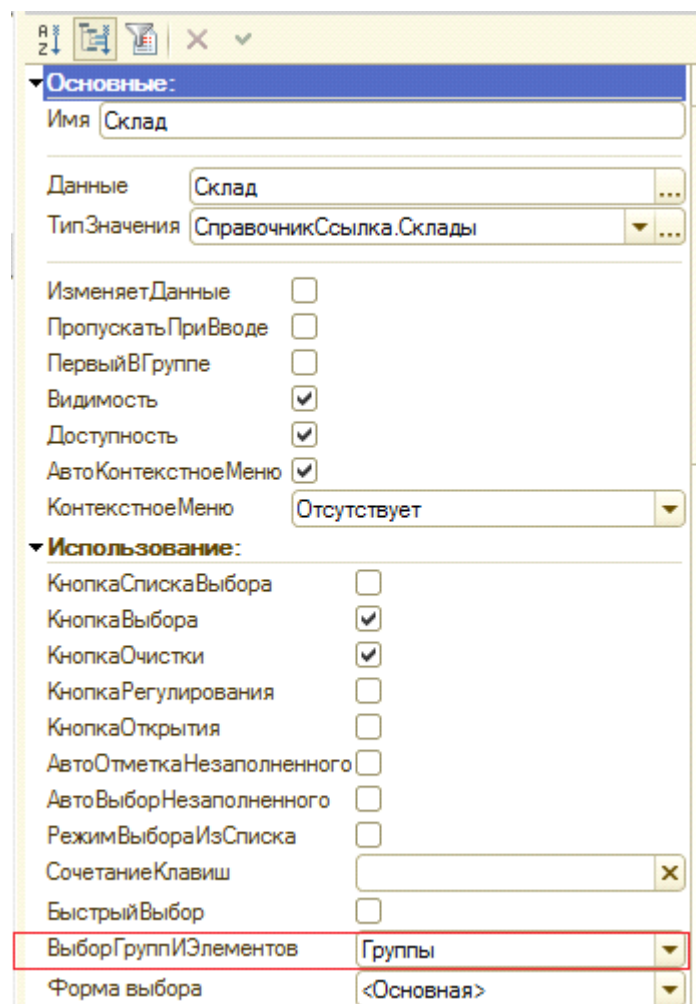


Рисунок 10.11 – признак «*Выбор групп и элементов*»

Доработайте предыдущий код, следующий образом:

```

ВыборкаСкладов = Справочники.Склады.Выбрать(Склад, , );
Склады.Очистить ( );
Пока ВыборкаСкладов.Следующий ( ) Цикл

```

Рисунок 10.12 – Код обработки

Посмотрите, как будет работать в этот раз Ваша обработка.

Вы должны были заметить, что если Вы выбираете группу, у которой внутри есть еще одна группа, то выводиться элементы, которые входят в группу выбора. Но элементы, входящие в подчиненную группу, не вышли.

Для того чтобы получить все элементы, включая элементы подчиненных справочников необходимо использовать метод менеджера справочника *ВыбратьИерархически*. Синтаксис данного метода точно такой же, как и у метода *Выбрать*, поэтому мы не будем его разбирать, а просто продемонстрируем, как он работает. Замените метод *Выбрать* на метод *ВыбратьИерархически*.

```

ВыборкаСкладов = Справочники.Склады.ВыбратьИерархически(Склад, , );
Склады.Очистить ( );
Пока ВыборкаСкладов.Следующий ( ) Цикл

```

Рисунок 10.13 – Код обработки

Посмотрите, как он работает.

Как видите, данный метод очень удобно использовать, когда работаем с иерархическими справочниками, и необходимо получить все элементы данного справочника.

Обращаю Ваше внимание, что сначала будут выбраны группы самого верхнего уровня, потом группы следующего уровня и так далее. И только после этого будут выбраны элементы, начиная с самого нижнего уровня и заканчивая нулевым уровнем. Чтобы убедиться в этом, прокомментируйте код, в котором мы отказываемся

выводить группы, и посмотрим, что получится, когда в поле ввода нет элемента справочника. Обратите внимание на порядок, в котором вышли строки в таблице значений.

В дальнейшем применительно к справочнику *Склады*, будем работать с методом *ВыбратьИерархически*.

Теперь научимся работать с отбором.

Создадим отбор складов по *Времени начала работы склада*. Мы сделали данный реквизит справочника индексируемым, поэтому он должен отбираться как надо.

Поскольку нам необходимо сделать отбор по времени, добавьте на форму поле ввода *Время* (Тип *Дата*, состав даты - *Время*).

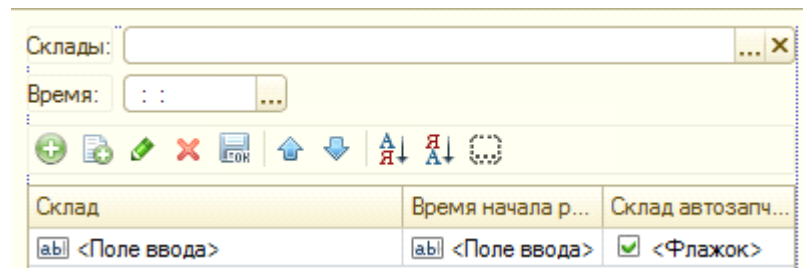


Рисунок 10.14 – Добавление поле ввода *Время*

И доработайте Ваш код следующим образом:

```
Если Время = '00010101000000' тогда
    ВыборкаСкладов = Справочники.Склады.ВыбратьИерархически(Склад, , );
иначе
    Отбор = Новый Структура;
    Отбор.Вставить ("ВремяНачалаРаботыСклада", Время) ;

    ВыборкаСкладов = Справочники.Склады.ВыбратьИерархически(Склад, , Отбор) ;
КонецЕсли;
Склады.Очистить () ;
```

Рисунок 10.15 – Код обработки

Обращаю Ваше внимание, что выборка осуществляется для всех объектов, включая группы, и Вы знаете, что сначала выводится группа, а потом элементы, которые входят в группу. Но отбор тоже

применяется для всех элементов, включая группы, поэтому если мы будем на верхнем уровне и зададим отбор *10* часов, то ничего не выйдет. Почему? Потому что у нас не отберутся группы первого уровня, для них не установлен данный реквизит, следовательно, не выведутся и элементы, которые входят в них. Поэтому список будет пустой.

Теперь попробуйте сделать отбор для неиндексируемого поля, и посмотрите, что получится.

Измените код:

```
Отбор = Новый Структура;
Отбор.Вставить ("СкладАвтозапчастей", Истина);
ВыборкаСкладов = Справочники.Склады.ВыбратьИерархически (Склад, , Отбор);
Склады.Очистить ();
Пока ВыборкаСкладов.Следующий () Цикл
```

Рисунок 10.16 – Код обработки

Запустите обработку и посмотрите, что должно возникнуть. Как видите, возникла ошибка.

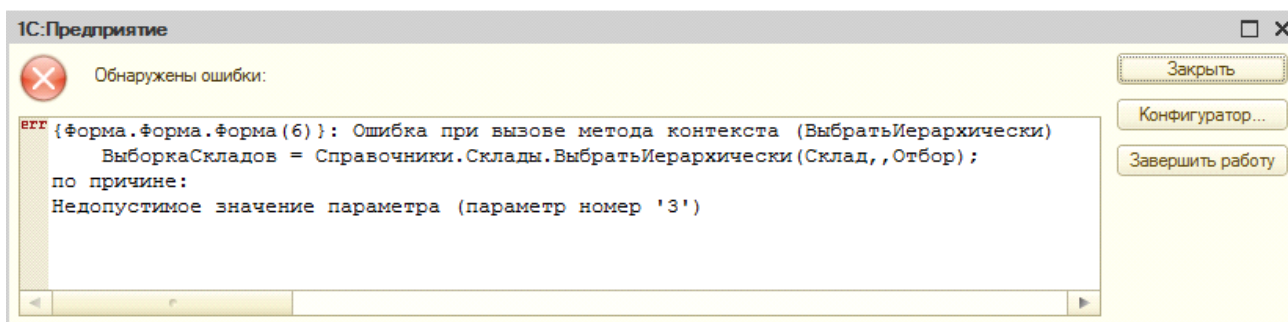


Рисунок 10.17 – Ошибка выполнения обработки

Таким образом, еще раз повторяюсь, отбор работает только с индексируемыми полями.

Теперь рассмотрим последний параметр выборки - это *Порядок*. Выведем все наши элементы, упорядочив их по *Времени начала работы склада*.

Измените код следующим образом:

```

ВыборкаСкладов = Справочники.Склады.ВыбратьИерархически(Склад,,, "ВремяНачалаРаботыСклада");
Склады.Очистить();
Пока ВыборкаСкладов.Следующий() Цикл

```

Рисунок 10.18 – Элементы, упорядоченные по *Времени начала работы склада*

Поскольку параметр *Порядок*, так же, как и параметр *Отбор*, действует на все элементы справочника, включая группы, то для наглядности измените время нескольких элементов в одной группе на разное и выведете элементы только этой группы.

Если Вам необходимо упорядочивание по убыванию, то изменим написание последнего параметра.

```

ВыборкаСкладов = Справочники.Склады.ВыбратьИерархически(Склад,,, "ВремяНачалаРаботыСклада УБЫВ");

```

Рисунок 10.19 – Упорядочивание по убыванию

Сохраните обработку, перезапустите, и Вы увидите, что элементы упорядочились по убыванию.

Итак, мы рассмотрели все параметры кроме второго – *Владелец*. Допишите Вашу обработку – создайте новое табличное поле *Помещение*, с одной колонкой *Помещение*.

The screenshot shows a software window with a yellow background. At the top, there are two input fields: "Склады:" and "Время:". Below them is a toolbar with various icons including a plus sign, a minus sign, a green checkmark, a red X, a printer icon, and arrows. The main area contains a table with three columns: "Склад", "Время начала р...", and "Склад автозап...". The first row of the table has input fields for "ab" and "ab", and a checkbox labeled "<Флажок>". Below the table is a section labeled "Помещение" with an input field for "ab". At the bottom right, there are two buttons: "Выполнить" and "Заккрыть".

Рисунок 10.20 – создание нового табличного поля *Помещения*

И в событии *ПриАктивизацииСтроки* табличного поля *Склады* напишите следующий код:

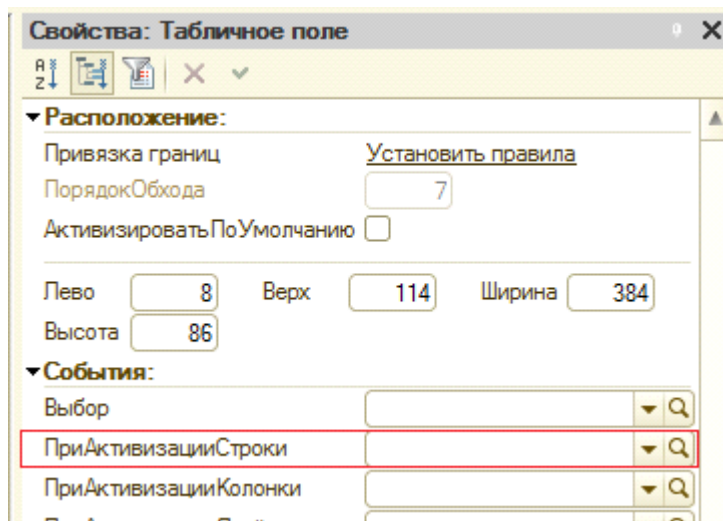


Рисунок 10.21 - Событие *ПриАктивизацииСтроки*

```

□ Процедура СкладыПриАктивизацииСтроки (Элемент)
    СкладТекущий = ЭлементыФормы.Склады.ТекущиеДанные.Склад;
    ВыборкаПомещений = Справочники.Помещения.Выбрать (, СкладТекущий);
    Помещения.Очистить ();
    Пока ВыборкаПомещений.Следующий () цикл
        НоваяСтрока = Помещения.Добавить ();
        НоваяСтрока.Помещение = ВыборкаПомещений.Ссылка;
    КонечЦикла;
□ КонечПроцедуры

```

Рисунок 10.22 – Код события *ПриАктивизацииСтроки*

В данном коде при активизации строки мы получаем текущий склад, и вставляем его в качестве второго параметра в метод *Выбрать*.

Посмотрите, что у Вас получилось.

На этом мы закончим изучать выборку справочников и перейдем к изучению выборок документов.

### Задание

1. Создать выборку справочника по примеру лабораторной работы проделав все шаги
2. Каждый шаг зафиксировать и прокомментировать в отчете

3. Выполнить индивидуальное задание на создание выборки справочников

### **Контрольные вопросы**

1. Что такое выборка?
2. Для чего используется выборка?
3. Какими параметрами обладает выборка?
4. Какие бывают выборки?

## ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №9. РАБОТА С ПЕРЕЧИСЛЕНИЯМИ И СПРАВОЧНИКАМИ НА ОСНОВЕ ОБЪЕКТНОЙ МОДЕЛИ

### Перечисления

Перечисление можно рассматривать, как некий линейный и не изменяемый список. Это может быть список видов номенклатуры, операций того или иного документа, времен года, сторон света и т.п.

Перечисления создаются в соответствующей ветке дерева объектов конфигурации. После создания на закладке «Данные» определяется нужный перечень значений. *В режиме исполнения это список не подлежит ни исправлению, ни переопределению состава.* Пользователь может либо выбрать значение из этого списка, либо сбросить выбор.

Например, перечисление ВидыТоваров ('Товар', 'Услуга', 'Материал'). То есть перечисление ВидыТоваров может принимать значение 'Товар', 'Услуга', 'Материал'.

Создание перечисление в окне редактирования объектов показано на Рисунок 11.

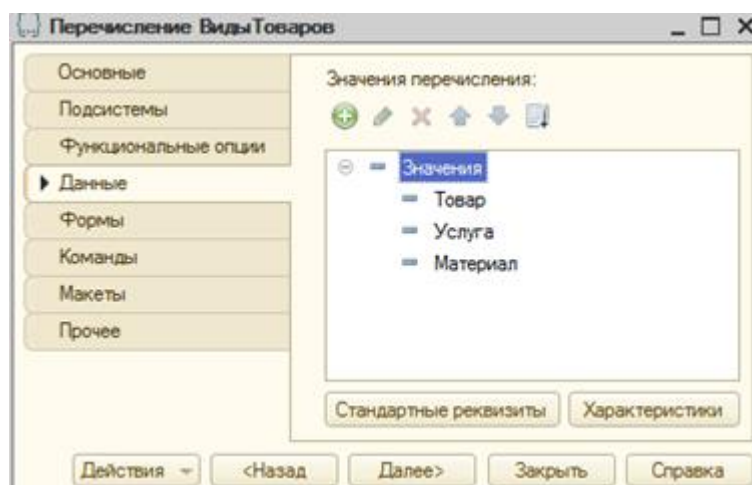


Рисунок 11 - Создание перечислений

В программном модуле для доступа к перечислениям используются на стороне клиента специальный тип данных **ПеречисленияСсылка**. На стороне сервера используется специальный объект **Перечисления**.



### Работа со справочниками.

Справочник – агрегатный объект, хранящий однотипные данные

При создании справочника, как и при создании других объектов дерева конфигурации открывается окно редактирования объектов конфигурации, в котором можно задать основные свойства объекта.

Рисунок 11.1 - Окно редактирования объектов конфигурации при создании и корректировки справочников

Справочники, как и другие объекты конфигурации, должны быть отнесены к некоторой подсистеме, иначе они не будут отображаться в интерфейсе.

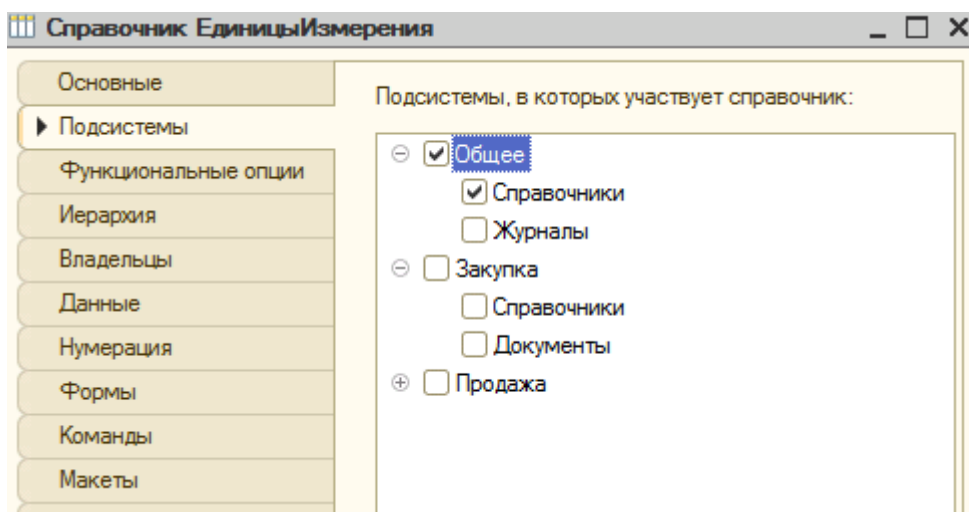


Рисунок 11.2 - Отнесение справочник к подсистеме

В справочниках 1с 8.X существуют predetermined элементы.

Предetermined элементы – элементы, которые всегда будут присутствовать в справочнике, они задаются в режиме конфигурирования в окне конструктора справочника – закладка "Прочие", команда "Предetermined"

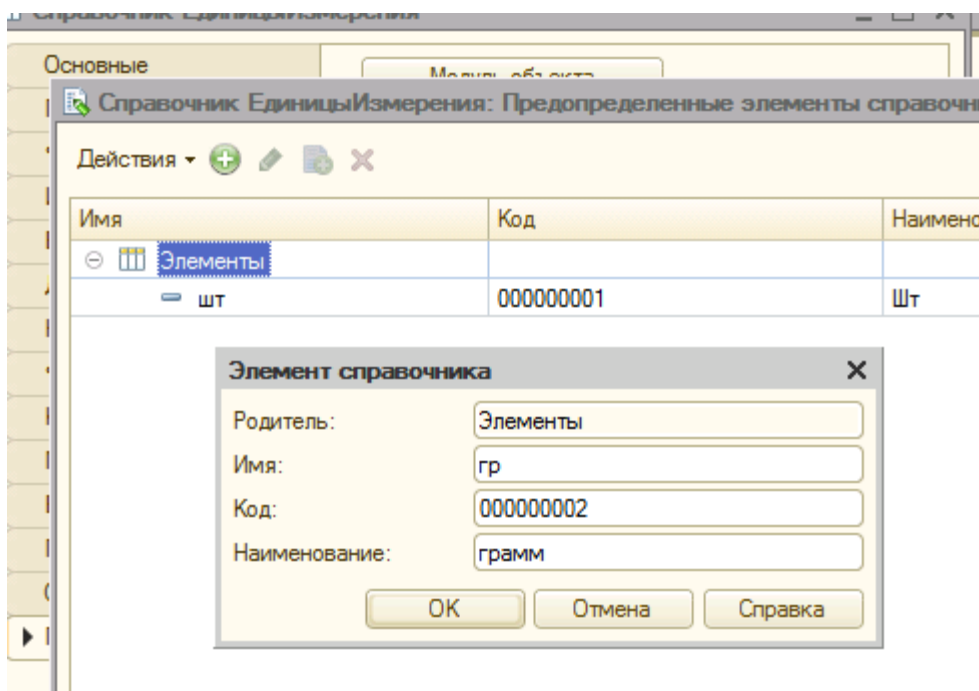


Рисунок 11.3 – Задание predetermined элементов

Реквизиты справочников – это данные, которые характеризуют отдельные элементы справочников. Для отдельного реквизита должны быть определено его *Имя* и тип данных. По умолчанию справочники содержат реквизиты *Код* и *Наименование*. Реквизиты справочников определяются на закладке "Данные".

Рисунок 11.4 – Создание реквизитов справочник.

### Формы справочников.

Со справочником может быть связано несколько типов форм.

Форма элемента справочника – используется для создания и редактирования элемента справочника.

Форма группы справочника – используется для создания и редактирования группы справочника.

Форма списка справочника – используется для работы со списком элементов справочника. Создаётся по умолчанию.

Форма выбора – используется для выбора элементов справочника.

Форма выбора групп – используется для выбора групп справочника.

Справочник может не иметь всех типов форм. По умолчанию создаётся форма списка справочника и форма элемента.

Формы справочников, создаются, как и другие составляющие объектов конфигурации через окно редактирования объектов

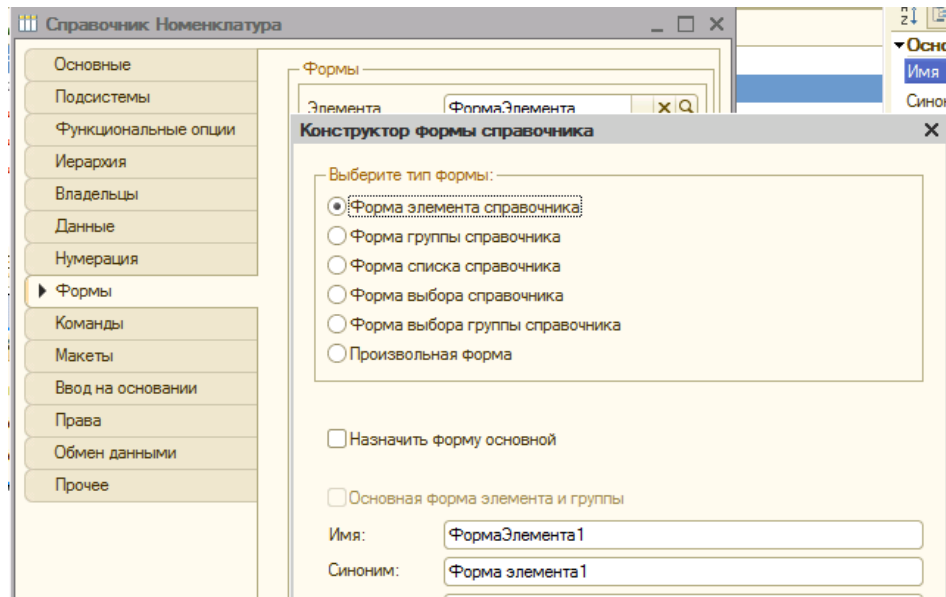


Рисунок 11.5 – Окно редактирования объектов

Для формы можно задать свойства в окне свойств объектов.

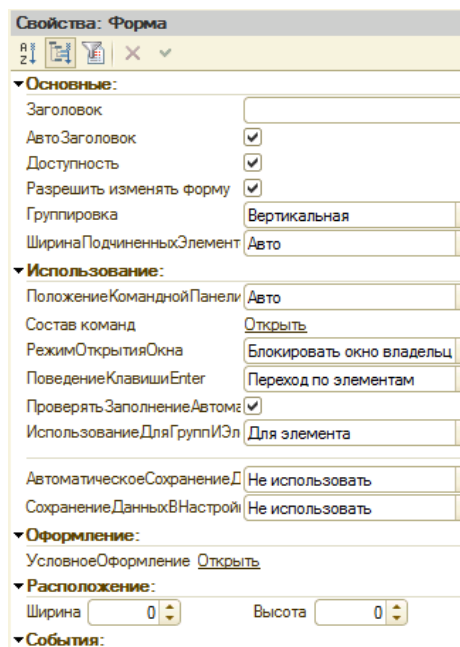


Рисунок 11.6 – Свойства Формы

### Работа со справочниками на основе объектной модели.

Объектная модель подразумевает работу со справочниками как с объектами среды исполнения, обладающими некоторыми свойствами и операциями.

Выборка элементов справочника.

Выбрать элементы в отдельную переменную(объект). Для этого используется метод **Выбрать()**.

// На чтение

**Выборка = Справочники.Номенклатура.Выбрать ();**

**Пока Выборка.Следующий() Цикл**

**Переменная = Выборка.Наименование;**

**// обработка полученного значения**

**КонецЦикла;**

Используемые команды, свойства и объекты.

**Справочники** – Коллекция объектов метаданных, которые описывают все справочники, определенные в конфигурации.

Объект **Справочники** доступен только в методах, исполняемых на сервере.

**Справочники.Номенклатура** – обращение к справочнику – номенклатура.**Выбрать()** – метод справочника, обеспечивает получение набора для перебора данных. Метод существует у многих агрегатных объектов – коллекций. Возвращаемое значение набор (**Выборка**), позиционированный на одном из элементов (первом).

Методы, свойства выборки.

**Выборка.Следующий()** метод получения следующего объекта из выборки

**Выборка.ПолучитьОбъект()** – Метод агрегатного объекта, для получения объектной ссылки на текущий объект.

**Об= Выборка.ПолучитьОбъект();**

**Об.Свойство =НовоеЗначение;**

**Выборка.ПолучитьОбъект().Свойство=НовоеЗначение;**

Что-то можно сделать с объектом, изменить его свойства, только получив его как объект, с помощью метода [ПолучитьОбъект](#). (предварительно, естественно, надо сделать выборку, из справочника ничего сделать нельзя)

Справочники.Номенклатура.ДляПереноса – Это имя группы, получается группа.

Фрагмент кода выборки и переноса всех элементов в предопределенную группу

[// на запись](#)

[ПредопределеннаяГруппа=Справочники.Номенклатура.КудаПереместить;](#)

[Выборка = Справочники.Номенклатура.Выбрать\(\);](#)

[Пока Выборка.Следующий\(\) Цикл](#)

[Если Выборка.Ссылка.Родитель = ПредопределеннаяГруппа](#)

[Тогда](#)

[Продолжить;](#)

[КонецЕсли;](#)

[ПолученныйОбъект = Выборка.ПолучитьОбъект\(\);](#)

[ПолученныйОбъект.Родитель = ПредопределеннаяГруппа;](#)

[ПолученныйОбъект.Записать\(\);](#)

[КонецЦикла;](#)

Предопределённая группа, это группа, которая создана в конфигураторе, поэтому её видно из программного модуля.

Другие группы справочника не видно в программном модуле. Для того, чтобы их увидеть, лучше всего, их туда ввести (выбрать нужную группу) из элемента управления.

А для этого нужно добавить в форму обработки соответствующий реквизит. Тип реквизит должен быть - СправочникСсылка.Номенклатура. Такой тип (ссылка) позволяет обращаться к элементам соответствующих объектов (справочников, документов, и т.д.)

Установить в форме элемент управления, позволяющий обращаться к данному реквизиту. Для этого можно просто

перетащить реквизит из области реквизитов в область элементов управления.

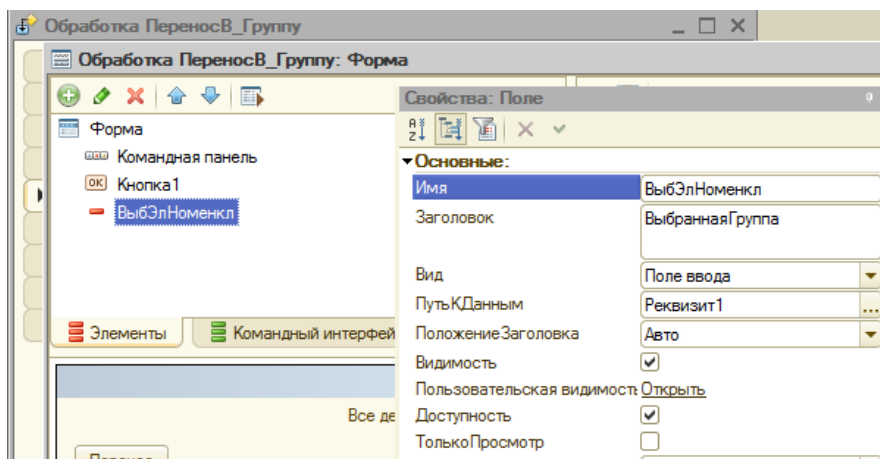


Рисунок 11.7 – Установка элемента управления для выбора группы в справочнике.

В рабочей среде данный элемент позволяет открыть форму списка справочника для выбора значений. Выбор элемента производится кнопкой "ВЫБРАТЬ".

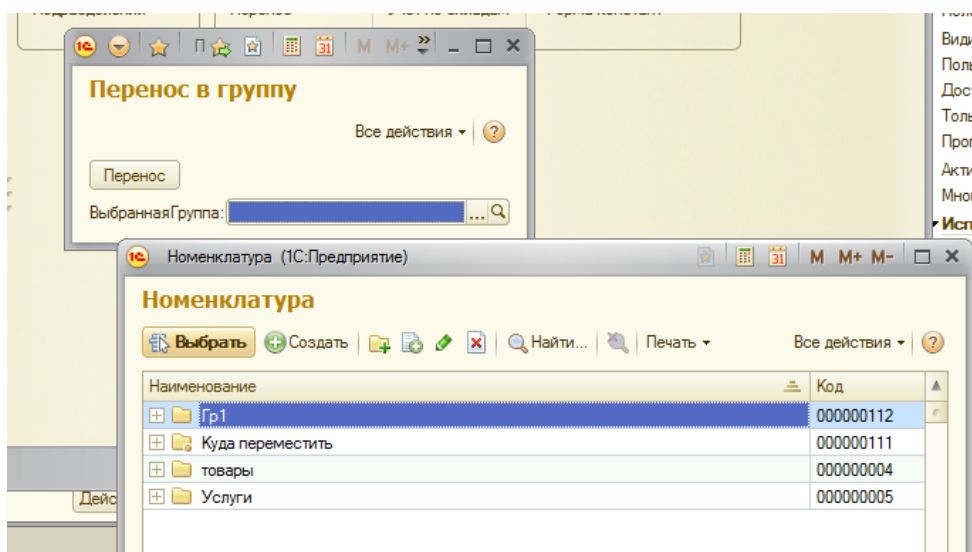


Рисунок 11.8 – Выбор элемента справочника или группы

Для обращения к выбранному элементу справочника в программном коде необходимо использовать идентификатор соответствующего реквизита формы.

*(Можно и через управляющий элемент, но это будет концептуально неправильно).*

ГруппаКуда = Реквизит1;

Процедура переноса в выбранную группу будет выглядеть следующим образом

&НаСервере

Процедура Пере()

ГруппаКуда=Реквизит1;

Выборка = Справочники.Номенклатура.Выбрать();

Пока Выборка.Следующий() Цикл

Если Выборка.Ссылка.Родитель = ГруппаКуда Тогда

Продолжить;

КонецЕсли;

// Проверка не переноса элементов отнесённым к группам

Если (ЗначениеЗаполнено (Выборка.Ссылка.Родитель)) Тогда

Продолжить;

КонецЕсли;

// Проверка не переноса групп

Если Выборка.Ссылка.ЭтоГруппа Тогда

Продолжить;

КонецЕсли;

ПолученныйОбъект = Выборка.ПолучитьОбъект();

ПолученныйОбъект.Родитель = ГруппаКуда;

ПолученныйОбъект.Записать();

КонецЦикла;

КонецПроцедуры

**ЗначениеЗаполнено (НННН)** – метод возвращает, если значение свойства задано

**Выборка.Ссылка.ЭтоГруппа** – свойство элемента справочника, возвращает Истина, если текущий элемент группа.



*Задайте возможность создание элементов справочника подразделения непосредственно с панели действий с помощью настройки соответствующей подсистемы (кнопка командный интерфейс, основной страницы конструктора объекта).*

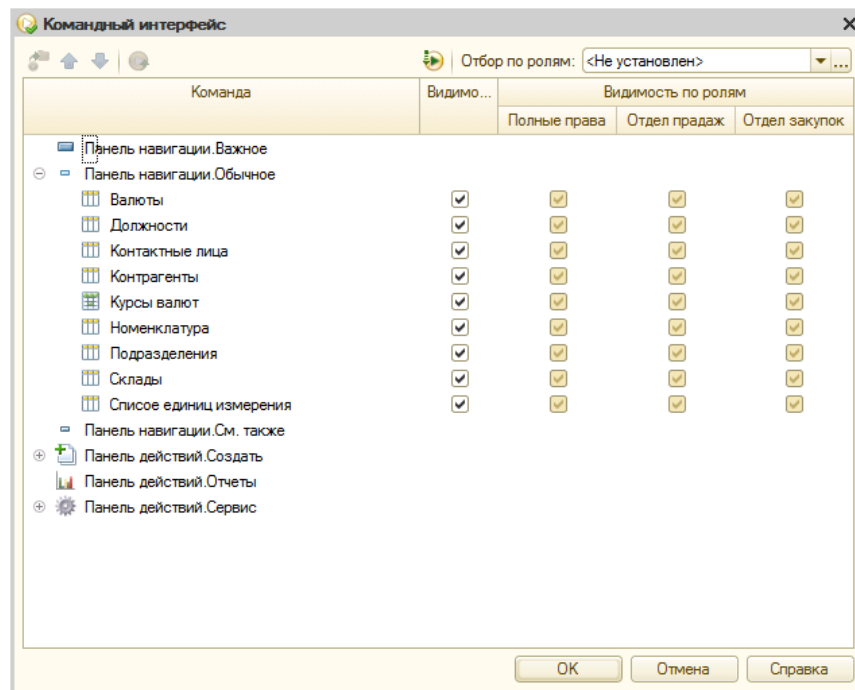


Рисунок 11.9 – Командный интерфейс

### Задание

1. Создать справочник «ЕдиницыИзмерения». Подсистема общие/справочники. Задайте у справочника несколько предопределённых единиц измерения. (штука, килограмм, литр)
2. Создать справочник "Подразделения". справочник должен быть отнесён к подсистеме "Общие", "Справочники".
3. Задайте возможность создание элементов справочника подразделения непосредственно с панели действий с помощью настройки соответствующей подсистемы (кнопка командный интерфейс, основной страницы конструктора объекта)
4. Создать справочник "Подразделения". (Подсистемы - "Общие", "Общие/Справочники". Задать возможность создания элементов справочника с панели действий.

5. Создать перечисление Виды товаров. Элементы перечисления – Товар, услуга, материал.
6. Создать справочник номенклатура. Реквизиты справочника. ЦенаПокупки (число (10,2), ЦенаПродажи – число (10,2), ВидНоменклатуры –Перечисление (ВидыТоваров), Основная единица измерения – (справочник – единицы измерения).
7. Создать справочник "Серии", подчинённый справочнику – номенклатура. (Подсистемы - Общие, Общие/Справочники)
8. Создать обработки для вывода содержимого справочника на основе объектной модели.
9. Вывод справочника подразделения
10. Вывод справочника номенклатура
11. Создать обработку для изменения значения реквизитов справочника
12. цены покупки элементов справочника номенклатура заданной группы на заданную величину.
13. Для нахождения средней цены продажи для элементов справочника номенклатуры выбранной группы.
14. Для увеличения покупной цены элементов справочников выбранной не отнесённых ни к какой группе на заданную величину
15. Для вычисления величины цены накрутки для элементов, не отнесённых ни к какой группе и увеличения на неё цены поставки в заданной группе.

### **Контрольные вопросы**

1. Что такое справочник.
2. Какие параметры справочника задаются при создании. Какие иерархии могут быть в справочниках. Что такое иерархия в справочниках.
3. Что такое предопределённые элементы справочника? Как они создаются. Можно ли в режиме предприятия удалить предопределённый элемент.
4. Каким образом задать возможность создания элементов справочников непосредственно с панели действий (где находится панель действий)

5. Что такое перечисление. Можно ли изменить значение перечисление из рабочего режима 1сПредприятия.
6. Какие формы у справочника есть как создаётся форма справочника. Задайте форму справочника по заданию преподавателя, откройте её в режиме предприятия.
7. Что представляет из себя объектная модель справочника. Каким образом можно перебрать содержимое справочника.

## ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №10. РАБОТА С ДАННЫМИ НА ОСНОВЕ ТАБЛИЧНОЙ МОДЕЛИ. ЗАПРОСЫ И ОБРАБОТКИ В СРЕДЕ 1С ПРЕДПРИЯТИЕ

### Запросы и обработки

Табличная модель подразумевает работу с данными на основе запросов.

**Запрос** – агрегатный объект, используемый для выборки данных аналогичный SQL в реляционных базах данных.

Для задания содержания запроса используется специальный язык аналогичный SQL

Пример, содержания запроса.

"ВЫБРАТЬ

```
|      Номенклатура.Наименование,
|      Номенклатура.ЦенаПродажи,
|      Номенклатура.ВидНоменклатуры,
|      Номенклатура.ОсновнаяЕдиницаИзмерения
|ИЗ
|      Справочник.Номенклатура КАК Номенклатура";
```

"ВЫБРАТЬ

```
|      Номенклатура.ВидНоменклатуры,
|      КОЛИЧЕСТВО(РАЗЛИЧНЫЕ Номенклатура.Наименование) КАК
|Наименование,
|      МИНИМУМ(Номенклатура.ЦенаПокупки) КАК ЦенаПокупки
|ИЗ
|      Справочник.Номенклатура КАК Номенклатура
|СГРУППИРОВАТЬ ПО
|      Номенклатура.ВидНоменклатуры"
```

Обычно работа с запросами включает следующие этапы.

- Создание запроса **Запрос1 = Новый Запрос;**
- Задание содержания запроса **Запрос1.Текст =**

**"ВЫБРАТЬ**

- Выполнение запроса **Запрос.Выполнить()**;
- Обработка результатов запроса

Создание запросов выполняется как создание объектов в среде 1с, с помощью конструктора Новый.

**Запрос1 = Новый Запрос;**

Задание содержания запроса – задание его свойства "Текст".

**Запрос1.Текст =**

**"ВЫБРАТЬ**

**|       Номенклатура.ВидНоменклатуры,**  
**|       КОЛИЧЕСТВО(РАЗЛИЧНЫЕ**  
**Номенклатура.Наименование) КАК Наименование,**  
**|       МИНИМУМ(Номенклатура.ЦенаПокупки)   КАК**  
**ЦенаПокупки**  
**|ИЗ**  
**|       Справочник.Номенклатура КАК Номенклатура**  
**|ГДЕ**  
**|       Номенклатура.ЭтоГруппа = ЛОЖЬ**  
**|**  
**|СГРУППИРОВАТЬ ПО**  
**|       Номенклатура.ВидНоменклатуры";**

**Результат = Запрос.Выполнить();**

**ВыборкаДетальныеЗаписи = Результат.Выбрать();**

Для создания запросов в тексте модуля можно воспользоваться конструктором запросов.

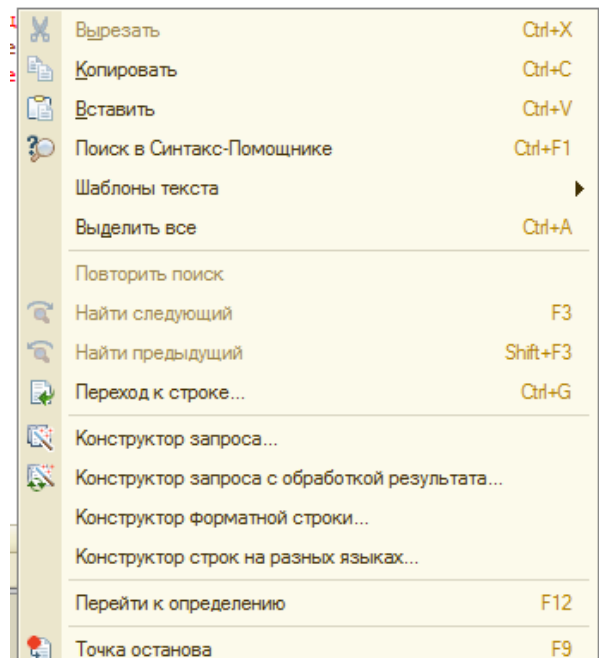


Рисунок 12 – Вызов конструктора запроса

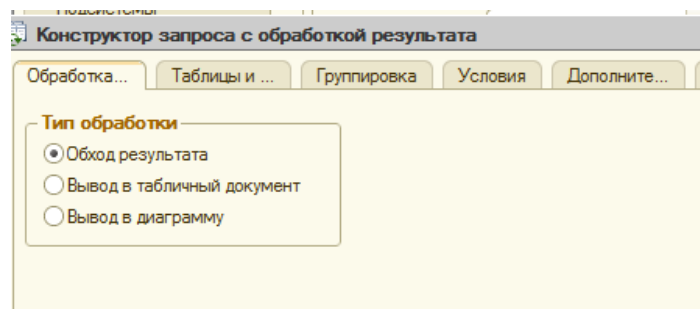


Рисунок 12.1 –Вызов конструктора запросов с обработкой результатов

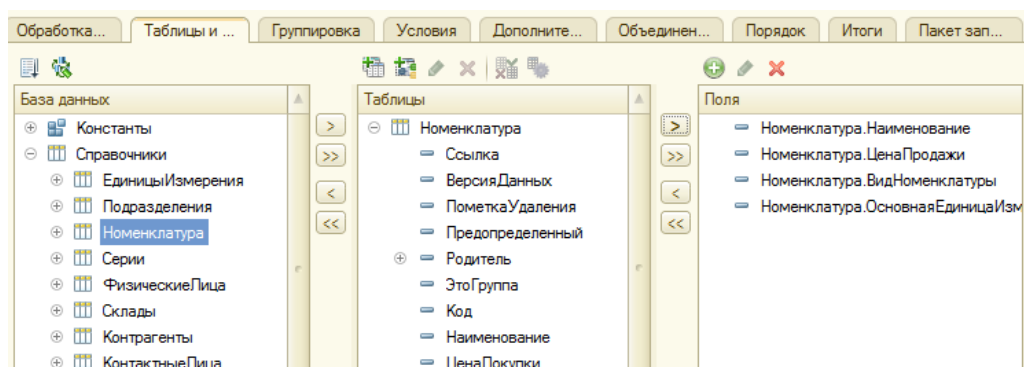


Рисунок 12.2 – Формирование текста запроса в конструкторе запросов

Кнопка "запрос" позволяет показать результаты.

В результате работы конструктора будет создан такой текст запроса

&НаСервере

Процедура ВыполнитьЗапросСервер()

//{{КОНСТРУКТОР\_ЗАПРОСА\_С\_ОБРАБОТКОЙ\_РЕЗУЛЬТАТА

// Данный фрагмент построен конструктором.

// При повторном использовании конструктора, внесенные вручную изменения будут утеряны!!!

Запрос = Новый Запрос;

Запрос.Текст =

"ВЫБРАТЬ

|       Номенклатура.Наименование,

|       Номенклатура.ЦенаПродажи,

|       Номенклатура.ВидНоменклатуры,

|       Номенклатура.ОсновнаяЕдиницаИзмерения

|ИЗ

|       Справочник.Номенклатура КАК Номенклатура";

Результат = Запрос.Выполнить();

ВыборкаДетальныеЗаписи = Результат.Выбрать();

Пока ВыборкаДетальныеЗаписи.Следующий() Цикл

    // Вставить обработку выборки

[ИмяПоля]

ВыборкаДетальныеЗаписи

    КонецЦикла;

//}}КОНСТРУКТОР\_ЗАПРОСА\_С\_ОБРАБОТКОЙ\_РЕЗУЛЬТАТА

### **Вывод на печать. Работа с табличными документами.**

Табличный документ — агрегатный объект. Таблица типа Excel, используется для вывода данных в качестве печатных форм, для ввода данных, могут быть и другие применения.

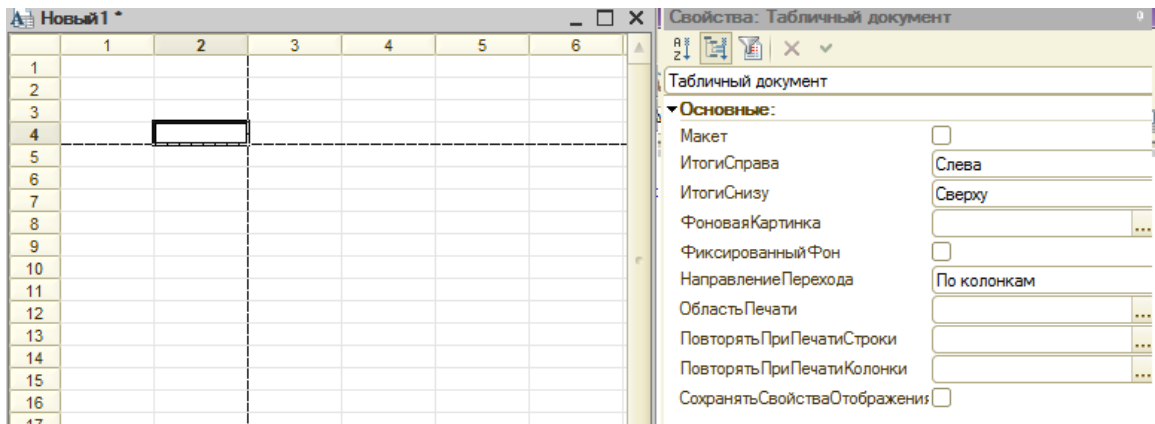


Рисунок 12.3 – Табличный документ

В программных модулях табличный документ создаётся с помощью универсального конструктора объектов **Новый ТабДок = Новый ТабличныйДокумент**;

При создании печатных форм табличный документ используется в виде особого типа табличных документов – макета. Для того что бы табличный документ стал макетом в нём должно быть задано специальное свойство.

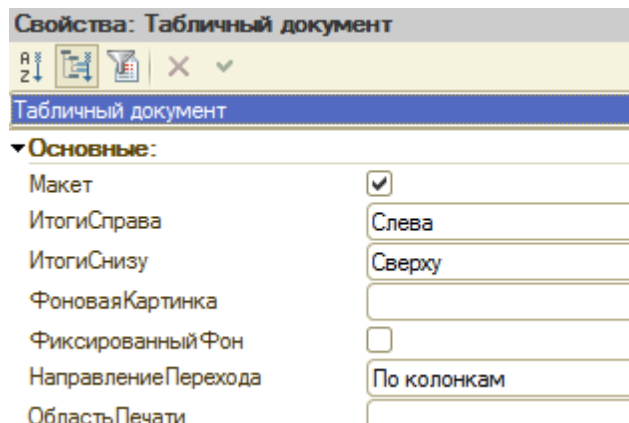


Рисунок 12.4 – Задание свойства «макет» для табличного документа

Макет отличается тем, что в нём могут быть заданы именованные области для вывода, к которым можно обращаться при формировании печатной формы и помещать их в формируемых документ как одно целое.



| Заголовок  | 1  | 2              | 3     | 4             | 5                 |
|------------|----|----------------|-------|---------------|-------------------|
| ШапкаТабл  | 4  | Наименование   | Код   | ЦенаПродажи   | ВидНоменклатуры   |
| Детали     | 5  | <Наименование> | <Код> | <ЦенаПродажи> | <ВидНоменклатуры> |
| ПодвалТабл | 6  |                |       |               |                   |
| Подвал     | 7  |                |       |               |                   |
|            | 8  |                |       |               |                   |
|            | 9  |                |       |               |                   |
|            | 10 |                |       |               |                   |

Рисунок 12.5 – Макет с именованными областями

Именованные области могут быть горизонтальными или вертикальными.

Создаются именованные области через команду основного меню режима configurator "Таблица/Имена/НазначитьИмя". При этом строки или столбцы, включаемые в область, должны быть выделены (Рисунок 12.6)

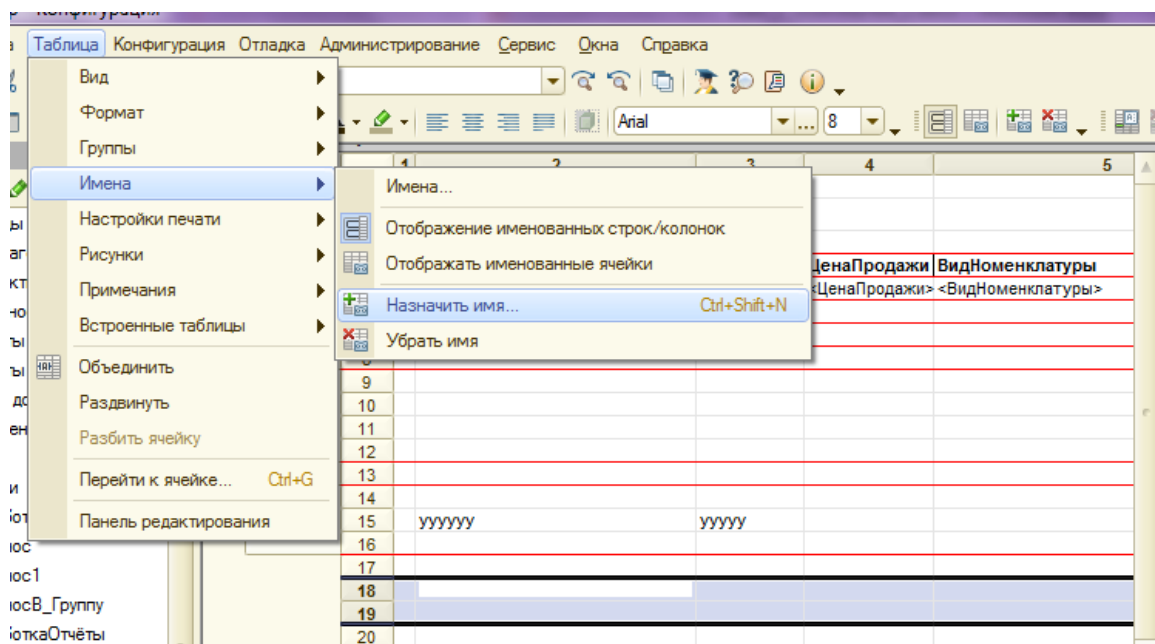


Рисунок 12.6 – Создание именованной области

Обращение к именованной области макета происходит методом ПолучитьОбласть("ИмяОбласти"):

**ОбластьЗаголовок = Макет.ПолучитьОбласть("Заголовок");**

Особенностью макетов является то, что их ячейки могут включать различный тип заполнения данными. Тип заполнения задаётся свойством «Заполнение» из группы свойств ячейки - макет.

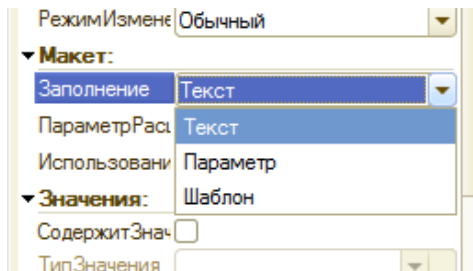


Рисунок 12.7 – Возможные типы заполнений данными ячейки макета

Возможны следующие типы заполнений:

- Текст
- Параметр
- Шаблон

Тип заполнения – Текст подразумевает, что данные как в ячейку вводятся, так и отображаются.

Заполнение «Параметр» позволяет использовать в качестве значения ячейки – реквизиты или переменные контекста, то есть объекта, для которого создаётся макет. Например, реквизиты справочника.

Заполнение «Шаблон» указывает, что в качестве значения используется шаблон, созданный на основе специального языка формул табличных документов 1с. В данном практикуме язык формул таблиц не будет рассматриваться.

Создание печатного документа должно включать следующие действия:

1. Создание печатного документа (на клиенте).

**ТабДок = Новый ТабличныйДокумент;**

2. Вызов макета (в приведённом ниже примере вызывается макет из обработки с именем "ОбработкаПечать" (на сервере).

**Макет= Обработки.ОбработкаПечать.ПолучитьМакет("Макет1");**

### 3. Подготовка данных, выводимых на печать.

В данной работе подготовка данных будет производится на основе запроса.

Запрос = Новый Запрос;

Запрос.Текст =

"ВЫБРАТЬ

| Номенклатура.Наименование,

| Номенклатура.Код,

| Номенклатура.ЦенаПродажи,

| Номенклатура.ВидНоменклатуры

| ИЗ

| Справочник.Номенклатура КАК Номенклатура

|

| СГРУППИРОВАТЬ ПО

| Номенклатура.ВидНоменклатуры,

| Номенклатура.Наименование,

| Номенклатура.Код,

| Номенклатура.ЦенаПродажи";

Результат = Запрос.Выполнить();

4. Заполнение формирование документа. Заполнение включает получение необходимой области в макете операцией макета ПолучитьОбласть() с последующим выводом данной области в документ методом документа Вывести(). При получении области происходит автоматическая подстановка значений параметров в документ и вычисление шаблонов. Пример кода заполнения документа приведён ниже.

ТабДок.Очистить();

ОбластьЗаголовок = Макет.ПолучитьОбласть("Заголовок");

ТабДок.Вывести(ОбластьЗаголовок);

СтрокаТабл = Макет.ПолучитьОбласть("Детали");

ВыборкаСрокиР = Результат.Выбрать();

Пока ВыборкаСтрокаР.Следующий() Цикл

СтрокаТабл.Параметры.Заполнить (ВыборкаСтрокаР);

ТабДок.Вывести(СтрокаТабл, ВыборкаСрокиР.Уровень());

КонецЦикла;

Непосредственно вставку в макет можно выполнять командой  
ОбластьШапка.Параметры.НомерДокумента = Док.Номер;

Вывод документ заполненных именованных областей макета.

ТабДок.Вывести(ОбластьШапка);

5. Отображение документа. Выполняется операцией документа  
Показать().

Необходимо помнить, что получение данных из информационной базы (обычно требуемая при заполнении документа) можно делать в программных модулях, выполняемых на сервере. Тогда как отображение документа, конечно же, возможно только на клиенте. Поэтому обычно формирование печатного происходит в двух процедурах. В процедуре, с директивой &наКлиенте – в данной процедуре происходит создание и отображение документа, и процедуре с директивой &НаСервере или &НаСервереБезКонтекста происходит заполнение документа.

&НаКлиенте

Процедура ПечатьСпр(Команда)

ТабДок = Новый ТабличныйДокумент;

ЗаполнениеДок (ТабДок);

ТабДок.Показать( );

КонецПроцедуры

&НаСервере

Процедура ЗаполнениеДок (ТабДок);

//Действия по заполнению документа (действия 2,3,4,5)

КонецПроцедуры

Формирование документа для печати может быть упрощено за счёт использования конструктора, вызываемого из программного модуля через контекстное меню. Конструктор, целесообразно вызывать для процедуры, выполняемой на сервере, то есть для создания действий по заполнению документа.

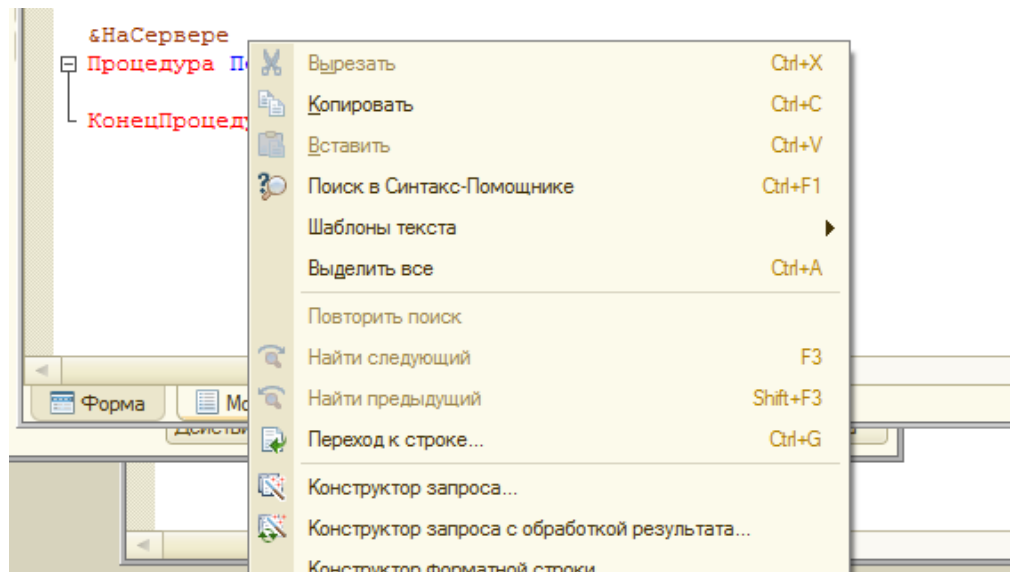


Рисунок 12.8 – Вызов конструктора запросов с обработкой результатов

При вызове конструктора необходимо задать режим вывода результата в табличный документ (закладка Тип обработки).

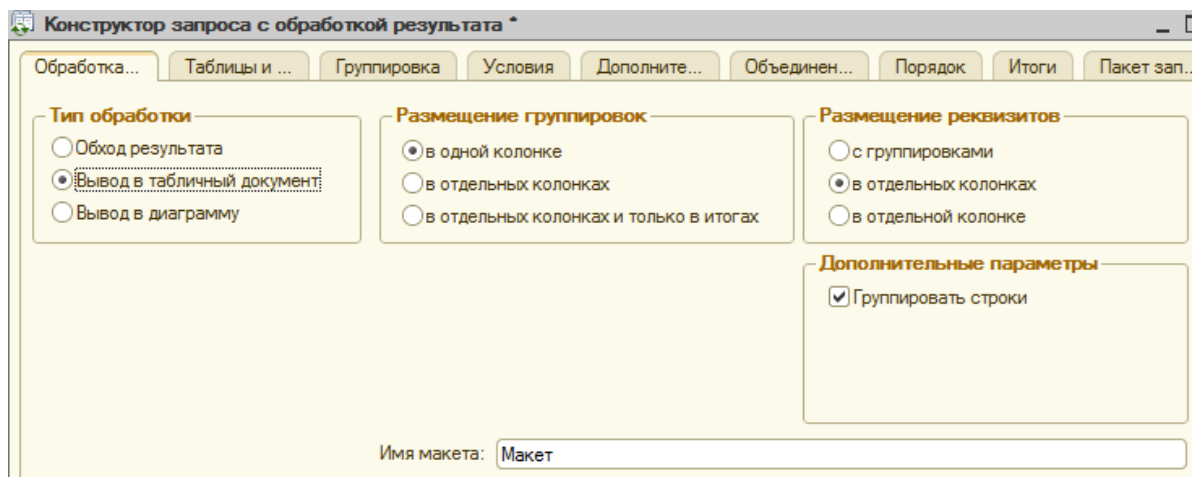


Рисунок 12.9 – Конструктор запросов с обработкой результатов

### Задание

1. Создайте следующие справочники и перечисления:

1.1. Перечисление Вид контрагента (Юридическое лица, физическое)

1.2. Справочник «Склады». Справочник без иерархии, без подчинения. Реквизитов и табличных частей нет.

1.3. «Контрагенты». Справочник иерархический (иерархия групп и элементов), без подчинения. Дополнительный реквизит «НаименованиеПолное», тип «Строка» 300 символов, допустимая сумма кредита, тип "число" 10 знаков, два знака после запятой. Вид контрагента – перечисление – вид контрагента.

1.4. «КонтактныеЛица». Справочник без иерархии, подчинен справочнику «Контрагенты». Дополнительный реквизит «Телефон», тип «Строка» 15 символов.

1.5. «Должности». Справочник без иерархии, без подчинения. Реквизитов и табличных частей нет. В нем определены 3 предопределенных элемента с именами «Бухгалтер», «ГлавныйБухгалтер», «Кассир».

2. Создайте обработки для вывода на печать содержимого справочника номенклатура. Сделать группировку по полям – вид номенклатуры. Вывести для каждого вида – наименование с минимальной ценой, количество наименований в группировке.

3. Создайте обработки для вывода на печать содержимого справочника контрагенты, сделать группировку по виду контрагента, для каждого вида группировки вывести минимальный кредит и максимальный.

### **Контрольные вопросы**

1. Что такое запрос. Охарактеризуйте язык формирования запросов. Какие ключевые слова в нём присутствуют.
2. Какие этапы обычно выполняются при работе запросами.
3. Как в запросе задаётся его содержание (текст запроса)?
4. Что такое конструктор запросов. Создайте с помощью конструктора запросов запрос.
5. Что такое табличный документ. Создайте табличный документ. В среде конфигуратора.
6. Что такое макет, каким образом создаётся макет.
7. Что такое именованная область. Создайте в макете именованные области.
8. Какие типы заполнения ячеек макета может быть.

9. Какие этапы включает создание печатного документа. Как создаётся и отображается пользователю заполненный табличный документ? На какой стороне (сервер, клиент) могут выполняться различные этапы?

## ЛАБОРАТОРНАЯ РАБОТА №4. РАБОТА С ДОКУМЕНТАМИ В СИСТЕМЕ 1С ПРЕДПРИЯТИЕ

### Работа с документами

Документ - одно из основных понятий системы «1С:Предприятие». При помощи документов организуется ввод в систему первичной информации о совершаемых хозяйственных операциях.

В большинстве своем документы, которые создаются в процессе настройки конфигурации, являются электронными аналогами стандартных бумажных документов, однако, использование этого типа данных может выходить далеко за рамки простой фиксации информации о хозяйственных операциях.

Дата и время наиболее важные характеристики документов, так как позволяют устанавливать строгую временную последовательность совершения операций.

Документы создаются, как и другие объекты метаданных в соответствующем узле дерева конфигурации.

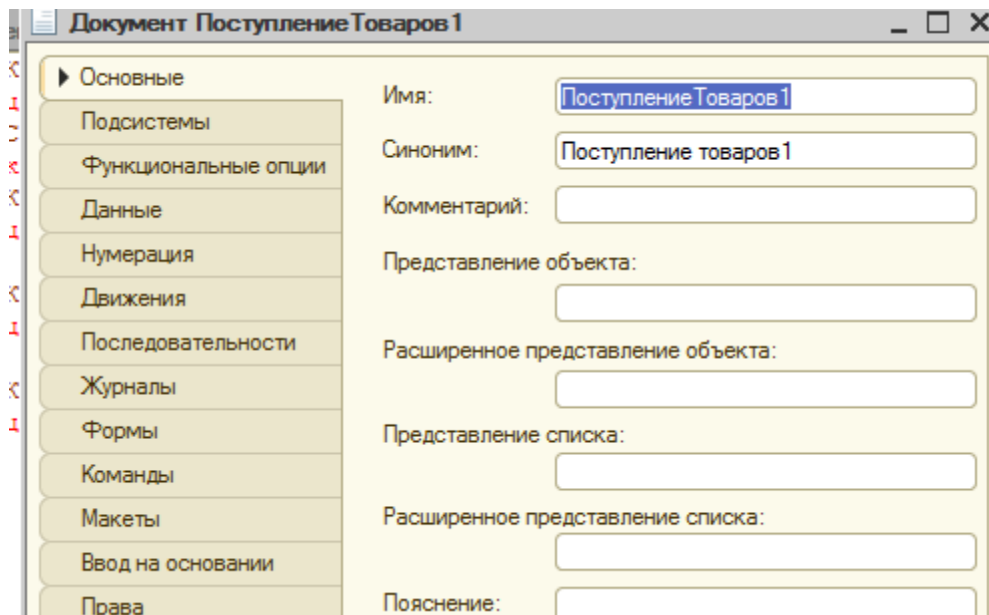


Рисунок 13 - Конструктор объекта – документ

### Реквизиты документа

Реквизиты документа. Разделяются на реквизиты и реквизиты



табличной части. Табличных частей может быть несколько

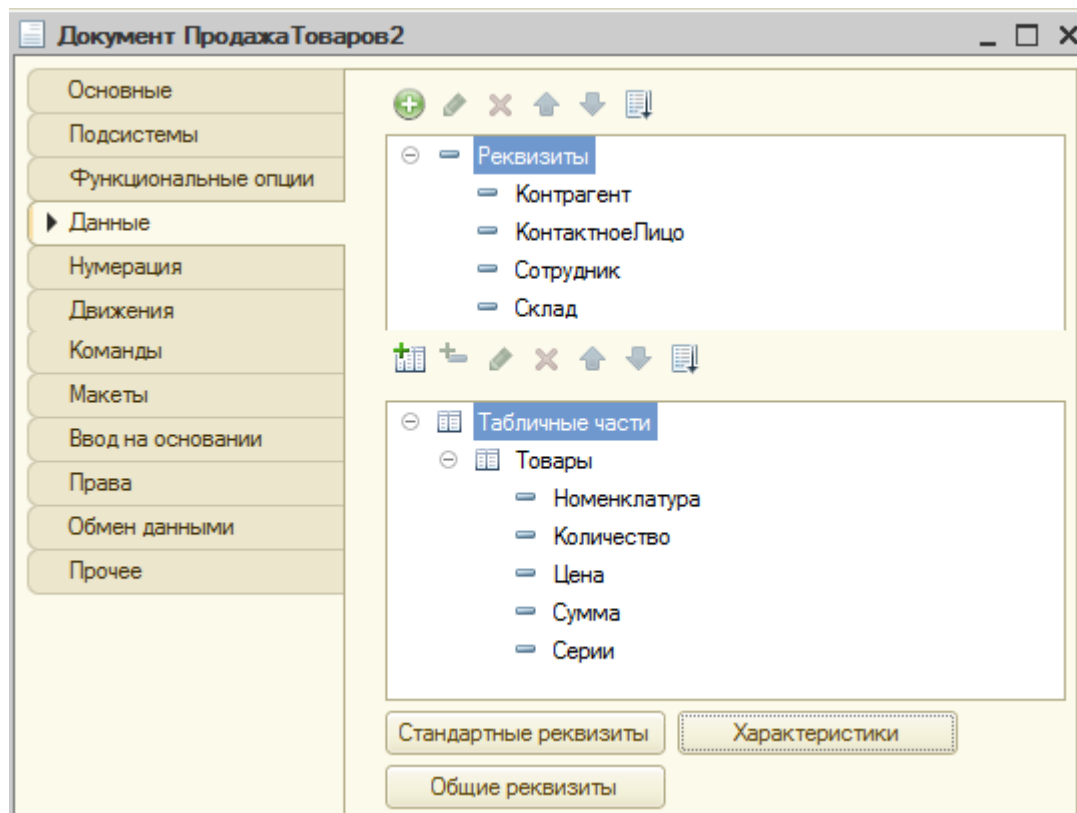


Рисунок 13.1 - Реквизиты документа

Среди реквизитов документа выделяются стандартные. То есть те, которые не определяет пользователь, но которые всегда есть во всех документах. Просмотр стандартных реквизитов – через соответствующую кнопку закладки данные.

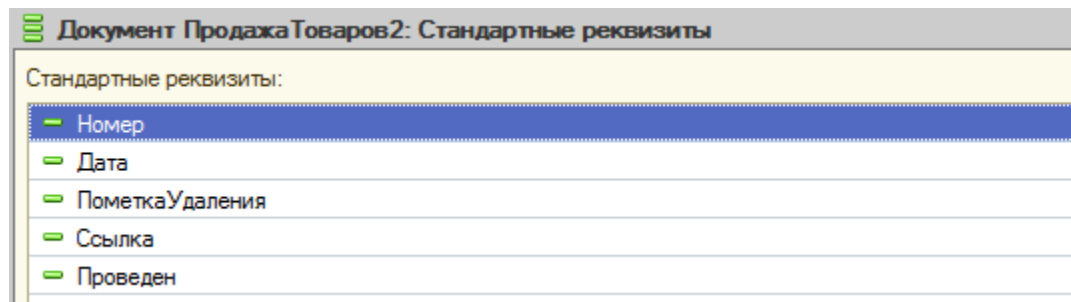


Рисунок 13.2 - Стандартные реквизиты документа

## Формы документа

Документ может содержать несколько видов форм

- Форма документа
- Форма списка документа
- Форма выбора документа
- Произвольная форма.

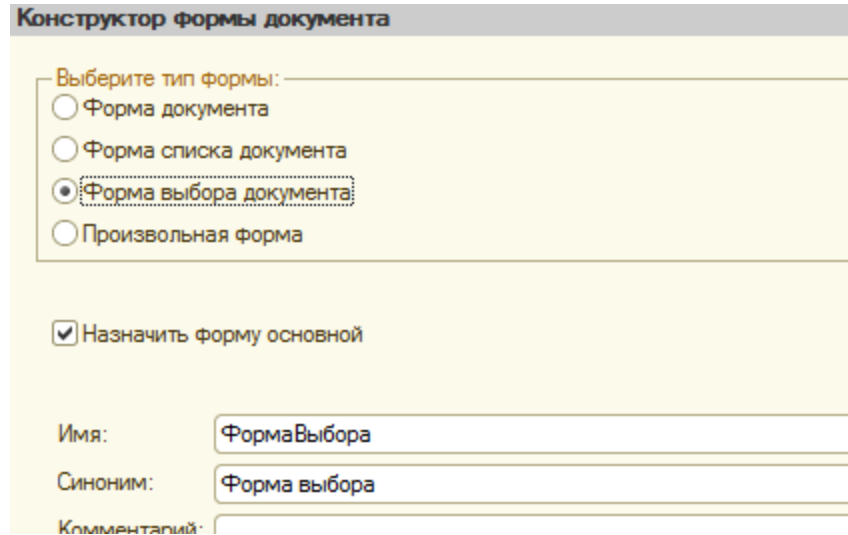


Рисунок 13.3 - Возможные типы форм документов

## Работа с формой документа

Форма документа предназначена для внесения данных и корректировки отдельного документа.

При создании формы документа на неё выносятся автоматически управляющие элементы, отображающие существующие реквизиты документа. Состав управляющих элементов можно корректировать, добавлять новые элементы.

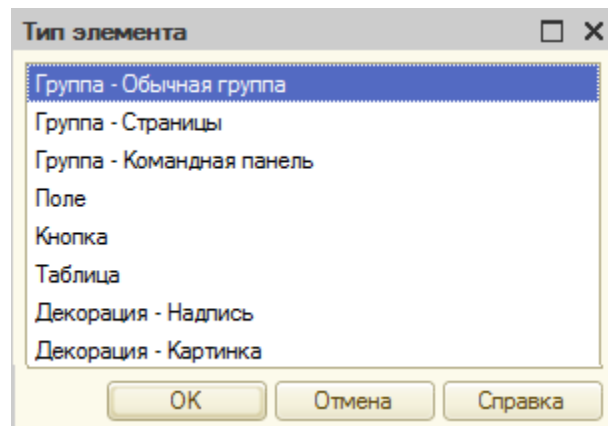


Рисунок 13.4 - Управляемые элементы, устанавливаемые на форму документа

Элемент "обычная группа" позволяет менять порядок расположения элементов, за счёт включения элементов в группу, кроме того, в свойствах элемента можно задать горизонтальную или вертикальную группировку.

#### События обработки управляющих элементов

Для управляющих элементов predetermined events. Events can be defined through the element's context menu (command Event) and through the element's properties. When a certain event is selected, a procedure for processing the corresponding event is automatically created.

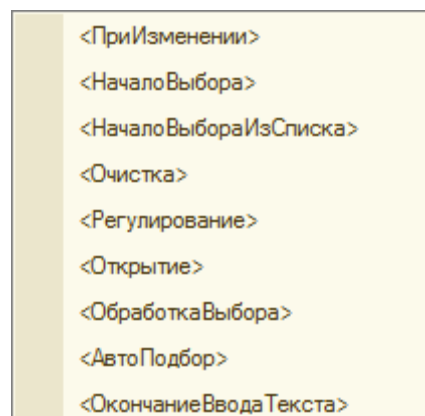


Рисунок 13.5 - Предопределённые события для управляющих элементов

Например, при выборе события <ПриИзменении>, для реквизита табличной части Количество будет создан программный модуль-обработчик, связанный с данным событием изменения значения этого реквизита. В качестве параметра для метода-обработчика передаётся ссылка на элемент - **Элемент**, с которым связано рассматриваемое событие.

**&НаКлиенте**

**Процедура ТоварыКоличествоПриИзменении (Элемент)**

**// Вставить содержимое обработчика.**

**КонецПроцедуры**

### **Обработка данных документа**

Обработка данных документа может производиться на основе модулей, выполняемых на клиенте и сервере, в зависимости от данных, которые требуется обработать и которые необходимы для обработки. На стороне клиента доступны данные управляющих элементов и реквизитов документа. Данные управляющих элементов доступны через коллекцию документа **Элементы** данная коллекция содержит все управляющие элементы Документа. Доступ к реквизитам осуществляется через главный реквизит **Объект**, соответствующий всему документу.

В том случае, когда требуются данные справочников, констант, макеты, сохранённые данные документа, необходимо использовать модули, выполняемые на сервере.

В частности, как рассмотрено в примере обработки.

Например, для созданного выше метода-обработчика **ТоварыКоличествоПриИзменении** пересчёт значения элемента Сумма может быть произведён следующим образом.

**&НаКлиенте**

**Процедура ТоварыКоличествоПриИзменении (Элемент)**

**// Вставить содержимое обработчика.**

**Стр=Элементы.Товары.ТекущиеДанные;**

Стр.Сумма = Стр.Количество \* Стр.Цена;  
КонецПроцедуры

Данный обработчик – выполняет пересчёт суммы при изменении количества соответствующих единиц номенклатуры в табличной части.

В процедуре используются следующие объекты.

**Элементы** – Коллекция управляющих элементов формы.

**Элементы.Товары** – табличная часть формы (в данном случае имя табличной части – Товары).

**Товары.ТекущиеДанные** – текущая строка табличной части. Обратите внимание, если реквизитом табличной части является агрегатный объект, то его свойства нельзя увидеть через **ТекущиеДанные**.

Через переменную **Стр** происходит выборка текущей строки и зачем пересчёт значения суммы.

Пусть необходимо сделать обработчика события изменения реквизита Номенклатура табличной части документа. При изменении номенклатуры автоматически должна подставляться цена новой выбранной номенклатурной единицы. В данном случае для получения цены необходимо создать вспомогательный программный модуль, например функцию, обеспечивающий получение цены номенклатурной единицы. В качестве входного параметра для данной функции целесообразно задавать экземпляр соответствующего справочника (номенклатура). Данный программный модуль должен выполняться на сервере.

Процедура ТоварыНоменклатураПриИзменении (Элемент)

// Вставить содержимое обработчика.

Стр = Элементы.Товары.ТекущиеДанные;

Стр.Цена = ПолучитьЦенуНоменклатуры  
(Стр.Номенклатура);

ТоварыКоличествоПриИзменении (Элемент);

КонецПроцедуры

Обработчик события "При изменении" поля Номенклатура в табличной части. Для получения цены номенклатуры создана функция [ПолучитьЦенуНоменклатуры](#) (цену нельзя получить через объект [стр.Номенклатура](#)).

[&НаСервереБезКонтекста](#)  
[Функция ПолучитьЦенуНоменклатуры \(Номенклатура\)](#)  
[Возврат Номенклатура.ЦенаПокупки;](#)  
[КонецФункции](#)

Для выборки данных всей табличной части документа лучше использовать объект специальной коллекции Документы. Данная коллекция включает все документы, сохранённые в системе. Для получения одного конкретного экземпляра документа необходимо позиционироваться на требуемом элементе данной коллекции с помощью метода НайтиПоНомеру или НайтиДокумент.

Док=Документы.<ВидДокумента>.НайтиПоНомеру(НомерДок);

В данном случае объект Док будет соответствовать документу с требуемым номером.

Для перебора данных табличной части документа можно использовать цикл.

[Для Каждого ТекущаяСтрока Из Док.Товары Цикл](#)  
[К = текущаяСтрока.Количество;](#)  
[КонецЦикла;](#)

В данном случае выводится в переменную ККК значение реквизита табличной части Количество (Товары – имя табличной части документа).

### **Создание печатных форм документа с помощью конструктора печати.**

Документ может иметь одну или нескольких печатных форм. Для создание печатных форм может использоваться конструктор

печати документа, вызываемый контекстным меню на элементе данного документа дерева конфигурации.

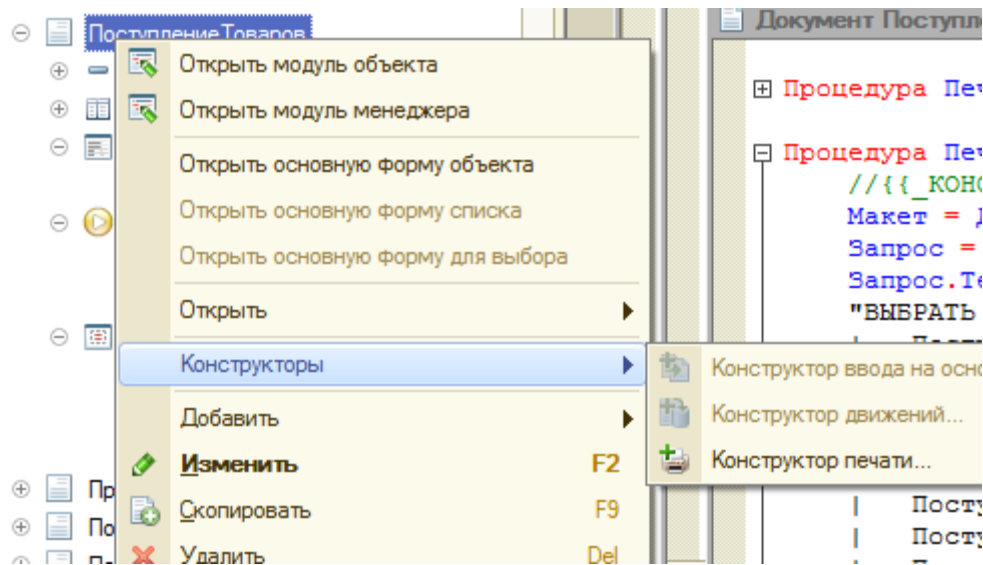


Рисунок 13.6 - Вызов конструктора печати документа

Так как конструктор настроен на отнесение команды к одной из групп команд. Предварительно необходимо создать соответствующую группу команд. Группа команд, это элемент интерфейса, обычно закрепляемый в верхней части формы объектов конфигурации. Группа команд позволяет отображать и вызывать закреплённые за ней команды. Причём команды отображаются только те, которые созданы в данном документе. Группы команд создаются же для конфигурации в целом.

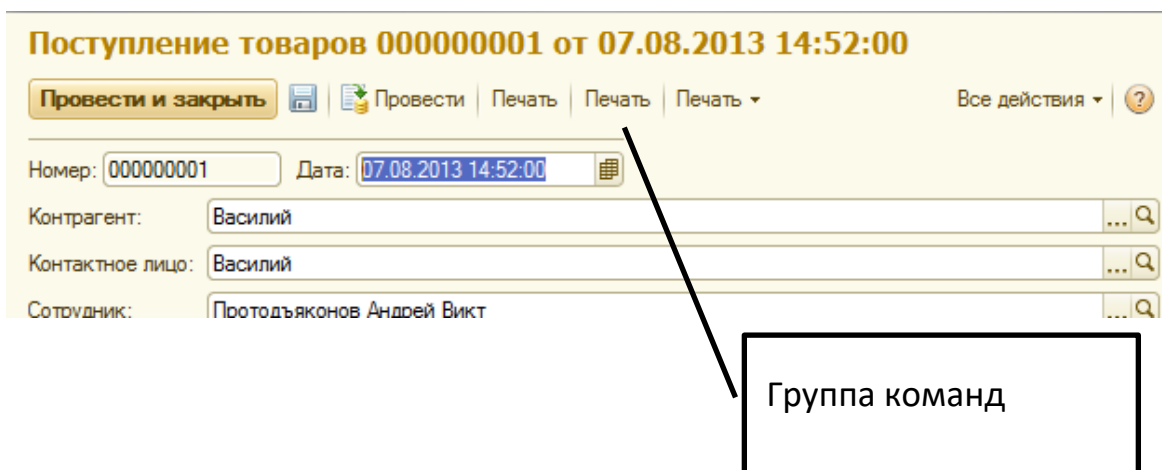


Рисунок 13.7 - Отображение группы команд на объекте конфигурации

Создаются группы команд в общем узле конфигурации.

Для созданной группы команд должна быть указана категория. Категория группы команд определяет, где, в каком интерфейсе будет отображаться данная группа.

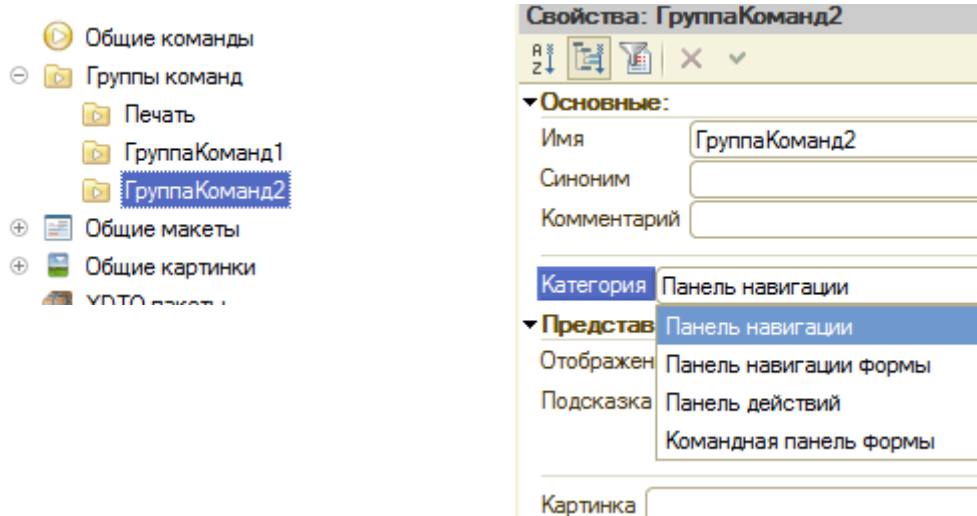


Рисунок 13.8 - Группы команд в дереве конфигурации и задание категории для группы команд

При создании печатной формы через конструктор печати конструктор создаёт следующие элементы.

1. Создаётся команда в соответствующем элементе объекта конфигурации (в данном случае в документе). Команда принадлежит не форме, а непосредственно документу, как объекту конфигурации.

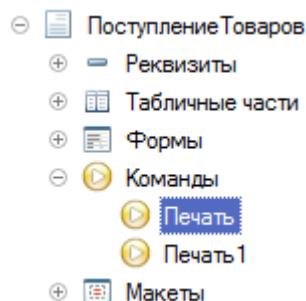


Рисунок 13.9 - Команда в соответствующем элементе объекта конфигурации

2. Создаётся макет печати.

3. Создаётся модуль команды, связанный с созданной командой. Модуль команды включает клиентскую процедуру,



вызываемую при вызове команды, и серверную, вызываемую из клиентской.

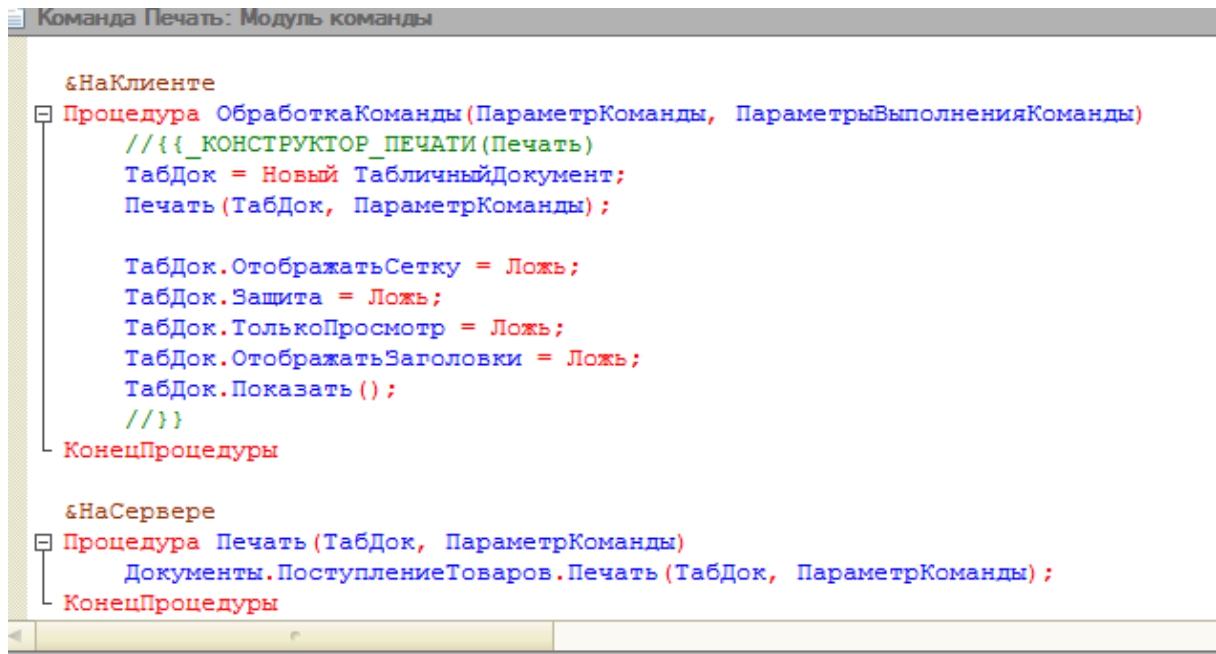


Рисунок 13.10 - Модуль команды

4. Кроме того, в модуле менеджера документа создаётся программный модуль, который непосредственно получение данных документа и внесение из в макет.

#### *Создание печатной формы вручную.*

Для создания печатной формы вручную необходимо:

1. Создать макет в документе. Разметить в макете именованные области (обычно - шапка, табличная часть, подвал). Задать выводимые ячейки – параметры (свойство тип заполнения в данных ячейках должно быть установлено как параметр).

2. Создать на форме управляющий элемент и команду, связанную с ним.

3. Создать клиентскую процедуру. В котором осуществляется создание табличного документа, вызов серверного метода (заполняющего документ) и отображение документа (ТабДок = Новый ТабличныйДокумент; ПроцПечать (ТабДок, ПараметрКоманды); ТабДок.Показать();).

4. Создать серверный метод, в котором осуществляется вызов макета, вызов и заполнение его именованных областей.

Макет = Документы.ПоступлениеТоваров.ПолучитьМакет ("Макет"); - вызов макета

ОбластьШапка = Макет.ПолучитьОбласть("Шапка"); - вызов именованной области – шапка.

Заполнение параметров:

ОбластьШапка.Параметры.НомерДокумента=Док.Номер;

Вывод документ заполненных именованных областей макета.

ТабДок.Вывести(ОбластьШапка);

### Задание

#### *Вариант 1.*

Должны быть созданы документы.

1. Поступление товаров
2. и Реализация.

Для документа Поступления товаров должны быть определены следующие реквизиты.

Реквизиты шапки

1. «Контрагент»  
(тип <СправочникСсылка.Контрагенты>)
2. «КонтактноеЛицо»  
(тип <СправочникСсылка.КонтактныеЛица>)
3. «Склад» (тип <СправочникСсылка.Склады>)
4. «СуммаДокумента»  
(тип <Число> длина 15, точность 2)

В справочник должна быть одна табличная часть (Товары)

Реквизиты табличной части

1. Номенклатура  
(тип <СправочникСсылка.Номенклатура>)
2. Количество (тип <Число> длина 10, точность 0)
3. Цена (тип <Число> длина 10, точность 2)
4. Сумма (тип <Число> длина 10, точность 2)
5. Серия (тип <СправочникСсылка.Серии>)

Для элементов табличной часть Количество, Номенклатура, Цена должны быть созданы обработчики событий "При изменении".

Документ Реализация необходимо создать копированием в дереве конфигурации документа Поступление товаров с последующей корректировкой.

### **Вариант 2.**

Должны быть созданы документы. Данные о выходах и Начисление заработной платы.

Для документа Данные о Выходах должны быть определены следующие реквизиты.

Реквизиты шапки

1. Подразделение СправочникСсылка.Подразделения>)
2. «Расчётчик»  
(тип СправочникСсылка.Сотрудники>)
3. «Сумма Документа»

(тип <Число> длина 15, точность 2)

В справочник должна быть одна табличная часть (Отработанные часы)

Реквизиты табличной части

1. Сотрудник (тип СправочникСсылка.Сотрудники>)
2. Сумма Часов (тип <Число> длина 10, точность 0)
3. Тариф (тип <Число> длина 10, точность 2)

Документ Начисление необходимо создать копированием в дереве конфигурации документа Данные О Выходах с последующей корректировкой (необходимо добавить атрибуты, премиальные, Начислено (тип <Число> длина 10, точность 2). Итого начислено

Для элементов табличной части документа начислено необходимо создать обработчики события "При Изменении" для реквизитов Сумма часов, тариф, премиальные обработчик должен пересчитывать Начислено, итого начислено.

Доп. задание обработчик "При Изменении" на реквизит табличной части "Сотрудник". При изменении сотрудника должен браться тариф должности данного сотрудника.

***Для обоих вариантов -***

Для документа должна быть создана печатная форма через, конструктор печати (вывод всех реквизитов документов, итог выводить в подвале).

Для документов должна быть создана печатная форма вручную, аналогичная созданной с помощью конструктора. То есть должен быть создан макет в документе, отображающий необходимые поля. Создана команда и модули (два модуля клиентский и серверный для вывода данных)

В серверном модуле произвести получение данных на основе выборки данных документа и заполнение их в макет.

**Контрольные вопросы**

1. Роль документа в системе 1с
2. Как создаются документы в режиме конфигурации.
3. Создать по заданию преподавателя заданный документ.
4. Как задаются реквизиты у документа. Какие группы реквизитов можно выделить. Что такое стандартные реквизиты.
5. Как можно получить данные документа в программных модулях.
6. Какие данные документа можно получить из модулей, выполняемых на клиентской части.
7. Каким образом можно получить данные текущей строки табличной части документа?
8. Какие predetermined события определены для управляющих элементов документа. Как создать обработчик какого-либо события.
9. Создать для заданного элемента заданный обработчик.

## ЛАБОРАТОРНАЯ РАБОТА №5. РАБОТА С РЕГИСТРАМИ ОПЕРАТИВНОГО УЧЁТА.РЕГИСТРЫ НАКОПЛЕНИЯ. РЕГИСТРЫ СВЕДЕНИЙ

Объектами, отображающими движение средств предприятия то есть, хозяйственную операцию являются документы. Однако, движение одних и тех же средств может отображаться различными документами. Кроме того, данные о движении хозяйственных средств предприятия в документах хранятся в несистематизированном виде, что затрудняет формирование отчётности. Поэтому для обеспечения удобства анализа данных деятельности предприятия в системе 1с были введены дополнительные объекты – Регистры.

*Регистры - объекты конфигурации, которые позволяют создавать в базе данных структуры, предназначенные для накопления информации в удобном для последующего анализа виде.*

Использование таких хранилищ данных позволяет, с одной стороны, накапливать в них данные, поставляемые различными документами (или другими объектами информационной системы), а с другой стороны, легко создавать нужные отчеты или использовать эти данные в алгоритмах работы конфигурации. Обычно регистры не доступны пользователю для непосредственного ввода данных ввод в данные регистров (движение регистров) - обычно производится через другие объекты – регистраторы. Для каждого регистра могут быть определены регистраторы - объекты, обеспечивающие его движение. С другой стороны, объекты информационной базы могут являться регистраторами для различных регистров.

Регистр имеет следующие характеристики:

- измерения – параметры, в которых задаётся информация, например – Склад, Товар;
- ресурсы – непосредственно сама информация, показатели, отображающие некоторое значение в введенных измерениях, например, *количество* некоторого товара на определённом складе, или *стоимость* данного товара на данном складе.
- Кроме того, может быть определена некоторая дополнительная информация в регистре – *реквизиты*.

- Данные об объекте-регистраторе, вызвавшем движение, также обычно сохраняются в виде характеристики регистра.

Выделяются несколько типов регистров, обеспечивающие учёт различных сторон деятельности предприятия.

- Регистры сведений
- Регистры накопления
- Бухгалтерские регистры
- Регистры расчёта.

*Регистры накопления* служат для накопления данных об остатках или движения хозяйственных средств. Соответственно выделяются регистры накопления остатков и регистры накопления движения.

Регистры накопления являются подчиненными объектами конфигурации.

Изменение состояния регистра накопления происходит, как правило, при *проведении документа* и заключается в том, что в регистр добавляется некоторое количество записей. Каждая запись содержит

- значения измерений,
- значения приращений ресурсов,
- ссылку на документ, который вызвал эти изменения (регистратор),
- и «направление» приращения (приход или расход).  
Направление приращения является системной характеристикой и создаётся автоматически для любого регистра накоплений.

Такой набор записей называется движениями регистра накопления.

Пример набора движения регистров приведён в табл. 5.

При отмене проведения документа удаляются соответствующие данному документу записи в регистрах накопления.

Для того что бы обеспечить использование регистров остатков необходимо

1. Создать в дереве объектов конфигурации требуемые регистры. Задать для регистров - измерения, ресурсы (при необходимости создаются реквизиты регистра)
2. Связать регистры с регистраторами. То есть с объектами системы, обеспечивающими движение регистров (с определёнными документами)
3. Задать в регистраторах движение регистров. Для документов движение выполняется при их проведении.

Создание регистров движений, как и других объектов конфигурации выполняется в дереве объектов конфигурации, как и для других. Для задания характеристик регистров используется, как и для других объектов конфигурации окно редактирования объектов.

Таблица 5 – Набор движения регистров

| Ссылка на регистратор (документ) | Номер строки в регистраторе | Измерения  |           | Ресурсы    |       |
|----------------------------------|-----------------------------|------------|-----------|------------|-------|
|                                  |                             | Товар      | Склад     | Количество | Сумма |
| Ref1                             | 1                           | чайник     | Основной  | 10         | 30000 |
| Ref1                             | 2                           | Вентилятор | Основной  | 5          | 25000 |
| Ref2                             | 1                           | фен        | Розничный | 20         | 40000 |
| Ref3                             | 1                           | Пылесос    | Основной  | 5          | 50000 |
| Ref3                             | 2                           | Чайник     | Основной  | 20         | 40000 |

Рисунок 14 - Создание параметров регистра накоплений

Регистры необходимо отнести к определённым подсистемам, для обеспечения возможности их просмотра через интерфейс подсистем.

Параметры реквизитов – измерения, ресурсы и реквизиты задаются на закладке данные.

Рисунок 14.1 - Данные регистров



Связывание регистраторов с регистрами происходит или в окне редактирования объекта конфигурации, назначаемого для регистра регистратором (закладка "движения") или в окне редактирования регистра (закладка регистраторы).

### **Задание движения регистров.**

Как уже отмечалось, движение регистров выполняется при проведении документа, являющегося регистратором для данного регистра.

Для задания движения необходимо связать данные объектов регистраторов с измерениями регистра и ресурсами.

Движение регистров, связанное с регистраторами, может быть создано с помощью конструктора движений. Кнопка вызова конструктора движений становится активна на окне редактирования объектов при задании для объекта (документа) регистров, в которых выполняется движение связанное с данным объектов (закладка окна редактирования объекта – "движения")

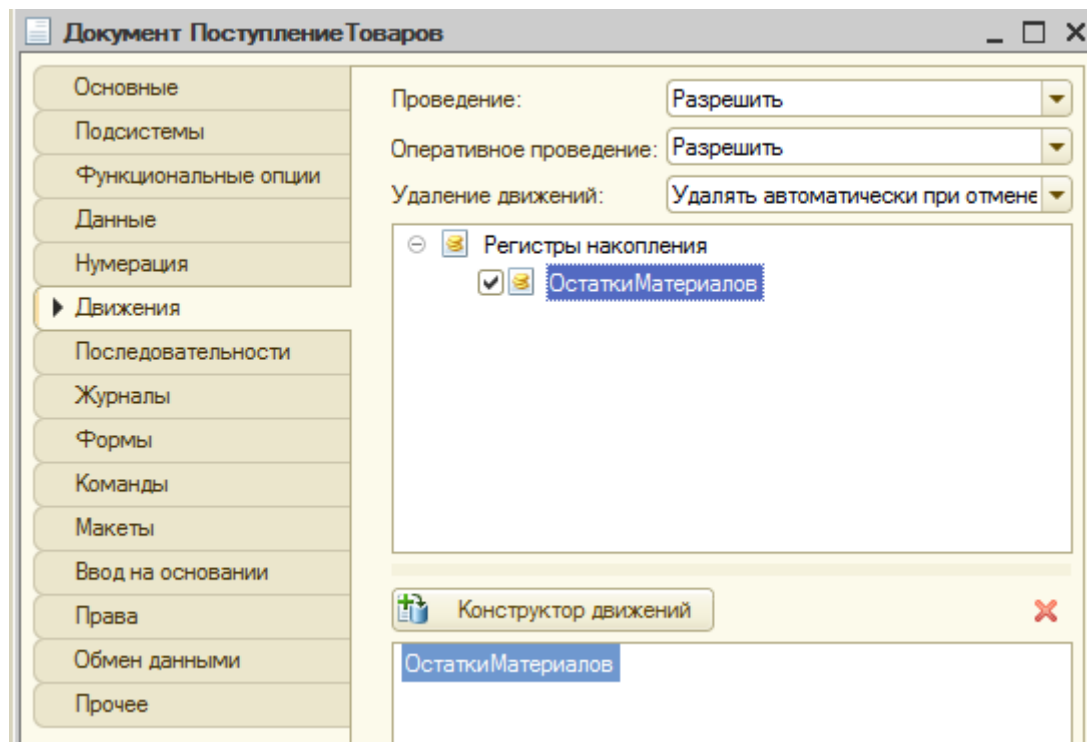


Рисунок 14.2 - Задание движения регистров для документа. Вызов конструктора движений

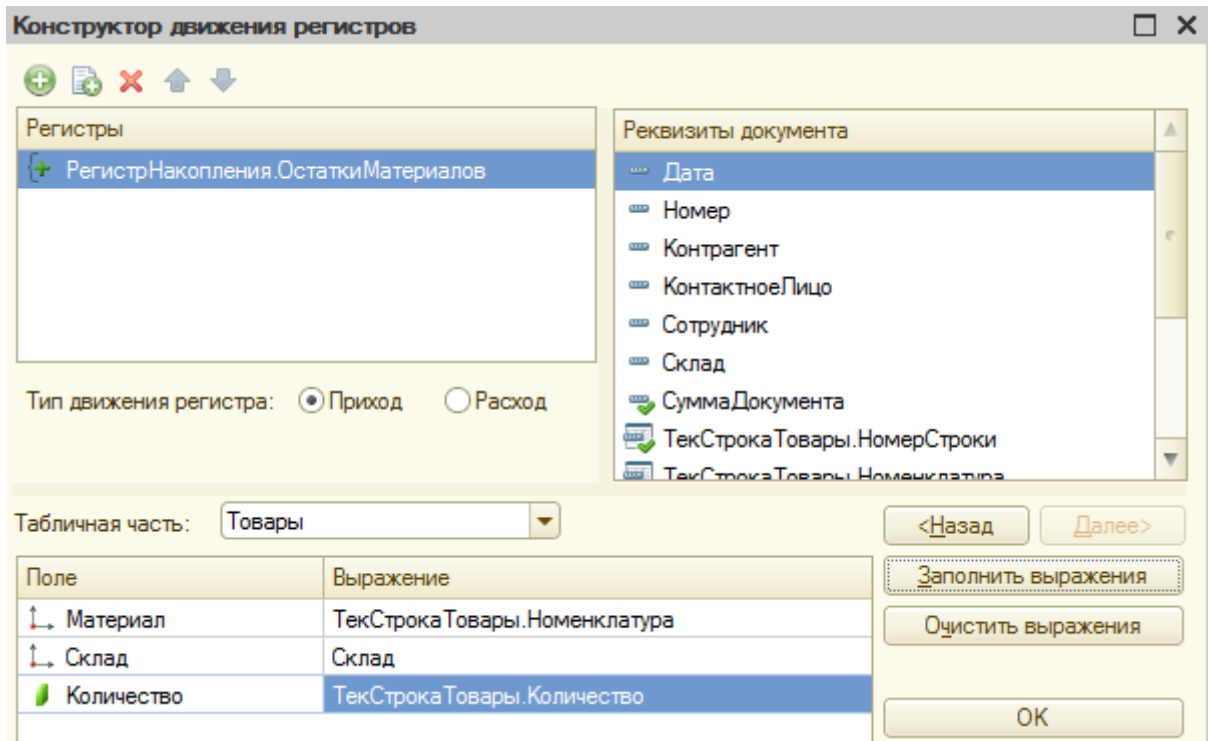


Рисунок 14.3 - Конструктор движений

В конструкторе движений необходимо

- в поле "Регистры" выбрать регистры, для которых выполняются движения.
- Для полей регистра (измерений и ресурсов) регистра задать формулы, по которым выполняются расчёт их значений.

При заполнении формул необходимо выбрать необходимую табличную часть документа из соответствующего выпадающего списка. Для заполнения можно воспользоваться кнопкой "заполнить выражения" при этом сопоставление полей регистра и реквизитов документа произойдёт на основе их типов. Для уточнения выражений может потребоваться ручное заполнение. При этом необходимо выбрать необходимое выражение и кликнуть по необходимому реквизиту в списке реквизитов.

После заполнения выражений при нажатии кнопки "Ок" будет сформирована процедура ОбработкаПроведения (Отказ, Режим) в модуле документа. Данная процедура будет выполняться при проведении документа.

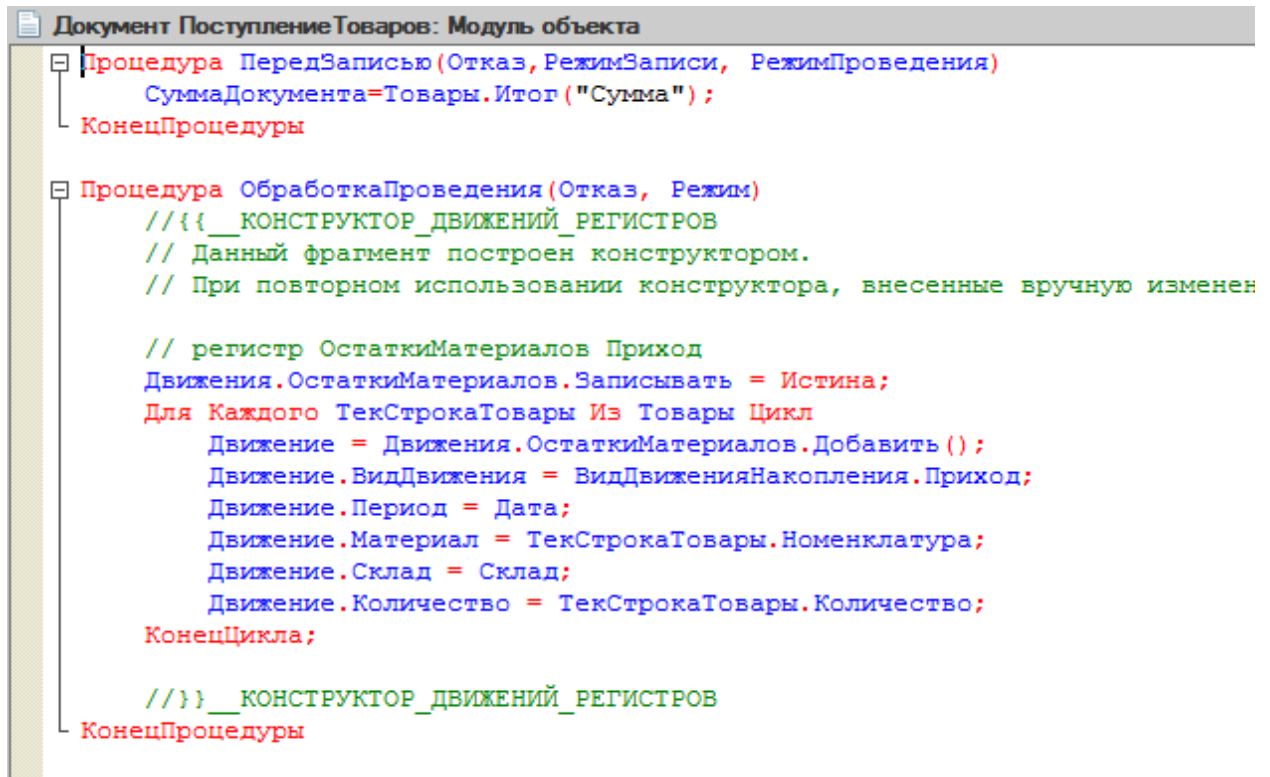


Рисунок 14.4 - Содержание процедуры обработка проведения

В данной процедуре используются следующие объекты.

Движения – коллекция объектов, связанных с движения документа (регистры). Доступна в контексте документа.

Движение – объект, соответствующий движению регистра, выбранного в коллекции Движения. Содержит все поля соответствующего регистра.

Товары – коллекция, соответствующая табличной части документа. (уже проходили)

Для выполнения движения необходимо

- Установить флаг для требуемого регистра Записывать в значение Истина - Движения.<ИмяРегистра>.Записывать = Истина.
- создать объект, соответствующий новой записи движения методом <ИмяДвижения> = Движения.<ИмяРегистра>.Добавить()
- Заполнить поля данного объекта движения (<ИмяДвижения>.<ИмяПоля>=Выражение). Данные выражения автоматически формировались в конструкторе движений. Данные действия не сложно выполнить вручную.

При этом создание объекта движения и его заполнение выполняется в цикле на каждую строку табличной части.

### Просмотр данных движения регистров.

В прикладных решениях 1с пользователю обычно непосредственно не предоставляется возможность изменять данные движения регистров. Однако система предоставляет возможность просмотра данных движения (это выполняется при отнесении регистров к подсистеме), однако, по умолчанию данная возможность в интерфейсе не активирована. Доступ к интерфейсу подсистем. Узел подсистемы/контекстное меню/Все подсистемы (ПравоеПодОкно-Командный интерфейс).

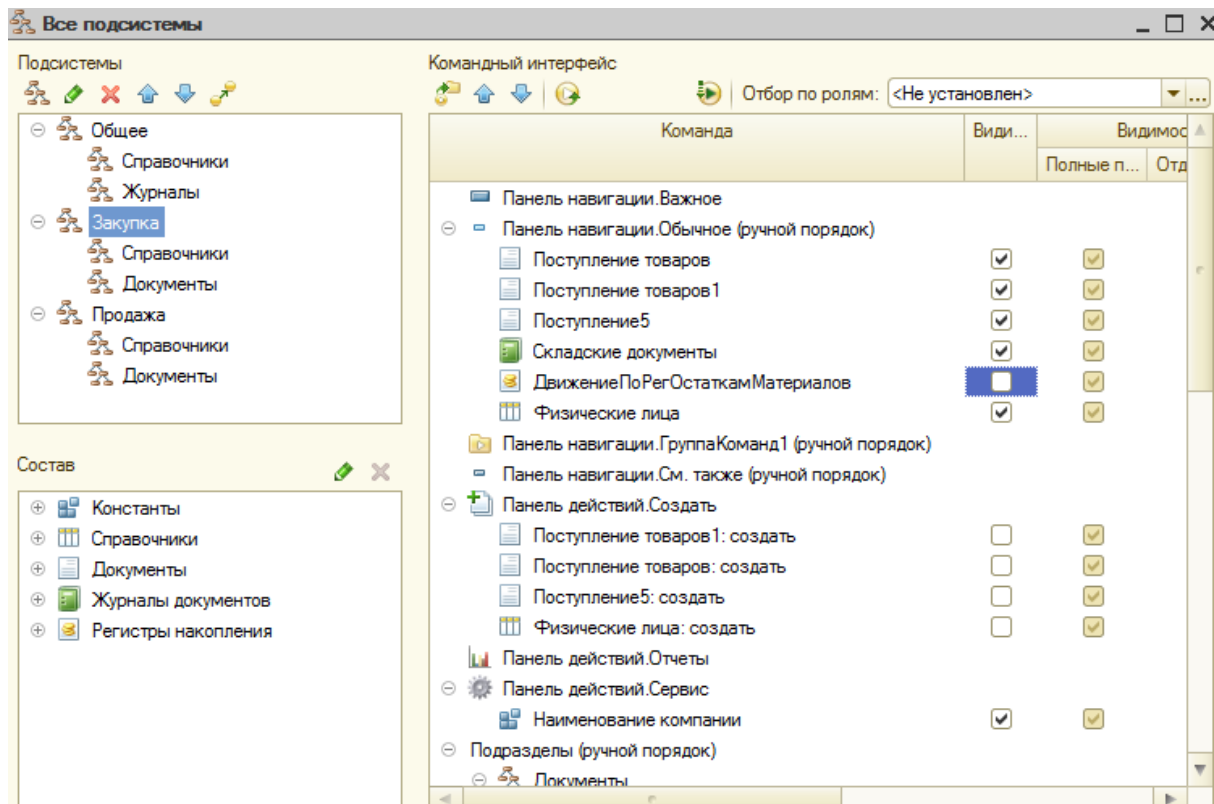


Рисунок 14.5 - Активация команды просмотра данных движения регистров

При этом целесообразно, чтобы просмотр регистров был отделён от других команд панели навигации переместит их в отдельную группу из обычной.

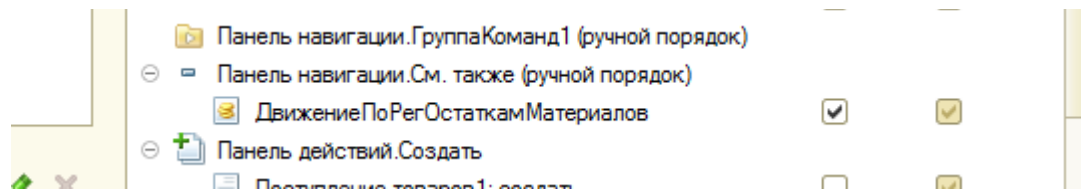


Рисунок 14.6 - Перемещение команды просмотра регистров в отдельную панель в панели навигации

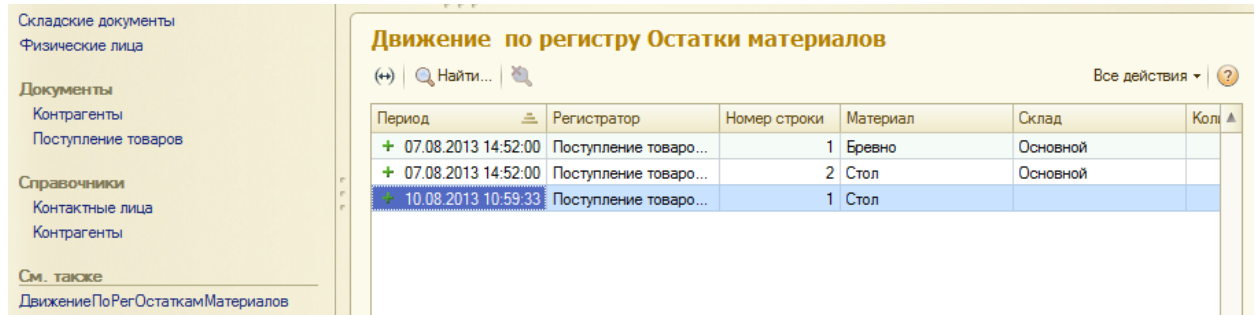


Рисунок 14.7 - Команда просмотра регистров и окно просмотра регистров в режиме предприятия

## Задание

### Вариант1

1. Создать регистры:  
ОстаткиТоваров (иНоменклатура, иСклады, рКоличество, рСумма).
2. Задать в документах.  
Поступление товаров и Реализация движение данного регистра в соответствии с логикой документа.
3. Создать справочник Сотрудники
4. Реквизиты -  
Фамилия строка (12), Имя Строка (12), Отчество Строка (12),  
ТабельныйНомер (Число (10)),
5. Добавить в документ Реализация атрибут – сотрудник.
6. Создать регистр  
ПродажиПродавца (иНоменклатура, иПродавец, рКоличество, рСумма)

7. Создать документ возвратТовара (шКонтрагент, табЧасть (товар, датаПокупки, количесство, ПродавецПродавшийТовар. справочникСотрудники))
8. Скорректировать вручную движения в документе реализацияТовара, что бы в них выполнялось движение регистра продажи продавца.
9. Написать вручную движение регистров при проведении документа ВозвратТоваров. Двигаться должны оба регистра.

### **Вариант 2**

1. Создать регистры:  
Выработанные Часы (иСотрудник, иПодразделение, рКоличествоЧасов).  
НакопленнаяЗП (иСотрудник, иПодразделение.рНачислено Сумма)
2. Создать документ – выплаченнаяЗпл (шПодразделение, шПериод, тчВыплаты (Сотрудник, Сумма).
3. Задать в документах ДанныоВыходах – движение соответствующего регистра.
4. Задать в документах НачислениеЗаработной – движение соответствующих регистра.
5. Задать движение регистров в документе ВыплатыЗПл вручную.
6. Обеспечить возможность просмотра данных движения регистров.

### **Контрольные вопросы**

1. Что такое регистры. Зачем они используются. Какие характеристики имеют регистры.
2. Какие типы регистров выделяются. Их назначение.
3. Как обычно вводятся данные в регистры. Что подразумевается под "движением регистров"? Что такое объекты "Регистраторы". Какие объекты обычно используются в качестве регистраторов
4. Что такое регистры накопления. Какие разновидности регистров накопления выделяются?
5. Какие характеристики существуют у регистров накопления.

6. Как создаются регистры накопления. Какие параметры регистров накопления создаются, что они обозначают.
7. Как связываются регистры с объектами, обеспечивающими их движение (регистраторы). Как в регистрах отображаются регистраторы вызвавшие движения.
8. Как вызывается конструктор движений регистров, и что нужно задать в нём для формирования движения регистров. Какие участвуют объекты в процедуре выполнения движений регистров.
9. Создать процедуру для выполнения движения регистров.

## ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №11. ИСПОЛЬЗОВАНИЕ МЕХАНИЗМА ОПОВЕЩЕНИЯ.

### Использования объекта Сообщение Пользователю

Объект «СообщениеПользователю» пришел на замену команды «Сообщить», которая использовалась в обычном интерфейсе, и перешла в управляемый. Команда удобна своей простотой и тем, что она, в отличии от других вариантов оповещения пользователя, доступна как на клиенте, так и на сервере. На данный момент команда является устаревшей и ее не рекомендуется применять для управляемого интерфейса.

Объект «СообщениеПользователю» более громоздкий, но имеет ряд дополнительных возможностей, о которых и пойдет речь в данной статье. Объект можно использовать только для управляемого интерфейса.

Объект также можно использовать как на клиенте, так и на сервере. Основным преимуществом перед командой «Сообщить», является возможность **привязки сообщений к реквизитам формы**. Дополнительно, при нажатии на текст сообщения, можно открывать новую форму с привязкой сообщения к ее реквизитам. Однако, следует понимать, что **основным назначением объекта является информирование об ошибках**, так как сообщение, привязанное к реквизитам формы всегда имеет заголовок «Ошибка»:

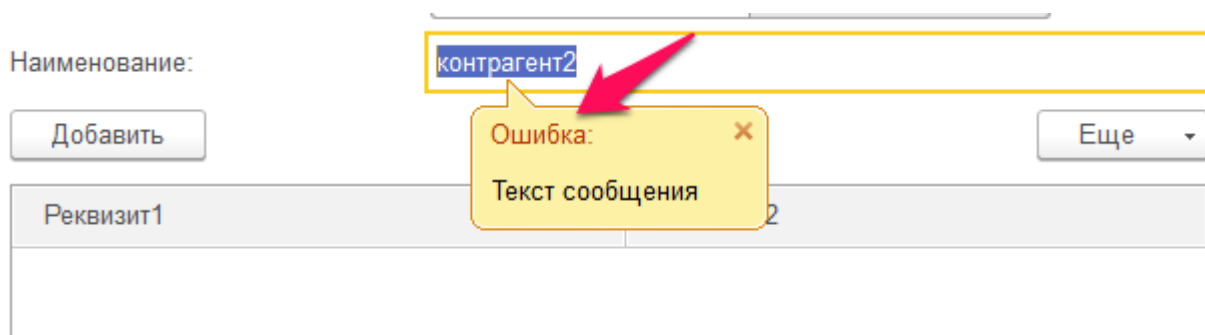


Рисунок 15 - Информирование об ошибке

Рассмотрим различные варианты использования объекта «Сообщение Пользователю».



**Вариант 1. Вывод сообщения в текущую активную форму с привязкой к ее элементам.**

Сообщение = Новый СообщениеПользователю;

Сообщение.Текст = "Сообщение";

Сообщение.Поле = "Объект.Наименование";

Сообщение.Сообщить();

**Поле** – путь к реквизиту, к которому необходимо привязать сообщение. Путь к реквизитам объекта формы необходимо указывать задав ключевое слово «Объект». Путь к остальным реквизитам формы указывается просто, в виде наименования реквизита:

Сообщение.Поле = "РеквизитФормы";

Можно прикрепить сообщение к строке табличной части объекта:

Сообщение.Поле = "Объект.Товары[1].Номенклатура";

Следует иметь в виду, что **сообщение всегда выводится в текущую активную форму**. Даже в том случае, если сообщение формируется в другой форме, но она еще не открыта.

Например, при обработке события формы «ПриОткрытии» можно выполнить проверку некоторого условия и выводить сообщение об ошибке не открывая форму. Сообщение будет выведено в форму, из которой выполнялось открытие новой формы.

**Вариант 2. Вывод сообщения в текущую активную форму с привязкой к элементам дополнительной формы**

Сообщение = Новый СообщениеПользователю;

Сообщение.Текст = "Сообщение";

Сообщение.Поле = "Наименование";

Сообщение.КлючДанных = Объект.Склад;

Сообщение.ПутьКДанным = "Объект";

Сообщение.Сообщить();

**Поле** – наименование реквизита без указания ключевого слова «Объект».

**КлючДанных** – ссылка на объект дополнительной формы, к реквизиту которой требуется привязать сообщение. Система попытается найти открытую форму по значению стандартного параметра формы «Ключ». Если форма не будет найдена, будет открыта новая.

**ПутьКДанным** – имя основного реквизита формы, через который можно получить доступ к реквизитам объекта.

В данном варианте использования, **сообщение можно привязать только к реквизитам объекта формы.**

**Вариант 3. Вывод сообщения в форму, которая не является активной**

```

ПараметрыФормы = Новый Структура;
ПараметрыФормы.Вставить("Ключ", Объект.ОсновнойДоговор);
Форма = ОткрытьФорму("Справочник.Договоры.ФормаОбъекта",
ПараметрыФормы, ЭтаФорма);

```

```

Сообщение = Новый СообщениеПользователю;
Сообщение.ИдентификаторНазначения =
Форма.УникальныйИдентификатор;
Сообщение.Текст = "Сообщение";
Сообщение.Поле = "Объект.Наименование";
Сообщение.Сообщить();

```

**УникальныйИдентификатор** – уникальный идентификатор формы, в которой будет выведено сообщение.

Сообщение может быть привязано как к реквизитам объекта, так и к остальным реквизитам формы.

**Вариант 4. Совмещение второго и третьего вариантов. Вывод сообщения не в текущую форму, с привязкой к реквизиту в третьей форме**

```

ПараметрыФормы = Новый Структура;
ПараметрыФормы.Вставить("Ключ", Объект.Контрагент);
Форма = ОткрытьФорму("Справочник.Контрагенты.ФормаОбъекта",
ПараметрыФормы, ЭтаФорма);

```

```

Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "Текст сообщения";
Сообщение.ИдентификаторНазначения =
Форма.УникальныйИдентификатор;
Сообщение.Поле = "Наименование";

```

Сообщение.КлючДанных = Объект.Договор;  
 Сообщение.ПутьКДанным = "Объект";  
 Сообщение.Сообщить();

В результате вызова сообщения из формы документа, оно будет выведено в форме контрагента. При нажатии на текст сообщения, будет открыта форма договора, и сообщение будет привязано к реквизиту договора.

### Вариант 5. Вывод сообщения из модуля объекта

Для вывода сообщения из модуля объекта, можно использовать следующий синтаксис:

Сообщение = Новый СообщениеПользователю;  
 Сообщение.Текст = "Текст сообщения";  
 Сообщение.Поле = "Наименование";  
 Сообщение.КлючДанных = Ссылка;  
 Сообщение.ПутьКДанным = "Объект";  
 Сообщение.Сообщить();

Сообщение будет выведено и привязано к реквизитам корректно, причем параметры: «КлючДанных» и «ПутьКДанным» можно опустить, указав в параметре «Поле» полный путь к данным формы.

**Важно.** Без указания параметров: «КлючДанных» и «ПутьКДанным» сообщение будет привязано к реквизитам формы только в том случае, если форма объекта открыта и является активной. Однако, код процедуры модуля объекта может быть выполнен и без открытия формы. В этом случае сообщение будет выведено в текущей активной форме. И, чтобы при нажатии на тексте сообщения, открывалась форма объекта с привязкой сообщения к реквизитам формы (например, при проведении документа из формы списка), должны быть заполнены параметры: «КлючДанных» и «ПутьКДанным». Параметр «Поле» в этом случае, должен содержать наименование реквизита объекта.

Также, как и на клиенте, есть возможность привязать сообщение к реквизитам формы объекта, который не является текущим. Для этого необходимо присвоить параметру «КлючДанных» ссылку на объект:

Сообщение.КлючДанных = ОсновнойДоговор;

В этом случае, будет открыта новая форма, по указанному ключу (ссылке) и сообщение будет привязано к ее реквизитам.

Однако, в модуле объекта, более корректно использовать метод объекта «УстановитьДанные» с указанием объекта, к которому требуется привязать сообщение:

Сообщение = Новый СообщениеПользователю;

Сообщение.Текст = "Текст сообщения";

Сообщение.Поле = "Наименование";

Сообщение.УстановитьДанные(ЭтотОбъект);

Сообщение.Сообщить();

По своей сути, данный метод устанавливает значения для параметров: «КлючДанных» и «ПутьКДанным». Ключу данных будет присвоена ссылка на объект, а вот установка параметра «ПутьКДанным» происходит несколько сложнее. Дело в том, что работа с объектом может производиться из разных форм. Имя основного реквизита формы, в общем случае, может отличаться от стандартного «Объект». Метод «УстановитьДанные» определяет имя основного реквизита формы по установленному соответствию. Причем, при записи объекта это соответствие устанавливается автоматически. В тех случаях, когда автоматического сопоставления нет, его необходимо указать явно с помощью метода «УстановитьСоответствиеОбъектаИРеквизитаФормы».

Например, необходимо выполнить какие-либо действия с объектом, привязанным к текущей форме, и вывести в процессе этих действий сообщение:

//Серверная процедура в модуле формы

&НаСервере

Процедура ОбработкаОбъекта ()

    КонтрагентОбъект = ДанныеФормыВЗначение (Объект,  
    Тип("СправочникОбъект.Контрагенты"));

        УстановитьСоответствиеОбъектаИРеквизитаФормы  
    (КонтрагентОбъект, "Объект");

    КонтрагентОбъект.ВыполнитьРаботуСОбъектом();

КонецПроцедуры

//Процедура в модуле объекта

## Процедура ВыполнитьРаботуСОбъектом () Экспорт

.....

Сообщение = Новый СообщениеПользователю;

Сообщение.Текст = "Текст сообщения";

Сообщение.Поле = "Наименование";

Сообщение.УстановитьДанные(ЭтотОбъект);

Сообщение.Сообщить();

## КонецПроцедуры

### Оповещение пользователя без привязки к конкретной форме

Следует иметь в виду, что сообщения, выведенные командой «Сообщить» или через объект «СообщениеПользователю» **всегда привязаны к какой-либо форме**. Если после вывода сообщения форма закрывается, пользователь не сможет его прочитать. В данном случае удобнее использовать другие методы глобального контекста: «ПоказатьОповещениеПользователю» или «Состояние». **Вызов методов возможен только на клиенте.**

Метод «ПоказатьОповещениеПользователю» выводит всплывающее оповещение в нижний часть экрана, которое не привязано к какой-либо открытой форме:

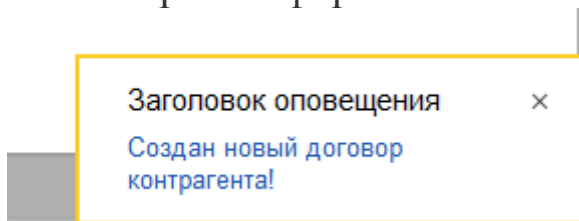


Рисунок 15.1 – Оповещение

Синтаксис метода в справке:

ПоказатьОповещениеПользователя(<Текст>,  
 <ДействиеПриНажатии>,  
 <Пояснение>,  
 <Картинка>,  
 <СтатусОповещенияПользователя>,  
 <КлючУникальности>)

Первый параметр «Текст» вводит в заблуждение, на самом деле – это заголовок окна оповещения.

Второй параметр *«ДействиеПриНажатии»* может содержать либо навигационную ссылку, по которой необходимо перейти при нажатии на сообщении, либо описание процедуры оповещения.

Параметр *«Пояснение»* - это как раз текст оповещения.

Дополнительно, есть возможность вывести картинку, указать важность оповещения и задать ключ уникальности для поиска уже открытых оповещений.

**Примечание.** *Следует использовать данный метод для информационных, не особенно важных сообщения, так как они исчезают самостоятельно, без каких-либо действий со стороны пользователя.*

Команда *«Состояние»* выводит информацию в специальную панель состояния. Ее следует использовать для информирования пользователя о ходе выполнения кого-либо действия. Дополнительно есть возможность показать прогресс выполнения:

**Состояние** (*«Выполнение операции, 50, «Операция выполняется»*);

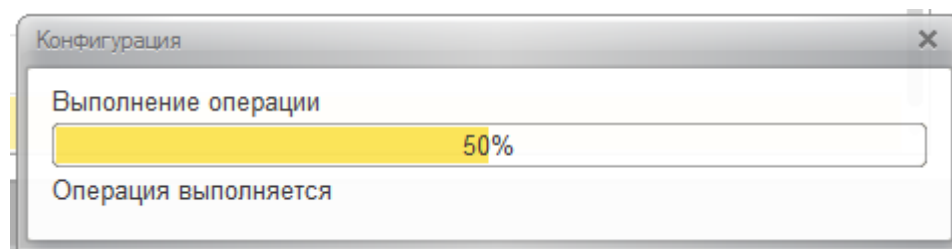


Рисунок 15.2 – Состояние операции

### Вывод сообщения пользователю в отдельном окне

В некоторых случаях требуется вывести сообщение пользователю в отдельном окне. Для этой цели можно использовать как собственную разработанную форму, так и не модальный метод глобального контекста *«ПоказатьПредупреждение»*. (В отличие от модальных методов, данный не ожидает выполнения действия с формой, а продолжает выполнение программного кода). **Вызов метода возможен только на клиенте.**

В качестве первого параметра необходимо указать процедуру - описание оповещения. В качестве второго – текст выводимого сообщения. Причем, в качестве текста сообщения можно

использовать форматированную строку. Это позволяет вывести в тексте сообщения ссылку на какой-либо объект:

```
СтрокаСообщения = Новый ФорматированнаяСтрока
(НовыйДоговор.Наименование,,,,
ПолучитьНавигационнуюСсылку (НовыйДоговор.Ссылка));
ТекстСообщения = Новый ФорматированнаяСтрока ("Создан новый
договор: """, СтрокаСообщения, """);
ОписаниеОповещения = новый ОписаниеОповещения
("ПредупреждениеЗавершение", ЭтаФорма);
ПоказатьПредупреждение (ОписаниеОповещения, ТекстСообщения,,
"Заголовок сообщения");
```

&НаКлиенте

```
Процедура ПредупреждениеЗавершение (Параметры) Экспорт
    //Обработка закрытия предупреждения
КонецПроцедуры
```

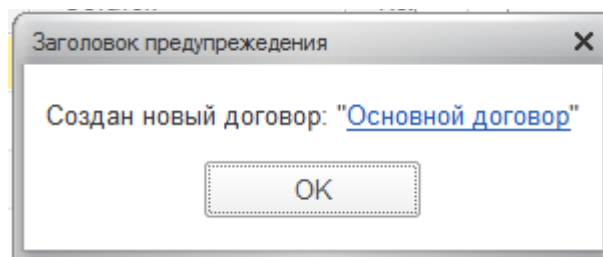


Рисунок 15.3 – Предупреждение пользователю

### Задание

1. Создать и вывести сообщение пользователю 5 различными вариантами.
2. Создать и вывести оповещение пользователя без привязки к конкретной форме.
3. Создать и вывести сообщение пользователю в отдельном окне.
4. В отчете представить примеры кода и результат выполнения работы с описанием.

### Контрольные вопросы

1. Что представляет собой объект «СообщениеПользователю»?
2. Какие есть варианты использования объекта «СообщениеПользователю»?
3. Как реализовать оповещение пользователя без привязки к конкретной форме?
4. Как реализовать вывод сообщения пользователю в отдельном окне?



## ПРАКТИЧЕСКОЕ ЗАНЯТИЕ №12. ФОРМИРОВАНИЕ ПРЕДУПРЕЖДЕНИЙ И ВОПРОСОВ ПОЛЬЗОВАТЕЛЮ

При разработке конфигурации на платформе 1С:Предприятие 8 периодически возникает потребность приостановить работу программы до того момента, когда пользователь примет какое-либо решение или выполнит какие-либо действия.

Например, при нажатии на кнопку заполнения табличной части у пользователя следует спросить, нужно ли очистить табличную часть, чтобы не произошло потери ранее введенных данных.

Такое поведение может обеспечить, например, следующий код:

&НаКлиенте

Процедура ЗаполнитьТовары (Команда)

Ответ = Вопрос (“Табличная часть будет очищена.

Продолжить?”, РежимДиалогаВопрос.ДаНет );

Если Ответ = КодВозвратаДиалога.Да Тогда

//алгоритм заполнения

КонецЕсли;

КонецПроцедуры

В результате работы этого фрагмента кода произойдет приостановка выполнения программного кода, на экране отображается вопрос, интерфейс приложения кроме диалога с вопросом становится недоступным, система ожидает принятия решения пользователем, выполнение кода продолжится только после ответа на вопрос.

Также к приостановке выполнения кода и блокировке интерфейса приводит открытие модальных окон при помощи вызова метода ОткрытьМодально().

При работе с конфигурацией в режиме веб-клиента через браузер в этом случае будет открыто новое окно – всплывающее окно, которое заблокирует не только текущую вкладку, но и весь интерфейс браузера, включая остальные открытые окна и вкладки.

Всплывающие окна в Интернете зачастую используются для злоумышленного распространения нежелательной рекламы, поэтому браузеры содержат функции блокировки всплывающих окон.

В таком случае для работы с конфигурациями 1С:Предприятие 8 через браузер необходимо запретить блокирование всплывающих окон.

Проблемы также возникают при работе на мобильных устройствах. Так, например, модальные окна не поддерживаются на iPad.

Для решения указанных проблем следует использовать блокирующие окна вместо модальных. Для пользователя визуально все выглядит так же: окно блокирует интерфейс веб-клиента.

Однако блокирующее окно как бы “рисует” поверх главного окна, и блокируется только текущая вкладка браузера, в которой открыта конфигурация, позволяя переключаться на другие вкладки, поскольку модальные окна браузера при этом не используются.

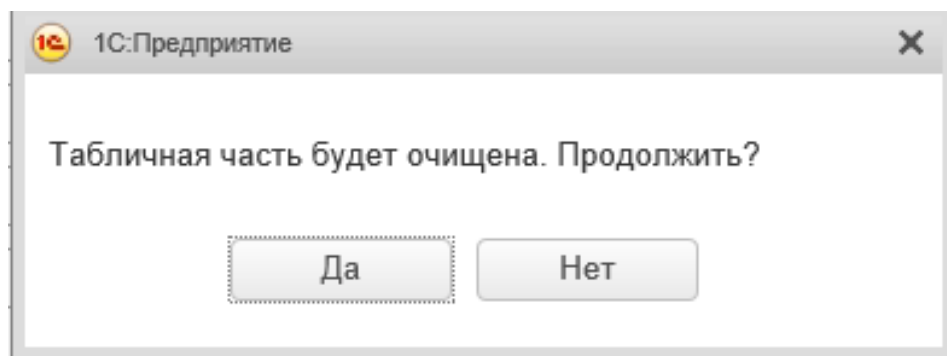


Рисунок 16 - Блокирующее окно

Таким образом, всплывающие окна в браузере не открываются и обеспечивается работа через веб-клиент на мобильных устройствах.

У корневого элемента конфигурации существует свойство “Режим использования модальности”, которое определяет, можно ли в конфигурации открывать модальные окна.

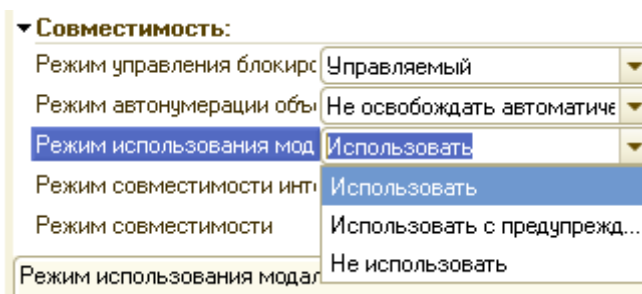


Рисунок 16.1 - Режим использования модальности

Если выбран вариант “Использовать”, то модальные окна можно открывать. Если выбран вариант “Не использовать”, то модальные окна недопустимы. При попытке вызвать метод, открывающий модальное окно, система выводит сообщение об ошибке:

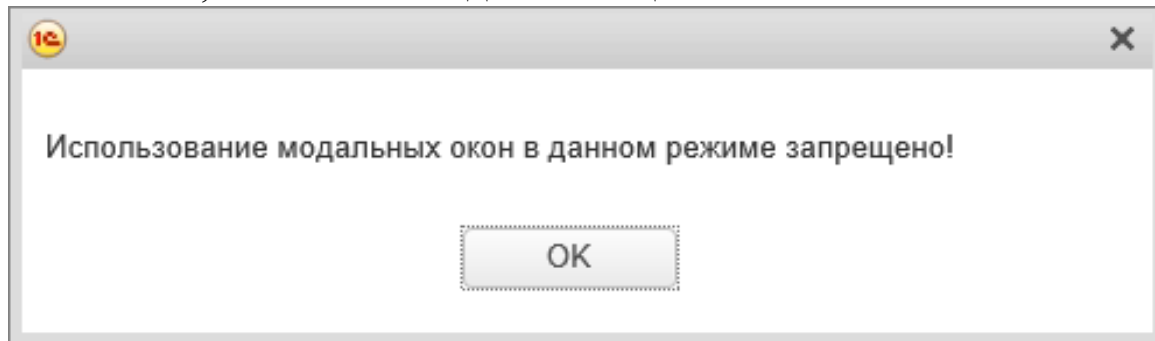


Рисунок 16.2 - Вывод сообщения об ошибке

При таком значении свойства “Режим использования модальности” допустимы только блокирующие окна.

Если выбран вариант “Использовать с предупреждениями”, то при открытии модальных окон в окно сообщений выводится текст:

Такой вариант работы может использоваться как промежуточный при переработке конфигурации с целью отказа от использования модальных окон.

Основное отличие блокирующих окон от модальных заключается в том, что открытие блокирующего окна не производит приостановки выполнения кода.

Поэтому разработчикам придется переписать программный код, использующий модальные окна, с учетом этой особенности.

Код нужно разделить на две части:

- открытие блокирующего окна;
- обработка выбора пользователя.

Фрагмент кода, приведенный в начале статьи, нужно переписать следующим образом:

&НаКлиенте

Процедура ЗаполнитьТовары (Команда)

Оповещение = Новый ОписаниеОповещения (, ЭтотОбъект);

РежимДиалогаВопрос.ДаНет );

КонецПроцедуры

&НаКлиенте

Процедура (Результат, ДополнительныеПараметры) Экспорт

Если Результат = КодВозвратаДиалога.Да Тогда

//алгоритм заполнения

КонецЕсли;

КонецПроцедуры

После выполнения процедуры ПоказатьВопрос() система не останавливается, ожидая ответ пользователя, исполнение кода продолжается.

Пользователь сможет сделать выбор только после завершения работы всей процедуры. При этом будет вызвана экспортная процедура ЗаполнитьТоварыВопросЗавершение(). Ее название мы передали в конструктор объекта ОписаниеОповещения.

Процедура, которая будет вызвана после осуществления выбора, может быть расположена в модуле формы, модуле команды, общем не глобальном модуле.

В рассмотренном примере вызываемая процедура расположена в модуле управляемой формы, поэтому мы передали в параметр ЭтотОбъект.

Рассмотрим вызов процедуры, расположенной в общем модуле. Для этого добавим новый общий модуль ОбработкаОповещений, установим для него флаг “Клиент (управляемое приложение)”, а признак “Глобальный” не устанавливаем. Расположим в этом модуле процедуру ЗаполнитьТоварыВопросЗавершение().

Тогда обработчик команды заполнения будет выглядеть так:

&НаКлиенте

Процедура ЗаполнитьТовары (Команда)

Оповещение =

Новый ОписаниеОповещения (“ЗаполнитьТоварыВопросЗавершение”, ОбработкаОповещений );

ТекстВопроса = “Табличная часть будет очищена. Продолжить?” ;

ПоказатьВопрос (Оповещение,

ТекстВопроса, РежимДиалогаВопрос.ДаНет );

КонецПроцедуры

После вызова любого метода, открывающего блокирующее окно, процедура должна завершаться, а выполняемый далее код следует

располагать в процедуре, которая будет вызвана после закрытия окна.

Для передачи контекста (вспомогательных данных, неких параметров, значений переменных) из процедуры, открывающей модальное окно, в процедуру, вызывающуюся при его закрытии, предусмотрен третий необязательный параметр конструктора объекта ОписаниеОповещения – ДополнительныеПараметры.

Этот объект (любого типа) будет передан в процедуру, описанную в ОписаниеОповещения, последним параметром.

На примере рассмотренного выше участка кода это можно сделать так:

&НаКлиенте

Процедура ЗаполнитьТовары (Команда)

Параметр1 = 0;

Параметр2 = 0;

СписокПараметров = Новый Структура (“Параметр1, Параметр2”,  
Параметр1, Параметр2);

Оповещение =

Новый ОписаниеОповещения (“ЗаполнитьТоварыВопросЗавершени  
е”, ЭтотОбъект,

СписокПараметров);

ПоказатьВопрос (Оповещение, “Табличная часть будет очищена.

Продолжить?”, РежимДиалогаВопрос.ДаНет);

КонецПроцедуры

&НаКлиенте

Процедура ЗаполнитьТоварыВопросЗавершение (Результат, Дополни  
тельныеПараметры) Экспорт

Если Результат = КодВозвратаДиалога.Да Тогда

//анализируем ДополнительныеПараметры.Параметр1

//анализируем ДополнительныеПараметры.Параметр2

КонецЕсли;

КонецПроцедуры

Если нужно передать только одно значение, то структуру можно не использовать, а присвоить это значение параметру

ДополнительныеПараметры конструктора объекта  
ОписаниеОповещения.

Рассмотрим несколько примеров работы с блокирующими окнами.

### Задача 1. Открытие другой формы

*Из формы документа по нажатию на кнопку “Открыть параметры” нужно открыть форму, на которой расположены два флажка Параметр1 и Параметр2, которые должен установить пользователь. После закрытия формы вывести в строку сообщений значения параметров.*

Создаем общую форму “ФормаПараметров”, на которой размещаем реквизиты Параметр1 и Параметр2, а также команду ЗакрытьФорму:

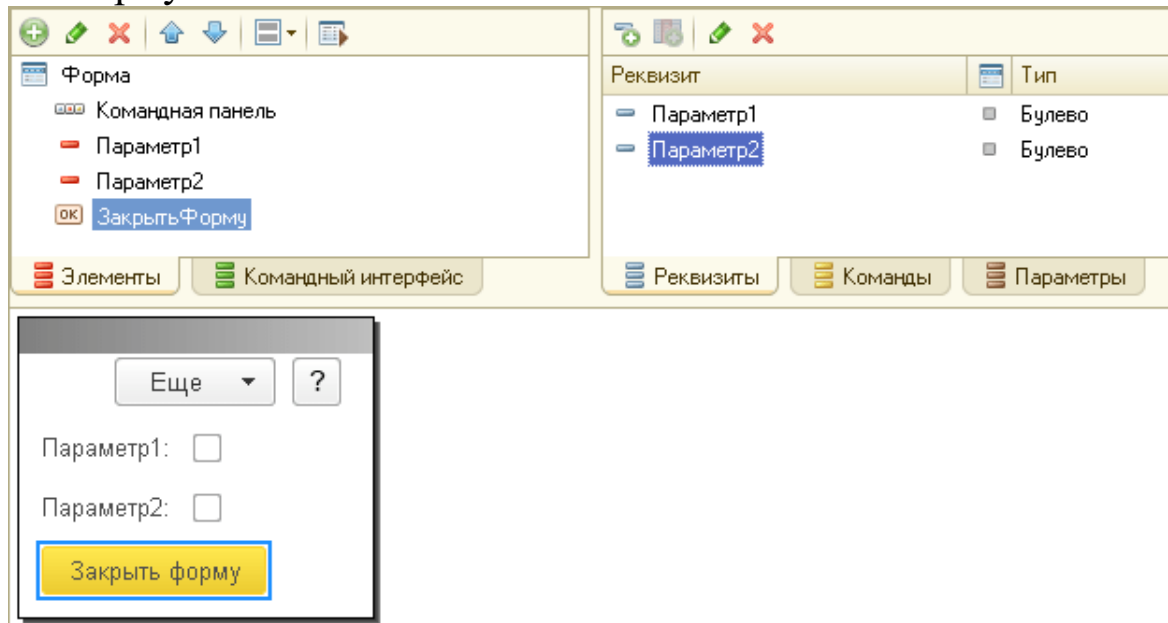


Рисунок 16.3 - Общая форма с реквизитами и параметрами

Обработчик команды выглядит следующим образом:

Обработчик команды выглядит следующим образом:

&НаКлиенте

Процедура ЗакрытьФорму (Команда)

СписокПараметров = Новый Структура (“Параметр1, Параметр2”,  
Параметр1, Параметр2);

Закрыть (СписокПараметров);

КонецПроцедуры

Для формы свойство РежимОткрытияОкна устанавливаем в “Блокировать весь интерфейс”:

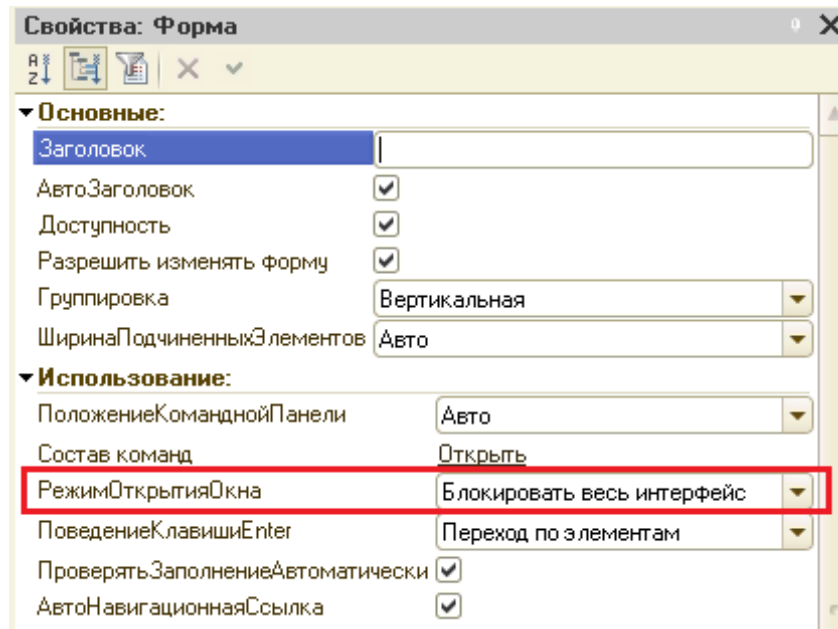


Рисунок 16.4 - Свойство РежимОткрытияОкна

На форме документа располагаем команду ОткрытьПараметры, обработчик которой описываем следующим образом:

&НаКлиенте

Процедура ОткрытьПараметры (Команда )

Оповещение =

Новый ОписаниеОповещения (“ОткрытьПараметрыЗавершение” ,  
ЭтотОбъект );

ОткрытьФорму (“ОбщаяФорма.ФормаПараметров” , , , , ,  
Оповещение );

КонецПроцедуры

&НаКлиенте

Процедура ОткрытьПараметрыЗавершение (Результат, Дополни-  
тельныеПараметры) Экспорт

Если ТипЗнч (Результат ) = Тип (“Структура” ) Тогда

Для каждого КлючЗначение Из Результат Цикл

Сообщение = Новый СообщениеПользователю ;

Сообщение.Текст = “Ключ: “” ” + КлючЗначение.Ключ + “”” ,  
значение = ” + КлючЗначение.Значение ;

Сообщение.Сообщить ();

КонецЦикла ;

КонецЕсли ;

КонецПроцедуры

В пользовательском режиме, запуская конфигурацию под веб-клиентом, получаем такие результаты работы:

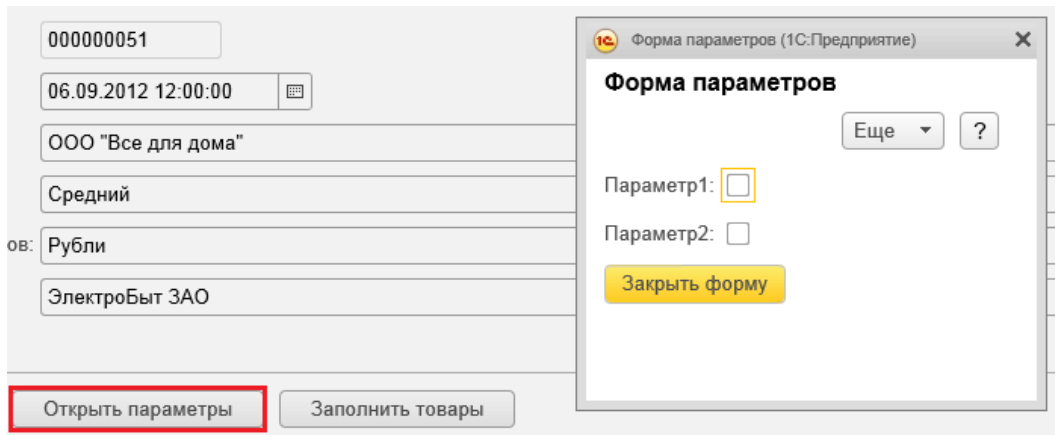


Рисунок 16.5 – Результат работы

*Для увеличения нажмите на изображение.*

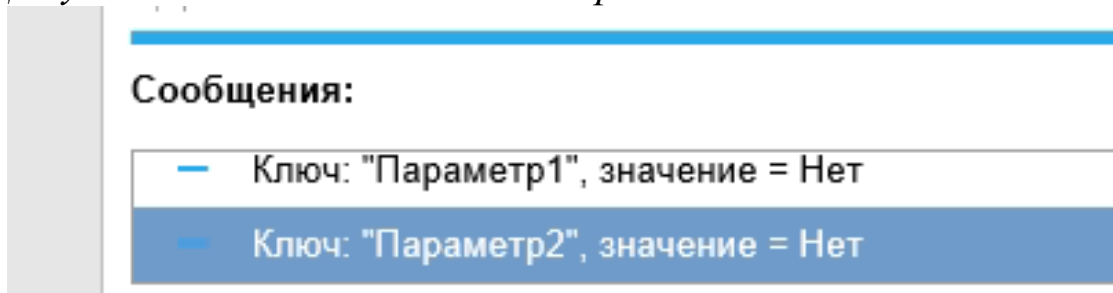


Рисунок 16.6 – Результат увеличения

Режим открытия окна можно также указывать в последнем параметре процедуры ОткрытьФорму.

&НаКлиенте

Процедура ОткрытьПараметры (Команда)

Оповещение =

Новый ОписаниеОповещения (“ОткрытьПараметрыЗавершение”, ЭтотОбъект );

ОткрытьФорму (“ОбщаяФорма.ФормаПараметров”, , , , ,

Оповещение ,



РежимОткрытияОкнаФормы.БлокироватьВесьИнтерфейс);  
КонецПроцедуры

## **Задача 2. Вопрос при закрытии формы**

*При закрытии окна обработки задавать пользователю вопрос, действительно ли он хочет закрыть окно.*

Эту задачу можно решить при помощи следующего кода, расположенного в модуле формы обработки:

```
&НаКлиенте
Перем НужноЗакрыватьФорму;
&НаКлиенте
Процедура ПередЗакрытием (Отказ, СтандартнаяОбработка)
Если НЕ НужноЗакрыватьФорму = Истина Тогда
Отказ = Истина;
Оповещение = Новый ОписаниеОповещения (“ПередЗакрытием
Завершение”, ЭтотОбъект );
ПоказатьВопрос (Оповещение, “Вы действительно хотите закрыть
окно?” ,
РежимДиалогаВопрос.ДаНет );
КонецЕсли;
КонецПроцедуры
```

```
&НаКлиенте
Процедура ПередЗакрытиемЗавершение (Результат, Дополнительные
Параметры) Экспорт
Если Результат = КодВозвратаДиалога.Да Тогда
НужноЗакрыватьФорму = Истина;
Закрыть ();
Иначе
НужноЗакрыватьФорму = Неопределено;
КонецЕсли;
КонецПроцедуры
```

В процедуре ПередЗакрытием формы пользователю задается вопрос, флаг Отказ выставляется в Истина, закрытие формы отменяется.

После утвердительного ответа на вопрос переменная НужноЗакрыватьФорму устанавливается в Истина, форма закрывается повторно.

### **Задача 3. Ввод числового значения**

*При нажатии на кнопку на форме обработки открывать стандартный диалог ввода числа.*

Для этого необходимо воспользоваться методом ПоказатьВводЧисла() вместо ВвестиЧисло(), который открывает блокирующее окно вместо модального.

&НаКлиенте

Процедура ВводЧисла (Команда)

Оповещение =

Новый ОписаниеОповещения (“ВводЧислаЗавершение”,

ЭтотОбъект);

ПоказатьВводЧисла (Оповещение, 0, “Введите количество”, 15, 3);

КонецПроцедуры

&НаКлиенте

Процедура ВводЧислаЗавершение (Результат, ДополнительныеПараметры) Экспорт

Сообщение = Новый СообщениеПользователю;

Сообщение.Текст = “Вы ввели количество” + Результат;

Сообщение.Сообщить ();

КонецЕсли;

КонецПроцедуры

После закрытия окна ввода числа будет вызвана процедура, в первый параметр которой будет передано введенное число или значение Неопределено, если пользователь отказался от ввода.

### **Задача 4. Выбор цвета**

*При нажатии на кнопку на форме обработки при помощи стандартного диалога выбора цвета пользователь указывает необходимый цвет. Этот цвет установить для фона нажимаемой кнопки.*

Добавим на форму команду ВыборЦвета со следующим обработчиком:

&НаКлиенте

Процедура ВыборЦвета (Команда)

ДиалогВыбораЦвета = Новый ДиалогВыбораЦвета;

Оповещение =

Новый ОписаниеОповещения (“ВыборЦветаЗавершение”,  
ЭтотОбъект);

ДиалогВыбораЦвета. Показать (Оповещение);

КонецПроцедуры

&НаКлиенте

Процедура ВыборЦветаЗавершение (Результат, ДополнительныеПар  
аметры) Экспорт

Если НЕ Результат = Неопределено Тогда

Элементы.ВыборЦвета.ЦветФона = Результат;

КонецЕсли;

КонецПроцедуры

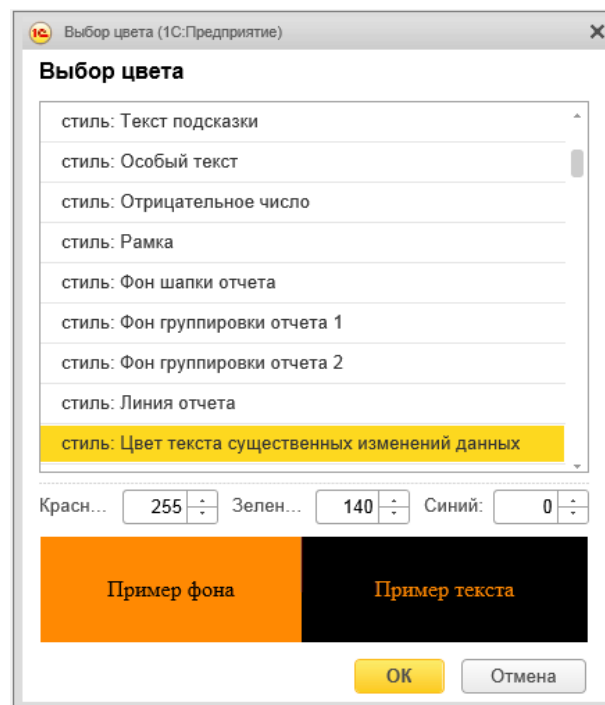


Рисунок 16.7 – Выбор цвета кнопки

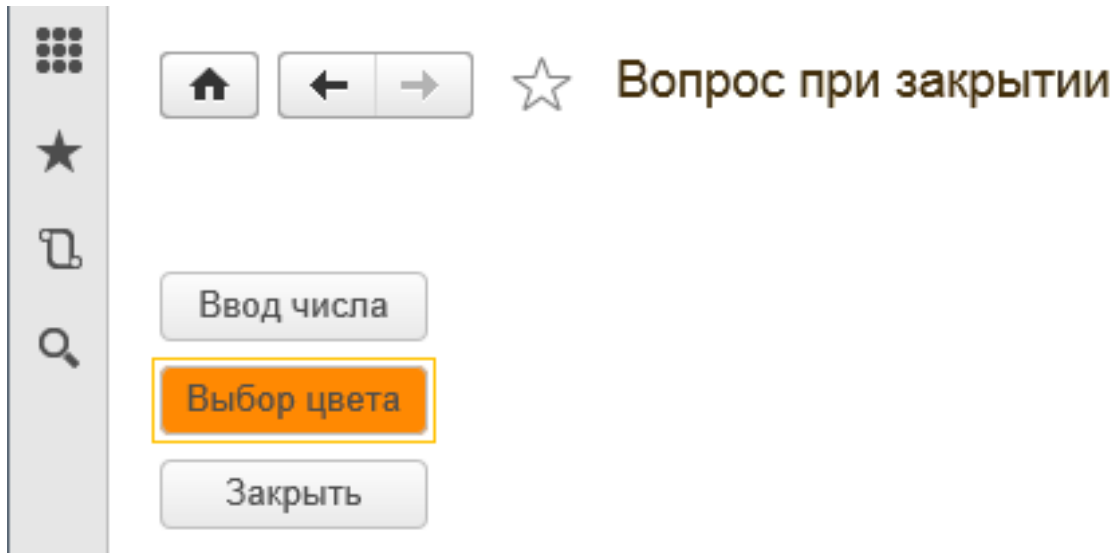


Рисунок 16.8 – Результат применения параметра

Для объектов `ДиалогВыбораЦвета` (а также `ДиалогРедактированияСтандартногоПериода`, `КонструкторФорматнойСтроки`, `ДиалогРасписанияРегламентногоЗадания`, `ДиалогВыбораШрифта`) метод `Показать()` открывает блокирующее окно.

После закрытия окна будет вызвана процедура, в первый параметр которой будет передано выбранное значение (цвет, шрифт и т.д.) или значение `Неопределено`, если пользователь отказался от выбора.

Следует обратить внимание, что объект `ДиалогВыбораФайла` не имеет метода `Показать()` в отличие от диалогов выбора цвета или шрифта, поскольку реализация этих диалогов существенно разная.

Для использования диалога выбора файла на веб-клиенте необходимо предварительно подключить расширение работы с файлами.

Диалоги, реализуемые через расширение работы с файлами, не создают таких проблем в работе, как модальные окна браузеров, поэтому не было реализовано открытие блокирующих окон для объекта `ДиалогВыбораФайла`.

### Задание

1. Выполнить задачи под номерами 1-4 данной практической работы.

2. В отчете должно быть отображено выполнение каждой задачи скриншоты кода и результат работы программы с описанием.

### **Контрольные вопросы**

1. Как реализовать открытие другой формы?
2. Как реализовать вопрос пользователю при закрытии формы?
3. Как вывести числовое значение?
4. Как произвести выбор цвета для конкретного объекта?

## ЛАБОРАТОРНАЯ РАБОТА №6. РАБОТА С ОТЧЁТАМИ

Отчеты – это то, к чему в итоге сводится деятельность организации. Необходимо получать итоговые отчеты о том, сколько и какого товара было продано за определенный период и на какую сумму; какую выручку получила организация; кто из сотрудников, сколько договоров заключил и какую прибыль принес; какие контрагенты наиболее часто работают с организацией; сколько товара есть на складах на данный момент; и т.д. Такая сводная, итоговая информация хранится в регистрах (сведений, накоплений и др.), а, если точнее, в виртуальных таблицах, формируемых на основании того, какие именно сгруппированные данные необходимо получить в отчете. Основным инструментом формирования отчетов служат запросы.

В нынешних версиях «1С: Предприятия» для создания отчетов используется специальный инструмент «Система компоновки данных» (СКД).

Создадим новый отчет «**Остатки товаров**», включим в его подсистему «**Отдел закупок**». На вкладке «Основные» окна настройки свойств выберем команду «Открыть схему компоновки данных» и нажмем «Готово». В результате откроется пустая схема компоновки данных, включающая в себя наборы данных для отчета и его настройки (Рисунок 17).

Изначально необходимо при помощи запроса сформировать данные, выводимые в отчет. Для этого добавляем очередной набор данных типа «запрос» (Рисунок 17). В результате формируется пустой набор данных, для которого с помощью специальной кнопки запускаем конструктор запроса.

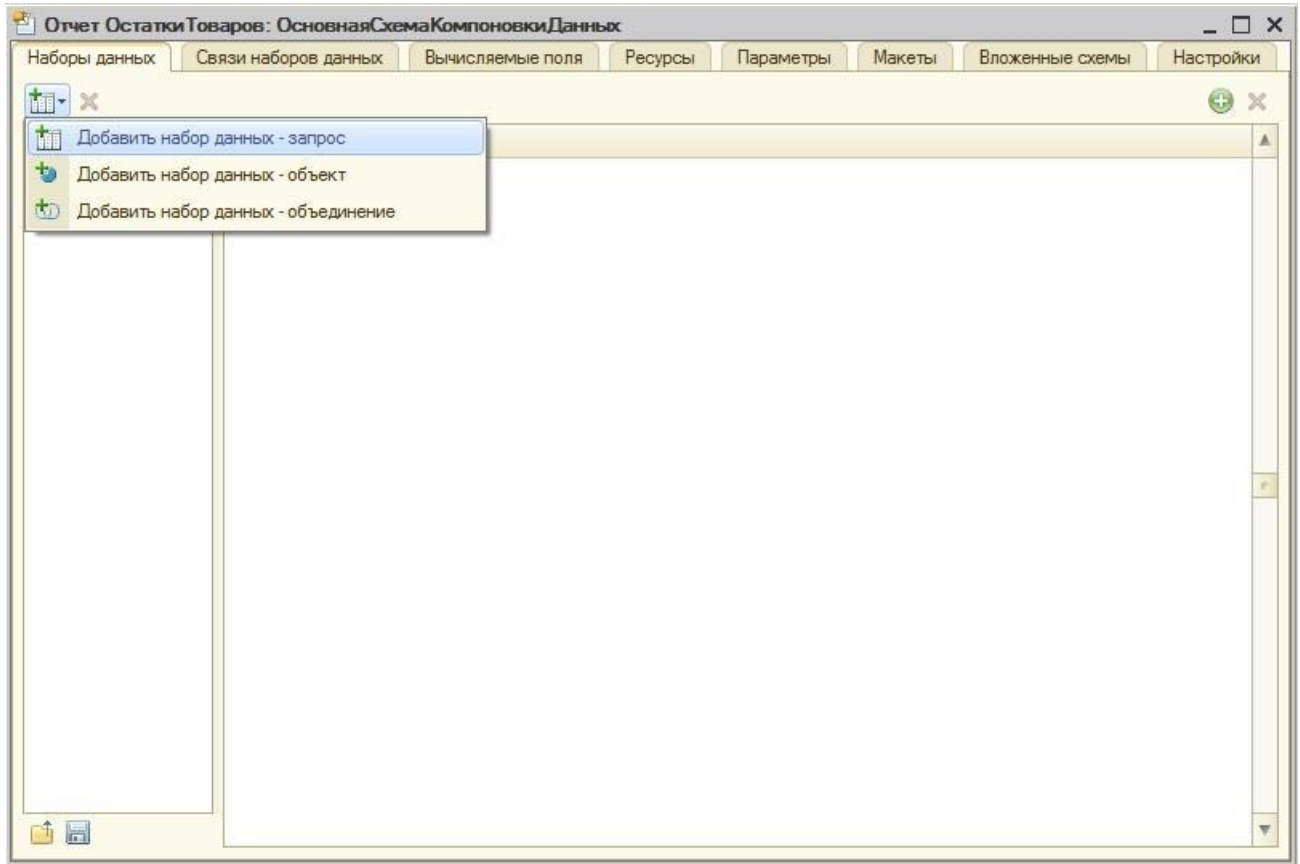


Рисунок 17 - Система компоновки данных. Добавление набора данных

Для получения остатков товаров, как и ранее, используем виртуальную таблицу **«ОстаткиТоваровОстатки»**. В данной таблице содержится информация только о тех товарах, которые имеются на складе. Если мы хотим вывести остатки по всей номенклатуре, то необходимо с данной виртуальной таблицей левым соединением связать таблицу – справочник **«Номенклатура»**. С учетом того, что в базе реализован учет различных вариаций одного и того же товара, то для получения остатков по вариантам товаров, еще одним источником данных будет справочник **«Варианты номенклатуры»** (который в свою очередь нужно связать со справочником-владельцем – **«Номенклатура»**). В запросе необходимо вывести все элементы справочников **«Номенклатура»** и **«Варианты номенклатуры»**, с учетом взаимосвязей между ними, а для тех комбинаций **«Номенклатура+Вариант»**, для которых есть остатки – выведем количество имеющегося товара из виртуальной таблицы.

В указанном запросе мы добавили таблицу «**Номенклатура**» (справочник), и в то же время в качестве измерения второго источника (виртуальной таблицы «**Остатки**») также фигурирует «**Номенклатура**». Данная ситуация с точки зрения конструктора может вызвать неоднозначную интерпретацию того, к чему идет обращение: к справочнику или к измерению регистра. Для исключения конфликта, справочник-источник данных необходимо переименовать, допустим, в «**спрНоменклатура**».

Далее из «**спрНоменклатура**» выбираем поле «**Наименование**», из справочника «**Варианты номенклатуры**» – тоже «**Наименование**», а из виртуальной таблицы остатков – «**КоличествоОстаток**» и «**СтоимостьОстаток**». Переходим на закладку связи и организуем следующие левые соединения (Рисунок 17.1):

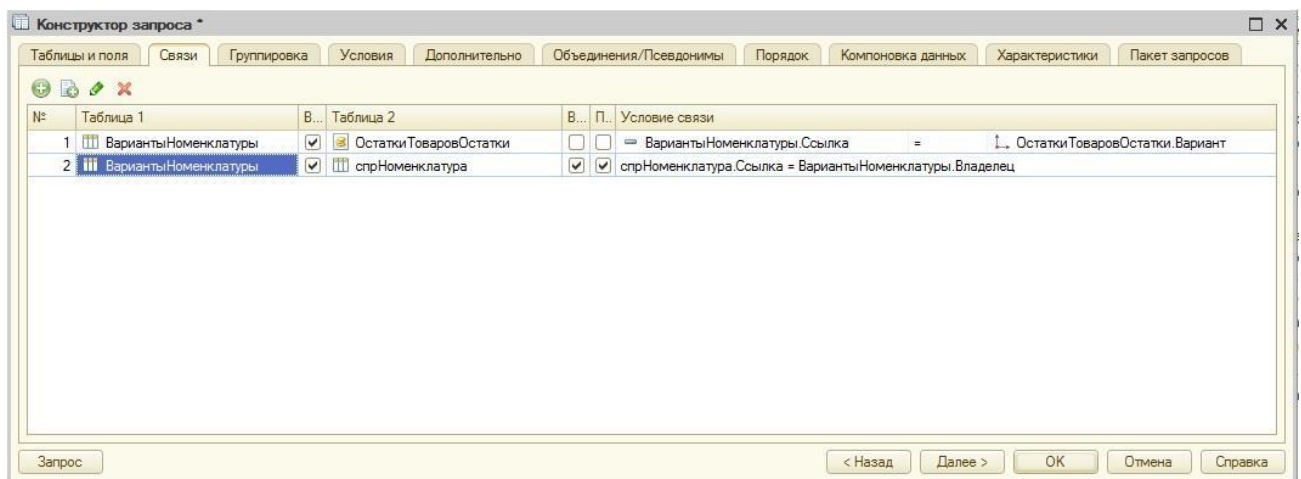


Рисунок 17.1 - Связи между таблицами - источниками данных в отчете «**Остатки товаров**»

Поясним описанные связи. Первая связь – левое соединение таблицы «**Варианты номенклатуры**» и «**ОстаткиТоваровОстатки**»: выбираются все возможные вариации номенклатурных позиций, а для тех из них, для которых есть соответствие в виртуальной таблице, присоединяются соответствующие записи. После введения ПВХ и возможности ведения учета товаров в разрезе вариантов, в регистре накопления хранится информация по каждому варианту, а не по номенклатуре в целом. Вторая связь – это полное внутреннее соединение таблиц



«**Варианты номенклатуры**» и «**Номенклатура**»: производится поиск соответствия между всеми элементами справочников по соответствующим полям условия связи (Рисунок 17.1). Напомним, что справочник «**Варианты номенклатуры**» подчинен справочнику «**Номенклатура**». Поэтому, поле «**Владелец**» у справочника «**Варианты номенклатуры**» заполняется ссылкой на элемент справочника «**Номенклатура**». Описанная связь (Рисунок 17.1) каждому элементу подчиненного справочника находит его «**Владельца**» среди элементов справочника-владельца.

Определим ряд условий в запросе на соответствующей вкладке. Справочник «**Номенклатура**» имеет иерархию групп и элементов. Остатки товаров задаются лишь в разрезе элементов, не групп. Соответственно, из результата запроса нужно исключить записи, содержащие группы; добавляем условие:

спрНоменклатура.ЭтоГруппа = ЛОЖЬ

В справочнике «**Номенклатура**» есть как товары, так и услуги. Остатки накапливаются лишь в разрезе товаров, поэтому из результата запроса нужно исключить записи, содержащие услуги: необходимо наложить условие на поле «**Вид номенклатуры**». В конструкторе запроса данное поле перенесем из левого окна в правое. Сформированное условие будет в виде:

спрНоменклатура.ВидНоменклатуры = &ВидНоменклатуры

В тексте запроса указать конкретное значение (хоть «**Вид номенклатуры**» задается в конфигураторе) не удастся т.к. оно хранится в перечислении (а в запрос можно передавать лишь значения предопределенных элементов справочника). Поэтому **&ВидНоменклатуры** – это параметр запроса, значение которого мы заполним позже в СКД.

На вкладке «Объединения/Псевдонимы» переименуем поля «**Наименование**» в «**Номенклатура**» и «**Вариант**» для соответствующих таблиц. На этом формирование запроса завершим и вернемся к настройкам СКД (Рисунок 17.2).

В результате сформированного запроса мы получим все необходимые данные, поэтому нет необходимости в дополнительных наборах данных. На вкладке «Вычисляемые поля» (Рисунок 17.2) можно

добавить поля, которые будут рассчитываться на основе получаемых из запроса полей.

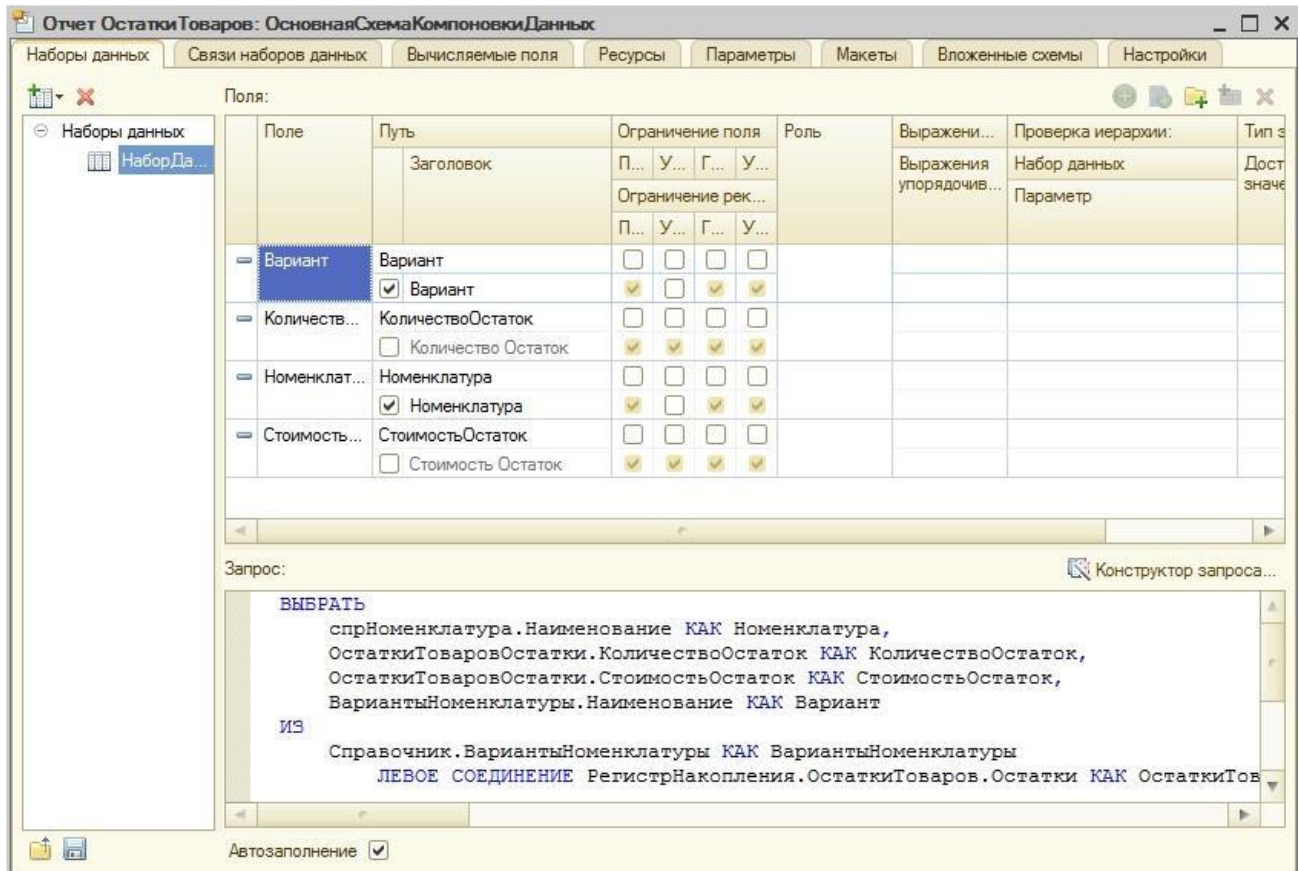


Рисунок 17.2 - Настройка СКД после формирования запроса (набора данных)

Так, необходимо, к примеру, получить среднюю себестоимость единицы каждой номенклатурной позиции. Средняя себестоимость единицы товара – **СтоимостьОстаток/КоличествоОстаток**.

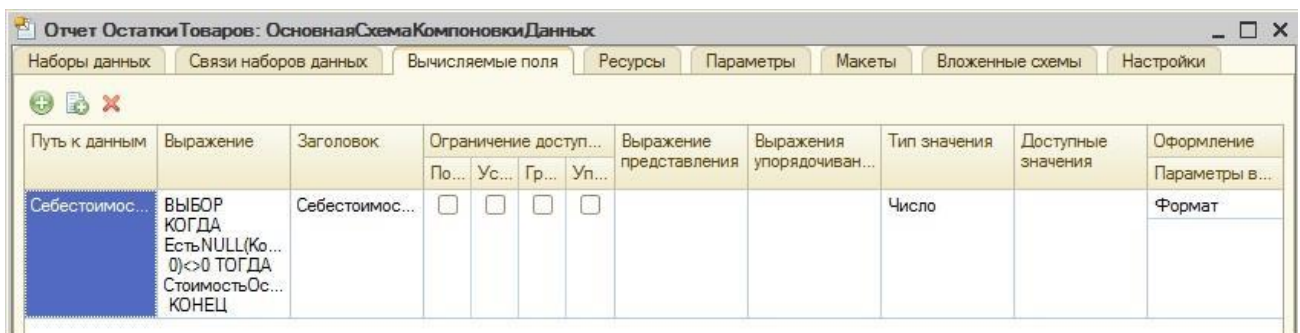


Рисунок 17.3 - Настройка вычисляемых полей

В поле «Путь данным» (Рисунок 17.3) указываем имя вычисляемого поля «**Себестоимость**».

В результате автоматически сформируется «**Заголовок**» (типа псевдонима), который при необходимости можно изменить. В поле «**Выражение**» необходимо задать выражение для расчета. Запишем следующее выражение:

**ВЫБОР**

**КОГДА** ЕстьNULL(КоличествоОстаток, 0)<>0 **ТОГДА**

СтоимостьОстаток/КоличествоОстаток

**КОНЕЦ**

Поясним данный код. Поле «**КоличествоОстаток**» может принимать значение NULL (для тех товаров, по которым остатков нет). При помощи функции *ЕстьNULL* производится приведение данного поля к заданному числовому представлению – нулю.

Также очевидно, когда остаток равен нулю, себестоимость не должна считаться; во всех остальных случаях она вычисляется по приведенной ранее формуле.

Для реализации такого условного оператора, в языке СКД (как и в языке запросов) есть специальная конструкция **ВЫБОР КОГДА...ТОГДА**, которую мы используем для проверки на неравенство нулю знаменателя («**КоличествоОстаток**») и вычисления себестоимости, когда это допустимо.

В поле «Тип значения» (Рисунок 17.3) в раскрывающемся списке выбираем: Число, длина 15, точность 2. В конце настраиваем представление данного числа. В поле «Оформление» (Рисунок 17.3) открываем список, настраивающий формат поля (Рисунок 17.4а).

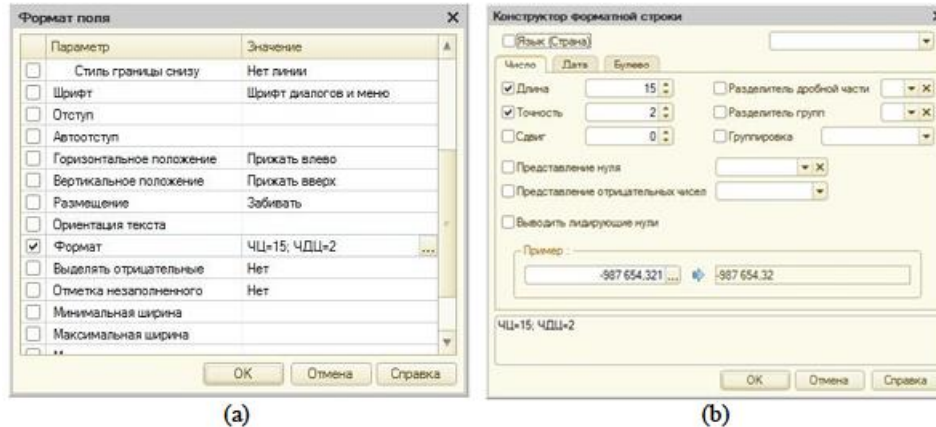


Рисунок 17.4 - Настройка формата поля (a) при помощи конструктора форматной строки (b)

Для этого выбираем параметр «Формат», и в поле «Значение» открываем конструктор форматной строки (Рисунок 17.4b). Данный конструктор в удобном режиме позволяет настроить формат выводимых данных. Производится настройка вывода числа, поэтому указываем длину 15, точность 2. После чего закрываем конструктор форматной строки и настройку формата поля.

На вкладке «Ресурсы» (Рисунок 17.2) можно добавить те поля, по которым система будет формировать Итоговые (суммируемые записи) по полям группировок (Рисунок 17.5).

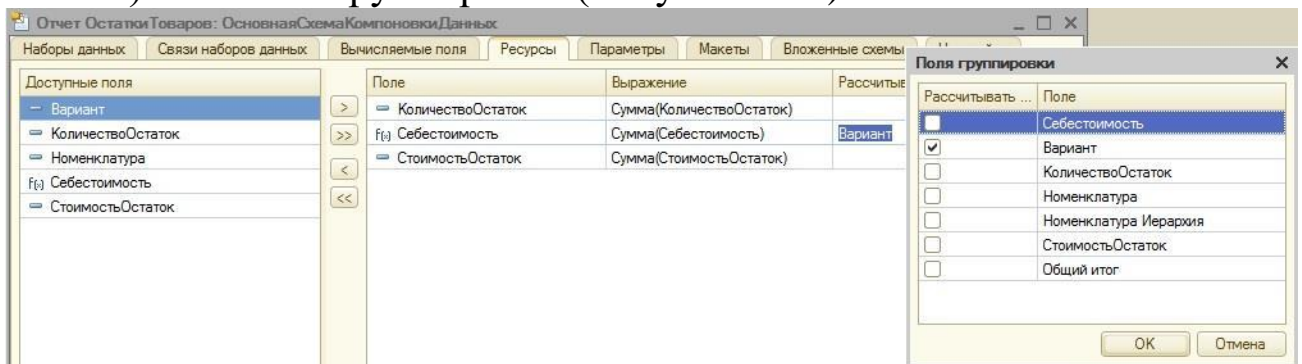


Рисунок 17.5 - Настройка ресурсов в отчете

Поля группировок будут настроены позже, но, забегая вперед отметим, что ими будет поле «**Номенклатура**»: мы получим остатки не только по каждому варианту некоторой номенклатуры, но и по номенклатуре в целом. Нажав **>>** (Рисунок 17.5), мы предоставим

системе возможность автоматически определить, какие ресурсы можно считать, т.е. все числовые поля. В поле «Рассчитывать по» вкладки «Ресурсы» для каждого ресурса можно настроить поля запроса, по которым необходимо вычислять итоги (Рисунок 17.5). По умолчанию итоги вычисляются по всем полям. В нашем примере, нет смысла считать ресурс «Себестоимость» (для единицы товара) по всей номенклатуре, или даже по каждой группе: нас будет интересовать себестоимость лишь отдельных вариантов номенклатуры. Поэтому среди всех полей отметим лишь поле «Вариант».

На вкладке «Параметры» СКД (Рисунок 17.2) настраиваются и задаются параметры, передаваемые в запрос.

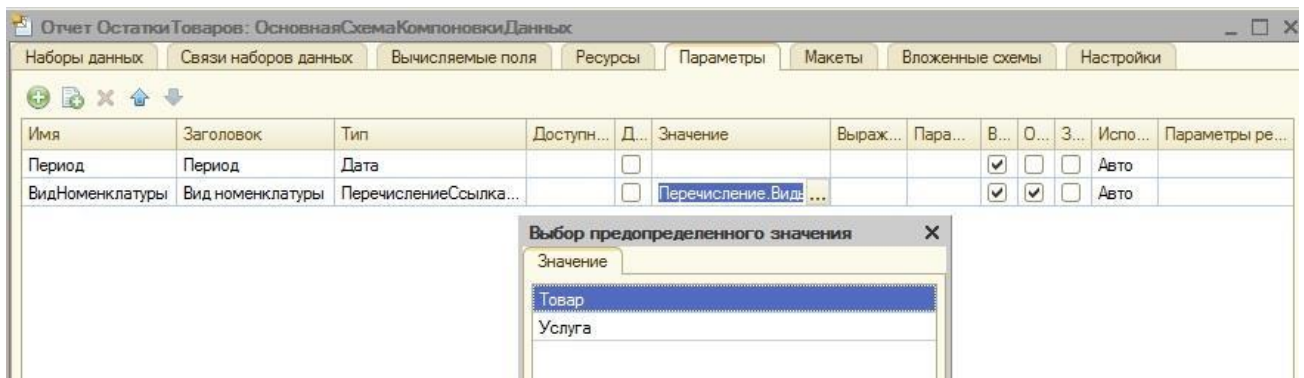


Рисунок 17.6 - Настройка параметров в СКД

Мы видим два параметра (Рисунок 17.6): «Период» и «ВидНоменклатуры». В конструкторе запроса для отчета мы не настраивали параметры виртуальной таблицы остатков неслучайно: СКД уже «знает» что такое виртуальные таблицы и как с ними работать. Поэтому при их добавлении в запрос СКД автоматически определяет параметры, особенно связанные с периодом, которые можно задать. Поэтому ряд «стандартных» параметров виртуальных таблиц можно не задавать вручную – все равно в СКД при формировании отчета они будут присутствовать.

Параметр «ВидНоменклатуры» - это параметр, в котором необходимо указать на то, что остатки необходимо определять лишь для «Товаров», «Услуги» следует исключить из результата запроса.

Для этого заполняем поле «Значение» (Рисунок 17.6) соответствующим образом. Чтобы пользователь не смог этот параметр изменить, отмечаем галочку «Ограничение доступности». Если есть необходимость, чтобы при формировании отчета пользователь мог изменять значение параметра (как, например, для «Периода»), то «Ограничение доступности» следует отключить. В представленной настройке (Рисунок 17.6), также возможно добавить свои собственные параметры, настроить их тип, значение и т.д.

На вкладке «Настройка» (Рисунок 17.2) формируется внешний вид отчета. Все, что было сделано до этого, относилось лишь к данным, которыми оперирует и которые выводятся в отчете. А то, как эти данные будут отображены в интерфейсе, устанавливается лишь при настройке отчета. Во-первых, сразу отметим, что в СКД есть возможность создать несколько типов отчета (списком, таблицей, диаграммой). Дополнительно можно сделать несколько различных вариантов (Рисунок 17.7), а в режиме исполнения выбирать нужный способ представления.

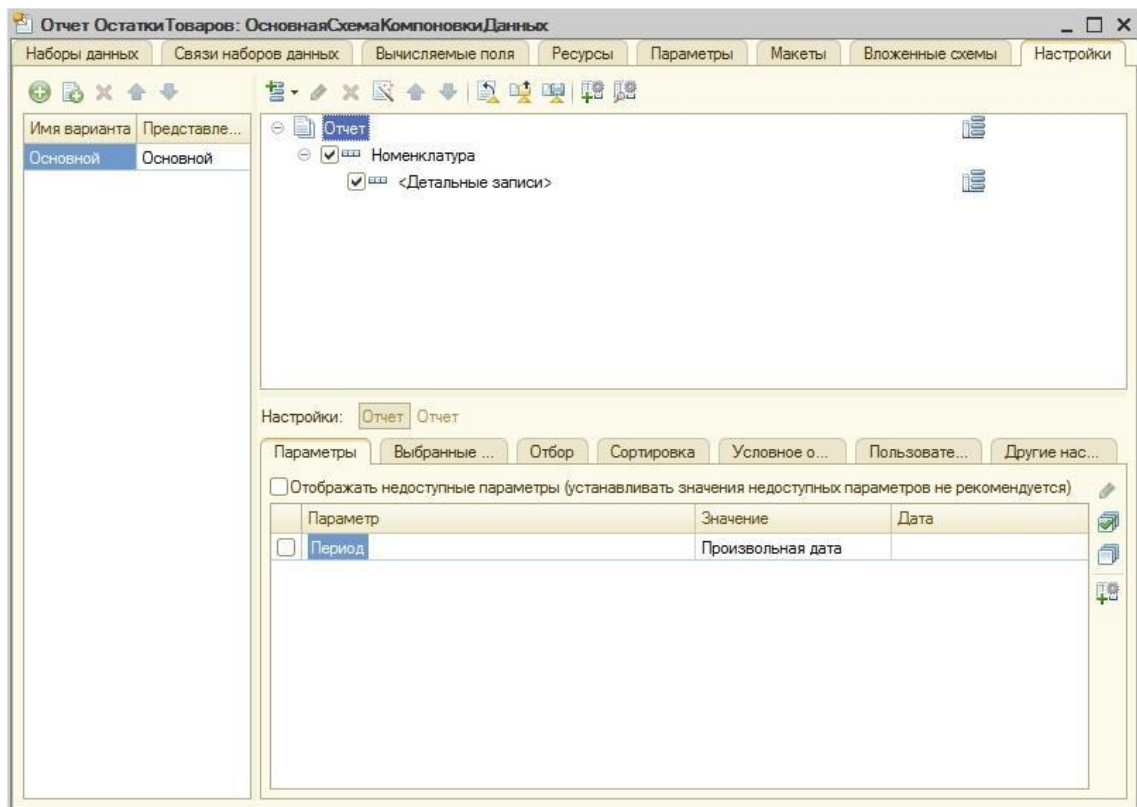




Рисунок 17.7 - Настройка вариантов отчета в СКД



Для формирования любого из вариантов, самый простой способ – использование конструктора настроек. Для этого в панели инструментов нажмем  (Рисунок 17.7). Выберем формирование отчета в виде списка. Далее укажем вывод в отчет всех полей. Далее укажем группировку по полю «**Номенклатура**», тип группировки «Иерархия». При настройке варианта отчета возможно три типа группировки: без иерархии (будут выводиться все записи без учета иерархии), иерархия (будут выводиться все записи с учетом иерархии), только иерархия (будут выводиться лишь записи верхнего уровня, т.е. только элементы-родители). На заключительной вкладке конструктора настроек не будем задавать упорядочение.

В результате сформируются настройки по умолчанию. Сформированные настройки в дальнейшем можно изменить: отредактировать выбранные поля, настроить условное оформление и т.д. (Рисунок 17.5). При этом все настройки можно выполнить как для отчета в целом («Отчет»), так и для отдельных группировок («Номенклатура», «Детальные записи»), что указано на Рисунок 17.5. Следует отметить, что группировка «Детальные записи» существует всегда, вне зависимости от того, были иные группировки или нет. Данная группировка содержит все записи, возвращаемые запросом в наборе данных.

На вкладке «Параметры» окна «Настройка» (Рисунок 17.7) необходимо указать параметры, которые необходимо отображать в интерфейсе. Здесь представлен список параметров, указанных в соответствующем разделе ранее (Рисунок 17.6) для которых не установлен флаг «Ограничение доступности». В данном случае – среди параметров присутствует только «Период», для указания даты, на которую необходимо получить остатки номенклатуры. Для отображения данных параметров в интерфейсе, указанные настройки необходимо включить в «Пользовательские настройки» (Рисунок 17.8) при помощи  (Рисунок 17.7).

«Быстрый доступ» (Рисунок 17.8) необходим для того, чтобы при запуске отчета указанные параметры отображались на главной странице отчета. Если режим редактирования изменить на

«Обычный», то для задания параметров отчета необходимо перейти в раздел «Еще - Настройка».

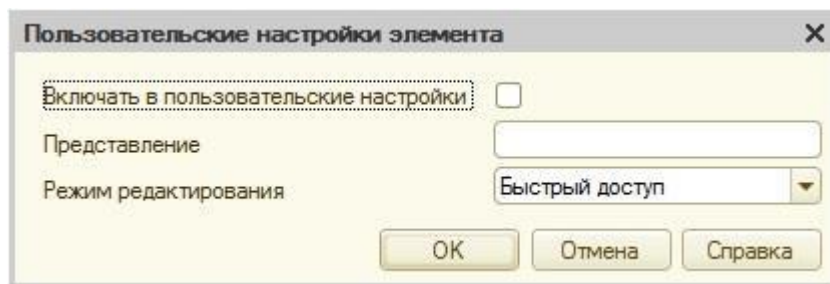


Рисунок 17.8 - Свойства элементов пользовательских настроек

После того, как все указанные действия (формирование набора данных запросом, создание вычисляемых полей, задание ресурсов, параметров отчета, настройка представления отчета) выполнены, можно запустить систему в режиме исполнения и сформировать отчет на текущую дату, а также на ряд предыдущих (чтобы посмотреть, как изменялись остатки). Обратите внимание на формирование итоговых записей (те поля, которые были добавлены в ресурсы отчета).

### Задание

Сформируйте следующие отчеты:

1. сколько товара поступило, сколько было израсходовано за определенный период, а также, сколько товара было на начало и конец выбранного периода (представить в виде таблицы);
2. какова выручка от продаж контрагентам в выбранном периоде (представить в виде диаграммы);
3. какие услуги принесли больше всего прибыли в порядке убывания (представить в виде двух вариантов: в виде списка и в виде диаграммы);
4. \*создайте отчет, в котором можно задать отбор с учетом характеристик товаров. К примеру, чтобы пользователь мог получить данные о том, сколько у него есть товара, допустим, красного цвета или сколько товаров из России и т.д.



**Контрольные вопросы**

1. Что такое отчет?
2. Как создать новый отчет?
3. Какие нужно включать параметры для создания отчета?
4. Как сформировать отчет на текущую дату?

## СОДЕРЖАНИЕ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Цель самостоятельной работы обучающихся - получить новые знания по дисциплине «Программные решения для бизнеса на платформе 1С».

Самостоятельная работа необходима для формирования у обучающихся способности самостоятельно решать задачи профессиональной деятельности, формирования умения и навыков планирования времени, формирования стремления развиваться и совершенствоваться.

Виды самостоятельной работы обучающихся указаны в таблице 1.

Таблица 1. Виды самостоятельной работы

| № п/п | Вид СРС  |
|-------|--|
| 1     | Самостоятельное изучение интернет-ресурсов и литературы на тему: «Использование списков и перечислений». |
| 2     | Самостоятельное изучение интернет-ресурсов и литературы на тему: «Журналы документов».                   |
| 3     | Самостоятельное изучение интернет-ресурсов и литературы на тему: «Автоматизированное тестирование».      |
| 4     | Самостоятельное изучение интернет-ресурсов и литературы на тему: «Конструктор печати».                   |
| 5     | Самостоятельное изучение интернет-ресурсов и литературы на тему: «Создание сложных отчётов».             |
| 6     | Подготовка к лабораторным занятиям.  |
| 7     | Оформление отчетов по практическим и лабораторным работам.   |

## УЧЕБНО-МЕТОДИЧЕСКИЕ МАТЕРИАЛЫ ПО ДИСЦИПЛИНЕ

### Основная литература

1. Шадрина, Г. В. Анализ финансово-хозяйственной деятельности: учебник и практикум для СПО / Шадрина Г. В.. – 2-е изд., пер. и доп. – Москва: Юрайт, 2020. – 431 с. – ISBN 978-5-534-04620-5. – URL: <https://urait.ru/book/analiz-finansovo-hozyaystvennoy-deyatelnosti-452784> (дата обращения: 13.02.2024). – Текст: электронный.

### Дополнительная литература

1. Голубева, О. Л. 1с: бухгалтерия.: учебник для СПО / Голубева О. Л.. – Москва: Юрайт, 2023. – 158 с. – ISBN 978-5-9916-7063-0. – URL: <https://urait.ru/book/1s-buhgalteriya-520323> (дата обращения: 13.02.2024). – Текст: электронный.

2. Гантц, И. С. 1С: Предприятие. Программирование для начинающих: Практикум: учебное пособие / И. С. Гантц. — Москва: РТУ МИРЭА, 2023. — 71 с. — ISBN 978-5-7339-1725-2. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/331547> (дата обращения: 13.02.2024). — Режим доступа: для авториз. пользователей.

3. Гамидова, Н. Г. Автоматизация бухгалтерского и налогового учета в программе «1С: Бухгалтерия 8.3»: учебное пособие / Н. Г. Гамидова. — Орел: ОрелГАУ, 2021. — 137 с. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/213656> (дата обращения: 13.02.2024). — Режим доступа: для авториз. пользователей.

4. Балданова, Т. С. Введение в 1С: Предприятие 8: учебно-методическое пособие / Т. С. Балданова, О. А. Лобсанова. — Улан-Удэ: БГУ, 2019. — 149 с. — ISBN 978-5-9793-1427-3. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/154244> (дата обращения: 13.02.2024). — Режим доступа: для авториз. пользователей.

5.Гантц, И. С. Конфигурирование в среде 1С: Предприятие: Практикум: учебное пособие / И. С. Гантц. — Москва: РТУ МИРЭА, 2021. — 66 с. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/176533> (дата обращения: 13.02.2024). — Режим доступа: для авториз. пользователей.

6.Гамидова, Н. Г. Начальная настройка и подготовка к работе программы «1С: Бухгалтерия 8.3»: учебное пособие / Н. Г. Гамидова. — Орел: ОрелГАУ, 2021. — 106 с. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/213509> (дата обращения: 13.02.2024). — Режим доступа: для авториз. пользователей.

7.Даева, С. Г. Основы разработки корпоративных информационных систем на платформе 1С: Предприятие 8.3: учебно-методическое пособие / С. Г. Даева. — Москва: РТУ МИРЭА, 2020. — 74 с. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/163859> (дата обращения: 13.02.2024). — Режим доступа: для авториз. пользователей.

8.Гладких, Т. В. Программирование на платформе 1С:Предприятие: учебное пособие / Т. В. Гладких, Л. А. Коробова, И. С. Толстова. — Воронеж: ВГУИТ, 2023. — 91 с. — ISBN 978-5-00032-634-3. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/345248> (дата обращения: 13.02.2024). — Режим доступа: для авториз. пользователей.

9.Салмина, Н. А. Управление производственными процессами в среде 1С: Бухгалтерия предприятия 3.0: практикум: учебное пособие / Н. А. Салмина, П. С. Салмин. — Нижний Новгород: ННГУ им. Н. И. Лобачевского, 2022. — 111 с. — Текст: электронный // Лань: электронно-библиотечная система. — URL: <https://e.lanbook.com/book/344534> (дата обращения: 13.02.2024). — Режим доступа: для авториз. пользователей.

### **Программное обеспечение и интернет-ресурсы**

При обучении используются следующие интернет-ресурсы:

-<https://msdn.microsoft.com/ru-ru/library> - Каталог API (Microsoft)  
и справочных материалов

-<https://habrahabr.ru/> - многофункциональный сайт, созданный  
для публикации новостей, аналитических статей, мыслей, связанных с  
информационными технологиями, бизнесом и Интернетом

-<https://kuzstu.ru/> - сайт КузГТУ