

Практические рекомендации и примеры

Лоция Софтвэа

*Опубликовано
Лоция Софтвэа
127422, г. Москва, Тимирязевская ул., д.1, стр. 2.
Телефон: (495) 748-04-74
Факс: (495) 748-03-74
E-mail: sales@lotsia.com
Интернет: <http://www.lotsia.com>*

*Авторские права
Авторскими правами обладает ООО «Лоция Софтвэа».*

Никакая часть данного документа не может быть воспроизведена или передана в любой форме и любыми способами в каких-либо целях без предварительного письменного разрешения ООО «Лоция Софтвэа».

Лицензионное соглашение, поставляемое с программным обеспечением, определяет процедуру пользования продуктом.

© 1997-2018 ООО «Лоция Софтвэа». С сохранением всех прав.

Лоция Софтвэа, Lotsia PDM, Lotsia PDM PLUS, LS Flow являются зарегистрированными торговыми марками ООО «Лоция Софтвэа».

Все остальные упомянутые в документе торговые марки являются собственностью их законных владельцев.

Информация, содержащаяся в данном документе, может быть изменена без предварительного уведомления.

*PN: 05-2500-09-RU
30.05.2018*

Содержание

<u>1</u>	<u>ВВЕДЕНИЕ В ПРАКТИЧЕСКИЕ РЕКОМЕНДАЦИИ</u>	<u>6</u>
<u>2</u>	<u>ПРИМЕРНАЯ ПОСЛЕДОВАТЕЛЬНОСТЬ СОЗДАНИЯ ПРИКЛАДНЫХ РЕШЕНИЙ...7</u>	
<u>3</u>	<u>ВВОД ДАННЫХ И ПРИМЕНЕНИЕ СПРАВОЧНИКОВ. ОБЩИЕ ЗАМЕЧАНИЯ</u>	<u>9</u>
<u>3.1</u>	<u>ОСНОВНЫЕ МЕТОДЫ РАБОТЫ С ИСПОЛЬЗОВАНИЕМ ОБЪЕКТОВ-СПРАВОЧНИКОВ.....</u>	<u>9</u>
<u>3.2</u>	<u>ИМЕЮЩИЕСЯ СПОСОБЫ ВВОДА ДАННЫХ И ПРИМЕНЕНИЯ СПРАВОЧНИКОВ</u>	<u>11</u>
<u>3.2.1</u>	<u>ИСПОЛЬЗОВАНИЕ БАЗОВЫХ («ШТАТНЫХ») СРЕДСТВ</u>	<u>11</u>
<u>3.2.2</u>	<u>ИСПОЛЬЗОВАНИЕ ДЕЙСТВИЙ НАД ОБЪЕКТАМИ</u>	<u>12</u>
<u>3.2.3</u>	<u>ИСПОЛЬЗОВАНИЕ ОТЧЕТОВ В КАЧЕСТВЕ РАБОЧЕГО (ИНТЕРФЕЙСНОГО) ОКНА КОНЕЧНОГО ПОЛЬЗОВАТЕЛЯ</u>	<u>13</u>
<u>3.2.4</u>	<u>ИСПОЛЬЗОВАНИЕ СРЕДСТВ ДОКУМЕНТООБОРОТА.....</u>	<u>14</u>
<u>4</u>	<u>НАЗНАЧЕНИЕ ИСПОЛНИТЕЛЕЙ.....</u>	<u>16</u>
<u>5</u>	<u>ПОДПИСАНИЕ ДОКУМЕНТОВ</u>	<u>18</u>
<u>6</u>	<u>СБОР И ПРОСМОТР ВИЗ И ЗАМЕЧАНИЙ ПОЛЬЗОВАТЕЛЕЙ</u>	<u>20</u>
<u>7</u>	<u>ДИНАМИЧЕСКОЕ УПРАВЛЕНИЕ ВЕТВЛЕНИЕМ ПРОЦЕССА.....</u>	<u>23</u>
<u>8</u>	<u>РЕАЛИЗАЦИЯ НАПОМИНАНИЙ</u>	<u>27</u>
<u>8.1</u>	<u>ВАРИАНТ 1. АВТОМАТИЧЕСКАЯ ГЕНЕРАЦИЯ СВОБОДНЫХ СООБЩЕНИЙ ИЗ НЕВЫПОЛНЕННОЙ ЗАДАЧИ.....</u>	<u>27</u>
<u>8.2</u>	<u>ВАРИАНТ 2. АВТОМАТИЧЕСКОЕ ПОВТОРНОЕ ПОСТУПЛЕНИЕ НЕВЫПОЛНЕННОЙ ЗАДАЧИ..</u>	<u>28</u>
<u>9</u>	<u>РАСПРЕДЕЛЕНИЕ СТРОК МАССИВОВ МЕЖДУ ИСПОЛНИТЕЛЯМИ</u>	<u>29</u>
<u>10</u>	<u>ВЫБОР ПОЛЬЗОВАТЕЛЕЙ</u>	<u>31</u>
<u>10.1</u>	<u>ОПРЕДЕЛЕНИЕ ПОЛЬЗОВАТЕЛЯ-НАЧАЛЬНИКА ОТДЕЛА (ГРУППЫ, СЕКТОРА И Т.П.)</u>	<u>31</u>
<u>10.2</u>	<u>ИСПОЛЬЗОВАНИЕ ПРОЕКТА «СТРУКТУРА ПРЕДПРИЯТИЯ»</u>	<u>31</u>
<u>10.3</u>	<u>МНОЖЕСТВЕННЫЙ ВЫБОР ИСПОЛНИТЕЛЕЙ (АДРЕСАТОВ)</u>	<u>32</u>
<u>11</u>	<u>ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ ВНЕШНИХ РАБОТ И ИНФОРМАЦИОННЫХ РАССЫЛОК.....</u>	<u>39</u>
<u>12</u>	<u>ЦИКЛИЧЕСКАЯ ОДНОВРЕМЕННАЯ ОБРАБОТКА ДВУХ МАССИВОВ</u>	<u>40</u>
<u>13</u>	<u>ОСОБЕННОСТИ ИСПОЛЬЗОВАНИЯ АГРЕГАТНЫХ ФУНКЦИЙ ДЛЯ МАССИВОВ ЗНАЧЕНИЙ.....</u>	<u>41</u>
<u>14</u>	<u>ОПЕРАЦИИ НАД ДОКУМЕНТАМИ РАБОТЫ</u>	<u>42</u>

<u>15</u>	<u>ИМПОРТ ДОКУМЕНТОВ-ФАЙЛОВ ПРЕДОПРЕДЕЛЕННОГО ПРОЦЕССА В АРХИВ</u>	<u>43</u>
<u>16</u>	<u>ЗАЩИТА ОТ ИМПОРТА ДОКУМЕНТОВ В ОПРЕДЕЛЕННЫЕ ОБЪЕКТЫ</u>	<u>45</u>
<u>17</u>	<u>ПОСЛЕДОВАТЕЛЬНОЕ ДОБАВЛЕНИЕ ДАННЫХ В ЗНАЧЕНИЕ АТТРИБУТА С ПРОВЕРКОЙ УНИКАЛЬНОСТИ И СОРТИРОВКОЙ.....</u>	<u>47</u>
<u>18</u>	<u>УПРАВЛЕНИЕ ФОРМАМИ ЭТАПОВ РАБОТ С ПОМОЩЬЮ ДЕЙСТВИЙ НАД ПЕРЕМЕННЫМИ.....</u>	<u>49</u>
<u>18.1</u>	<u>ФОРМИРОВАНИЕ ВЫПАДАЮЩЕГО СПИСКА ЗНАЧЕНИЙ В ФОРМЕ ЗАДАЧИ.....</u>	<u>49</u>
<u>19</u>	<u>РАСЧЕТ КОЛИЧЕСТВА ОПРЕДЕЛЕННЫХ СИМВОЛОВ (ГРУПП СИМВОЛОВ) В СТРОКЕ</u>	<u>50</u>
<u>20</u>	<u>АВТОМАТИЧЕСКИЙ ПЕРЕБОР И ОБРАБОТКА ОБЪЕКТОВ.....</u>	<u>51</u>
<u>21</u>	<u>РЕКОМЕНДАЦИИ ПО SQL-ОТБОРУ ОБЪЕКТОВ ПО АТТРИБУТАМ И АТТРИБУТОВ ОБЪЕКТА</u>	<u>53</u>
<u>22</u>	<u>УДАЛЕНИЕ ГРУППЫ АТТРИБУТОВ.....</u>	<u>54</u>
<u>23</u>	<u>НУМЕРАЦИЯ ГРУПП В ФОРМАХ.....</u>	<u>55</u>
<u>24</u>	<u>ВСТАВКА ИЗОБРАЖЕНИЙ В ФОРМЫ.....</u>	<u>56</u>
<u>25</u>	<u>НЕКОТОРЫЕ ПРАКТИЧЕСКИЕ ПРИЕМЫ НАСТРОЙКИ ОТЧЕТОВ.....</u>	<u>58</u>
<u>25.1</u>	<u>ПРИМЕР СОЗДАНИЯ ОТЧЕТА</u>	<u>58</u>
<u>25.2</u>	<u>СОРТИРОВКА СТРОК ОТЧЕТА.....</u>	<u>61</u>
<u>25.3</u>	<u>НУМЕРАЦИЯ СТРОК ОТЧЕТА.....</u>	<u>62</u>
<u>25.4</u>	<u>ГРУППИРОВКА СТРОК ОТЧЕТА</u>	<u>62</u>
<u>25.5</u>	<u>ОБРАБОТКА ПОВТОРЯЮЩИХСЯ СТРОК ОТЧЕТОВ.....</u>	<u>64</u>
<u>25.6</u>	<u>БЫСТРОЕ УДАЛЕНИЕ ВСЕХ ТИПОВ ОБЪЕКТОВ ИЗ ВКЛАДКИ «РАЗДЕЛЫ» ОТЧЕТА</u>	<u>67</u>
<u>25.7</u>	<u>ИСПОЛЬЗОВАНИЕ АРГУМЕНТОВ ОТЧЕТА</u>	<u>67</u>
<u>25.8</u>	<u>ВВОД АРГУМЕНТОВ НЕПОСРЕДСТВЕННО В ОКНЕ ОТЧЕТА</u>	<u>68</u>
<u>25.9</u>	<u>ОСОБЕННОСТИ ЗАПУСКА И ВЫПОЛНЕНИЯ ОТЧЕТОВ С АРГУМЕНТАМИ</u>	<u>69</u>
<u>25.10</u>	<u>ИСПОЛЬЗОВАНИЕ ИНФОРМАЦИИ ИЗ РАЗЛИЧНЫХ ОБЪЕКТОВ ПРИ ФОРМИРОВАНИИ СТРОКИ ОТЧЕТА. ПРИМЕНЕНИЕ ИСТОЧНИКОВ ДАННЫХ.....</u>	<u>71</u>
<u>25.10.1</u>	<u>НЕСКОЛЬКО ЗАМЕЧАНИЙ ПО ИСПОЛЬЗОВАНИЮ ОТЧЕТОВ С ИСТОЧНИКАМИ ДАННЫХ</u>	<u>75</u>
<u>26</u>	<u>ОТОБРАЖЕНИЕ ИМЕНИ ПОЛЬЗОВАТЕЛЯ/ГРУППЫ ДЛЯ ЗНАЧЕНИЯ АТТРИБУТА С СООТВЕТСТВУЮЩИМ ИДЕНТИФИКАТОРОМ.....</u>	<u>77</u>
<u>27</u>	<u>ОРГАНИЗАЦИЯ ВСТРЕЧ С ИСПОЛЬЗОВАНИЕМ КАЛЕНДАРЯ</u>	<u>78</u>
<u>28</u>	<u>РЕКОМЕНДАЦИИ ПО ОПТИМИЗАЦИИ РАБОТЫ С ИНФОРМАЦИОННЫМИ ОБЪЕКТАМИ</u>	<u>80</u>

28.1	БЫСТРЫЙ ВЫБОР КОРНЕВЫХ ОБЪЕКТОВ	80
28.2	ДОБАВЛЕНИЕ ОБЪЕКТОВ В ПРОЕКТ И ПЕРЕМЕЩЕНИЕ ОБЪЕКТОВ МЕЖДУ ПРОЕКТАМИ....	80
28.3	УДАЛЕНИЕ ОБЪЕКТОВ ИЗ ПРОЕКТА	81
28.4	УДАЛЕНИЕ ОБЪЕКТА ИЗ ПРОЕКТОВ	81
28.5	ЗАДАНИЕ ЗНАЧЕНИЙ АТТРИБУТОВ ДЛЯ БОЛЬШОГО ЧИСЛА СУЩЕСТВУЮЩИХ ОБЪЕКТОВ.	81

1 Введение в практические рекомендации

Настоящее Руководство преследует цель обобщить возможные решения по реализации в системе Lotsia PDM PLUS наиболее часто встречающихся практических задач. Не все пользователи системы могут столкнуться с подобными задачами. Однако, если подобные задачи, возникнут, то настоящее руководство, возможно, поможет вас их решить.

Документ предназначен для администраторов и подготовленных пользователей системы Lotsia PDM PLUS, имеющих навыки настройки шаблонов работ, действий над объектами и переменными, составления выражений с помощью Редактора выражений.

2 Примерная последовательность создания прикладных решений

Прикладные решения на основе системы Lotsia PDM PLUS могут создаваться в следующей примерной последовательности:

1. Разработайте модель данных.
2. Выделите основные потоки документооборота, подлежащие автоматизации.
3. Определите, какую СУБД вы будете использовать.
4. Инсталлируйте Lotsia PDM PLUS (см. [«Порядок установки программы»](#)), [«Установка программы»](#), [«Установка аппаратной защиты»](#)), при наличии обновлений – произведите обновление установленных компонентов программы.
5. Создайте пустую БД (см. [«Создание пустой базы данных»](#), [«Инициализация базы данных»](#)).

Важно! Если планируется использование нескольких баз данных и модуля репликации для обмена данными между ними, обязательно, в каждой базе данных установите уникальный номер филиала. В противном случае, данные одного филиала могут быть перезаписаны данными другого филиала.

6. Зарегистрируйте в программе основные элементы структуры БД:
 - 6.1. [Типы объектов](#).
 - 6.2. [Атрибуты](#).
 - 6.3. [Типы связей](#).
 - 6.4. Задайте [допустимые атрибуты для каждого типа объекта](#).
 - 6.5. Задайте [входимость объектов](#).
7. Создайте необходимый набор средств, обеспечивающих работу пользователей в системе.
 - 7.1. [Пользовательские формы](#).
 - 7.2. [Сохраненные запросы](#).
 - 7.3. [Действия](#).
 - 7.4. [Отчеты](#).
 - 7.5. [Шаблоны для экспорта отчетов](#).
8. Создайте необходимые профили пользователей (см. [«Регистрация пользователей»](#), [«Создание профиля»](#)).
 - 8.1. Настройте [Главное меню](#), [панель инструментов](#) и [меню объектов](#) для каждого [профиля](#).
 - 8.2. Включите в профили соответствующие [пользовательские формы](#) и [действия](#).
9. Создайте [пользователей](#) и [группы пользователей](#), задайте [приоритеты](#), [привилегии](#).
10. Создайте [Глобальные адресные книги](#), сопоставьте их группам пользователей, создайте (сгенерируйте автоматически) контакты пользователям, определите права групп пользователей на адресные книги.
11. Создайте [защищенные хранилища](#).
12. Создайте [необходимые библиотеки](#), задайте для них группы пользователей.
13. Зарегистрируйте [типы документов](#).
14. Зарегистрируйте [приложения](#) для обработки документов и методы их запуска.
15. Проведите обучение пользователей.
16. Организуйте первичное наполнение базы данных (см. [«Импорт объектов»](#), [«Расширенный импорт объектов»](#), [«Импорт документов»](#)).
17. Если требуется, создайте типовые документы и настройте [синхронизацию атрибутов](#) с полями этих документов.
18. Если это необходимо, создайте [бизнес-правила](#), [представления проектов](#).
19. Создайте [шаблоны](#) типовых процессов.
 - 19.1. Создайте маршрут прохождения (блок-схему) процесса
 - 19.2. Создайте формы по умолчанию для элементов шаблона.

- 19.3. Задайте документы шаблона, необходимые для выполнения этапов работы.
- 19.4. Задайте свойства этапов и переходов, условия выполнения переходов.
- 19.5. Определите исполнителей этапов, выполните проверку карты.
- 19.6. Задайте переменные шаблона.
- 19.7. Создайте действия над объектами и переменными и добавьте их к задачам и переходам, задайте условия выполнения действий.
- 19.8. Настройте формы задач (и, при необходимости, переходов);
- 19.9. Запустите работу по шаблону, измените шаблон по результатам проведенного тестирования.

3 Ввод данных и применение справочников. Общие замечания

Ручной ввод пользователем информации в систему следует максимально ограничивать, причины этого очевидны: помимо облегчения работы, минимизация свободного ввода позволяет сократить количество ошибок и в итоге получать гарантированно достоверный результат поиска данных или выполнения отчетов.

В большинстве случаев, при вводе данных в систему тем или иным образом используется уже имеющаяся в БД информация. Простейший случай – использование вместо ручного ввода списков значений для присвоения значений атрибутам.

Для практической работы, как правило, используются более сложные и гибкие механизмы автоматизации ввода значений и построения проектов с использованием имеющихся в БД данных. Основной способ хранения данных Lotsia PDM PLUS заключается в применении связанных между собой объектов, содержащих набор атрибутов с присвоенными значениями. В большинстве случаев при создании (или информационном наполнении) проекта происходит работа минимум с двумя проектами – условно назовем их «Основной проект» и «Справочник».

3.1 Основные методы работы с использованием объектов-справочников

Работа со справочниками ведется в основном следующим методами и/или их комбинациями.

1. *Суть метода:* **копирование значений атрибутов** объекта справочника, использование атрибутов объекта или проекта.

Возможное использование: копирование реквизитов контрагентов при создании различных документов.

Преимущества, значимые при выборе данного метода: простота настройки, фиксация в основном проекте данных, актуальных на данный момент.

Некоторые существенные ограничения: отсутствие однозначной связи объекта основного проекта с объектом справочника, дублирование информации.

2. *Суть метода:* **заимствование объектов справочника.**

Возможное использование: использование единственного объекта, применяемого в различных проектах.

Преимущества, значимые при выборе данного метода: динамическое обновление информации о связанном объекте справочника. Удобство поиска связанных объектов. Возможность независимого ввода данных применительно к конкретному проекту (использование атрибутов проекта). Отображение информации о родителях объекта в окне данного объекта. Возможность установки защиты на каждое вхождение объекта.

Некоторые существенные ограничения: необходимость согласования изменений заимствованных компонентов для всех его применений. Обязательность создания и учета правил входимости. Следует учитывать возможность распространение наследуемых прав с родительских объектов.

3. *Суть метода:* **включение объектов справочника в основной проект по отдельному (другому) типу подчиненной связи.**

Возможное использование: указание абонентов документов, построение технологии производства изделий на основе его структуры.

Преимущества, значимые при выборе данного метода: динамическое обновление информации о связанном объекте справочника. Удобство поиска связанных объектов. Возможность независимого ввода данных применительно к конкретному проекту (использование атрибутов проекта). Разделение информации о связанных по различным типам связи объектах. Возможность установки защиты на каждое вхождение объекта.

Некоторые существенные ограничения: необходимость согласования изменений заимствованных компонентов для всех его применений. Обязательность создания и учета правил входимости. Следует учитывать возможность распространение наследуемых прав с родительских объектов.

4. *Суть метода:* **связывание объектов справочника и объектов основного проекта горизонтальными связями.**

Возможное использование: указание ссылочных документов, похожих изделий, замен.

Преимущества, значимые при выборе данного метода: динамическое обновление информации о связанном объекте справочника. Удобство поиска связанных объектов. Возможность независимого ввода данных применительно к конкретному проекту (использование атрибутов проекта). Объединение информации об объектах, независимо от их расположения в различных проектах, отсутствие ограничений по входимости.

Некоторые существенные ограничения: отсутствие отдельного механизма защиты данных связей. Невозможность использования атрибутов проекта.

5. *Суть метода:* **копирование объекта справочника** и использование в рабочем проекте полученной копии.

Возможное использование: копирование типовых технологических операций, типовых проектных решений и т. п. с последующей привязкой к основному проекту.

Преимущества, значимые при выборе данного метода: фиксация в основном проекте данных, актуальных на данный момент применительно к данному проекту.

Некоторые существенные ограничения: отсутствие автоматического обновления данных при изменении копируемого объекта справочника. Появление дополнительных входимостей у потомков копируемых из справочника объектов.

6. *Суть метода:* **копирование объекта справочника** с его дочерними объектами и документами архива (или без таковых) **по шаблону** копирования, включение полученной копии в основной проект.

Возможное использование: создание конфигураций изделий, создание экземпляров изделий, добавление типовых комплектов документов, использование существующих проектов в качестве прототипов при разработке новых.

Преимущества, значимые при выборе данного метода: возможность автоматического копирования структуры проекта, отсутствие обязательности доработки механизма копирования при изменении справочника. Гибкая настройка шаблонов копирования. Возможность копирования документов архива. Возможность изменения защиты объектов и типов связи при копировании.

Некоторые существенные ограничения: отсутствие автоматического обновления копии при изменении справочника.

7. *Суть метода:* **получение идентификатора (ID объекта справочника, ID связи или ID пользователя) и запись его в атрибут** в основном проекте.

Возможное использование: хранение защищенной ссылки на объект справочника, на вхождение объекта справочника (для получения из справочника атрибутов проекта), хранение ID сопоставленного данному объекту пользователя БД.

Преимущества, значимые при выборе данного метода: отсутствие двойного ввода информации, возможность автоматического получения актуальной информации из справочника. Простота и однозначность нахождения объекта справочника. Возможность защиты связи с объектом справочника, независимость установки и наследования прав доступа на основной проект и справочник.

Некоторые существенные ограничения: необходимость использования действий над объектами, либо полей с SQL-запросами для получения (отображения в форме) информации из справочника. В некоторых случаях это может привести к заметному замедлению работы программы.

Частным случаем большинства указанных методов работы можно считать также разрыв связей с найденными объектами и удаление значений атрибутов. В этом случае понятия «Основной проект» и «Справочник» фактически идентичны.

В зависимости от предметной области, используемой модели данных, конкретной задачи и требований, предъявляемых к настройке, применяется тот или иной из указанных методов использования справочников.

Независимо от метода использования справочника, до использования объекта справочника его следует найти в БД и выбрать.

Реализовать для конечного пользователя удобный выбор объектов из справочников – отдельная задача, решать которую также можно различными способами (см. раздел «Имеющиеся способы ввода данных и применения справочников»).

3.2 Имеющиеся способы ввода данных и применения справочников

Выбор используемого способа ввода данных зависит от конкретной задачи и требуемой степени автоматизации работы пользователя, а также от подготовленности конечного пользователя. Как правило, для построения максимально удобного интерфейса конечного пользователя требуется также выполнение максимального объема работ при настройке системы. Далее приводится часть возможных способов ввода данных, некоторые из которых не всегда сразу очевидны.

3.2.1 Использование базовых («штатных») средств

Это простейший вариант, не требующий дополнительной настройки интерфейса пользователя. Может использоваться при настройке системы, а также для выполнения слабо формализованных либо стандартных процедур пользователями, обладающими достаточными навыками работы в Lotsia PDM PLUS.

Данный вариант предусматривает открытие окна основного проекта и окна справочника, использование перетаскивания для заимствования (или связывания) объектов из справочника, копирование значений атрибутов через буфер обмена.

Вариант использования данного метода – **добавление дочерних (родительских, связанных) объектов на соответствующей вкладке объекта.**

К штатным средствам можно отнести использование **классификаторов** для присвоения значений атрибутам. При наличии настроенных форм для типов объектов данный метод используется в практической работе для решения соответствующих задач. Классификатор также применим и в формах действий для присвоения значений переменным.

Ограничения применения классификатора, в основном, следующие:

- окно для выбора значений из классификатора имеет минимум настроек, и для конечного пользователя, в ряде случаев может показаться не совсем удобным.
- значения атрибутов с выбранного объекта классификатора автоматически сцепляются со значениями атрибутов родительских объектов классификатора. Это свойство классификаторов в ряде случаев полезно, так как сокращает ввод данных при настройке. Но, если требуется получать значения атрибутов именно с выбранных объектов классификатора различного произвольного уровня вложенности в дереве, без учета значений атрибутов объектов верхних уровней – использовать классификатор затруднительно (требуется написание действий и использование дополнительных «служебных» атрибутов).

Если указанные ограничения для Вашей настройки существенны, в большинстве случаев вместо использования классификаторов можно рекомендовать написание действий над объектами и применение в них окон выбора объектов и окон выбора.

Для решения некоторых, сходных с рассматриваемыми, задач, возможно также использование таких инструментов, как **исполнения** и **варианты** (в данном контексте – списки исполнений или старых вариантов можно рассматривать как справочники). Данные инструменты могут использоваться напрямую или с написанием соответствующих действий над объектами. В частности, использование вариантов, помимо вариантного проектирования изделий, возможно в совершенно различных областях, например, подготовка нескольких вариантов ответа на входящее письмо, выбор варианта технологической операции с соответствующим набором оборудования, оснастки т.д.

Не следует забывать о возможности редактирования атрибутов из **списка значений**. Для некоторых случаев использование таких списков оказывается наилучшим решением (в данном контексте – список значений атрибута является простейшим справочником).

3.2.2 Использование действий над объектами

Для реализации более сложных алгоритмов выбора объектов справочника следует использовать действия над объектами. В большинстве случаев рекомендуется запуск действий осуществлять по нажатию кнопок в формах (атрибутивных, действий, отчетных, документооборота), а также с использованием настроенных пунктов меню – Главного меню (и соответствующих кнопок инструментальной панели) или контекстных меню.

В действиях над объектами имеются различные возможности выбора объектов справочника, в результате присваивается значение переменной с типом значения «Объект», либо переменным с типом значения число (или строка) присваиваются значения, соответствующие ID объекта или другой сущности – например – ID пользователя. Отметим, что в строковую переменную можно записать значения ID нескольких объектов, через разделитель (в качестве разделителя обычно удобно использовать запятую, такие значения можно, например, вставлять в SOL – запрос).

В действиях над объектами, для реализации выбора объектов рекомендуется использование следующих инструментов:

- **списки и выпадающие списки** для строковых и числовых переменных, с использованием выражений для построения списка. Выражение должно формировать список значений, соответствующих выбираемым объектам справочника. В списке для выбора значений визуализируется информация, позволяющая пользователю осуществить выбор, а присваивается ID выбранной в списке сущности (объекта). Это наиболее простой и очевидный для конечного пользователя способ редактирования данных. Ограничений использования два – можно выбрать только одно значение и количество строк в линейном (неструктурированном) списке не должно быть слишком большим, иначе работать с таким списком неудобно.
- **выбор объектов** – использование для переменных типа «Объект» настраиваемых окон выбора объектов. Позволяет выбирать один объект из требуемого справочника (проекта) или списка объектов, сформированного по условиям отбора. При выборе объекта учитываются заданные условия отбора – объект, не удовлетворяющий условиям, выбрать будет невозможно.
- **окна выбора**. Данный инструмент обеспечивает максимальную гибкость и функциональность. Как правило, используется отдельное (дополнительное)

действие с окном выбора, запускаемое по кнопке в форме (например – в форме шага действия над объектом). Кнопка располагается рядом с редактируемым полем формы. Данный способ рекомендуется, если нужно выбирать более одного значения одновременно, а также если для выбора требуется использовать структурированный список (дерево проекта), в котором требуется ограничить показ объектов (то есть строить дерево для выбора с использованием условий отбора). Для поиска выбираемого объекта в дереве окна выбора имеется специальная строка для ввода начала названия объекта. Выбранное в окне выбора значение может присваиваться переменной типа «Объект» (только для единичного выбора), либо строковой переменной (присваивается строка с ID выбранных объектов, разделитель – запятая). При использовании окна выбора в дополнительном действии, там же можно осуществить считывание данных с выбранного объекта (объектов) и их обработку, с тем, чтобы вернуть в переменные формы, из которой запускалось действие, требуемые значения в нужном формате. Например – выбрать из справочника объект сотрудника, далее считать требуемые атрибуты (например – ID сопоставленного пользователя). Если требуется – сформировать строку (строки) требуемого формата (например, вместо фамилии имени и отчества с помощью строковых функций сформировать фамилию с инициалами). Далее – считанные (сформированные) значения вернуть в переменные формы, из которой вызывалось данное действие.

В некоторых случаях объект справочника может быть **выбран автоматически**, то есть удастся определить условия отбора таким образом, что объект справочника идентифицируется однозначно. Для того чтобы автоматически присвоить переменной типа «Объект» значение согласно заданным условиям отбора, достаточно поместить такую переменную форму действия. В таком случае рекомендуется использование форм с автоматическим завершением. Аналогичный результат (то есть присвоения переменной значения на основании условий отбора) можно получить с помощью функции VarCheckValues.

Альтернативой применения условий отбора (для выбора объектов справочника) может быть использование функции редактора выражений f_ExecSQLSelect_3, внутри функций действия Set или SetByID. С помощью данных функций значения из справочника можно получить «напрямую», без дополнительных шагов с формами, а также без использования условий отбора в свойствах переменной. В ряде случаев это может быть удобнее.

3.2.3 Использование отчетов в качестве рабочего (интерфейсного) окна конечного пользователя

Применение отчетов в качестве рабочего (интерфейсного) окна, в ряде случаев позволяет, используя минимум настроек (в сравнении, например, с использованием средств документооборота), создать для конечного пользователя удобный рабочий инструмент. В первую очередь применение отчета рекомендуется рассмотреть при необходимости создания табличного интерфейса для пользователя. При настройке отчета может быть определена реакция системы на одинарный и/или двойной щелчок мыши на строке или отдельном поле отчета, возможно также добавление в отчет кнопок (на практике кнопки используются в большинстве случаев). По нажатию кнопок в отчете (а также по щелчкам мыши по строкам отчета и различным полям отчета), в зависимости от настроек, возможно выполнение действий, а также открытие окон объектов, выполнение отчетов, скриптов.

После выполнения действия, вызванного указанными событиями формы отчета, отчет может быть автоматически обновлен. В обновленном отчете будут отражены

изменения данных, произошедшие при выполнении указанных действий. Отметим, что для ускорения работы, в некоторых случаях вместо обновления отчета после, например, каждого нажатия кнопки, целесообразно использовать возврат данных из действия в поле отчета.

Некоторым ограничением применения отчета в качестве интерфейсного окна является невозможность ввода данных пользователем непосредственно в форму отчета (за исключением ввода аргументов отчета – см. раздел «Ввод аргументов непосредственно в окне отчета»). Для редактирования данных необходимо открытие дополнительного окна (например – формы действия) где и осуществляется ввод данных. Если данное ограничение для Вашей настройки существенно – следует использовать не отчет, а средства документооборота (см. раздел «Использование средств документооборота»).

Отметим также, что сам отчет может использоваться в качестве атрибутивной формы объекта, как отдельная вкладка или внедренная форма.

3.2.4 Использование средств документооборота

В некоторых случаях для повышения удобства работы пользователя требуется обеспечить возможность автоматического получения и обновления данных в форме при изменении пользователем аргументов, а также возможность работы с данными, представленными в виде таблицы, в том числе – ввод данных непосредственно в поля таблицы. Такая возможность в системе реализуется с использованием средств документооборота (рекомендуется также рассмотреть возможность использования отчета, в том числе с расположением формы аргументов в окне отчета (см. раздел «Ввод аргументов непосредственно в окне отчета») что может оказаться предпочтительнее с точки зрения трудоемкости выполнения настройки, см. раздел «Использование отчетов в качестве рабочего (интерфейсного) окна конечного пользователя»).

Если применение других инструментов не целесообразно, может применяться шаблон, состоящий только из начального и конечного этапов. Практически же используется только форма первого этапа работы.

Стандартные кнопки формы обычно [отключаются](#).

Для представления данных в виде таблицы используются переменные с типом значения «Массив».

Условно работу пользователя с подобным инструментом можно разделить на следующие основные операции:

1. Открытие формы начала работы. Данную операцию рекомендуется выполнять путем запуска работы из действия над объектом, выполняемым по нажатию кнопки в атрибутивной форме объекта.
2. Заполнение формы данными из справочников. Данные из справочников могут извлекаться как действиями над объектами, так и действиями над переменными. Действия над переменными использовать предпочтительно, так как скорость их выполнения, как правило, выше. Кроме того, при выполнении действий над переменными можно получать данные в колонки формы непосредственно, путем выполнения SQL-запроса, что дает широкие возможности отбора данных. Действия над объектами рекомендуется использовать в обоснованных случаях, например, при необходимости выполнения одновременно и изменений атрибутивной информации объекта. Заполнение формы данными может происходить действиями по различным событиям (например, «После открытия окна задачи», «После нажатия кнопки формы», «После нажатия кнопки-переменной»).

Примечание: по возможности рекомендуется использовать динамическое формирование и обновление [выпадающих списков](#) значений для переменных-аргументов.

3. Ввод/редактирование данных пользователем. В общем случае возможно использование следующих способов работы:

- редактирование данных в полях формы, затем, например, по нажатию кнопки в форме, выполнение циклического действия над объектами, которое выполняет последовательное считывание введенных данных (аргументами действия являются значения из строк массивов) и устанавливает атрибуты и/или устанавливает/удаляет связи для соответствующих объектов. По завершении – повторное выполнение считывания данных для обновления информации в форме;
- после каждого ввода информации в поле формы запуск действий, обрабатывающих соответствующие объекты, и обновление формы.

Выбор способа работы или некоторой промежуточной комбинации из приведенных способов, в каждом случае осуществляется индивидуально, в зависимости от конкретной задачи и предъявляемых пользователями требований.

4. Закрытие формы начала работы. Обычно выполняется по стандартной кнопке «Отмена», без сохранения процесса. В некоторых случаях возможен и запуск работы, с целью выполнения обработки введенных пользователем данных сервером автоматических этапов (это актуально при отсутствии у пользователя необходимых прав доступа для изменения данных).

4 Назначение исполнителей

В большинстве случаев назначение исполнителей производится через установку значений атрибутам объектов. В атрибут объекта можно заносить как имя пользователя, так и его идентификатор.

Если используется идентификатор, отображать его в форме для объекта в виде числа не следует. Для отображения имени соответствующего пользователя можно использовать вычисляемое поле, выражение которого содержит функцию `f_UserName (<атрибут_с_исполнителем>)`, либо настроить отображение поля с идентификатором пользователя («ID сопоставленного пользователя») следующим образом (см. Рисунок 1).

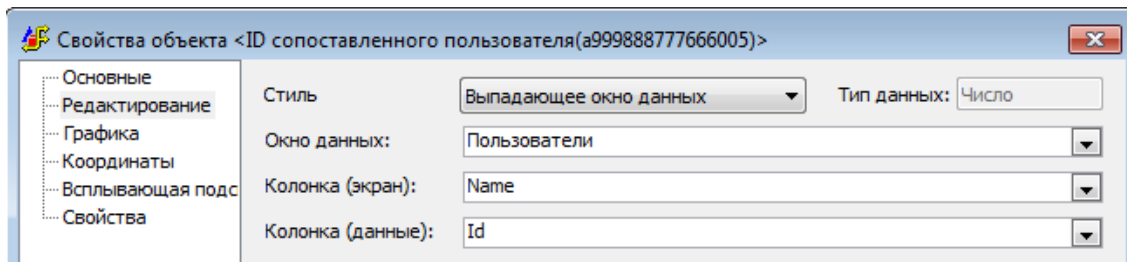


Рисунок 1 Настройка отображения поля с идентификатором пользователя

Использование идентификатора дает возможность автоматически однозначно определить пользователя БД, например, в ходе процесса значение атрибута с исполнителем можно получить (можно использовать как действия над объектами, так и действия над переменными), при необходимости обработать, и использовать для назначения исполнителей этапов работ.

Если назначение исполнителей производится в форме predetermined сообщения, рекомендуется включить в форму задачи или перехода переменные типа «Пользователь» и выбирать исполнителей, присваивая значения указанным переменным. Значение таких переменных можно передать в действие над объектами, устанавливающее объекту значение атрибута с исполнителем. Таким образом, в атрибут объекта попадет описание или идентификатор выбранного исполнителя. Сами же значения переменных типа «Пользователь» используются для установки исполнителей конкретных задач процесса.

Если при назначении исполнителей недостаточно отображения только имени пользователя–исполнителя, а требуется указание дополнительной информации, например, занимаемой должности, номера отдела и т.п., рекомендуется поступать следующим образом:

- создать в БД объекты пользователей. Удобно данные объекты структурировать по подразделениям предприятия, создав отдельный проект «[Структура предприятия](#)»;
- каждому объекту сотрудника присвоить необходимую атрибутивную информацию;
- каждому объекту сотрудника в атрибут записать ID соответствующего пользователя БД;
- при назначении исполнителя, действием над объектом выбрать соответствующий объект сотрудника (использовать условие отбора по ID пользователя и типу объекта) получить у объекта сотрудника необходимую атрибутивную информацию;

- установить объекту один или несколько атрибутов, занеся в них идентификатор пользователя и другую требуемую информацию.

Ведение структуры предприятия в виде отдельного проекта-справочника может использоваться для различных целей, в том числе выбора исполнителей этапов бизнес-процессов, создания списков рассылки, получение актуальных справочников предприятия и т.д.

Один из возможных вариантов назначения исполнителя – установка связи между объектом, на который назначается исполнитель, и соответствующим объектом сотрудника, с установкой необходимых атрибутов проекта (например, даты назначения и реквизитов приказа о назначении). Данный подход облегчает назначение произвольного количества исполнителей и позволяет использовать форму типа «Дочерние объекты» для просмотра списка исполнителей. Однако передать данные об исполнителях в документ путем синхронизации атрибутов при данном подходе невозможно (в случае документов MS Word и MS Excel можно использовать экспорт отчетов по шаблону).

Для практических целей, при назначении исполнителей обычно достаточно указания ID пользователя (или ID нескольких пользователей через разделитель) в атрибутах объекта.

5 Подписание документов

Использование внешних систем электронной подписи не всегда юридически и экономически оправдано. При отсутствии специальных требований, оптимальным решением для подписания документов может оказаться использование штатных средств Lotsia PDM PLUS – присвоение значений атрибутам и защита их от изменений. Подписываются информационные объекты, при необходимости выполняется синхронизация атрибутов с документами архива.

Возможны различные варианты подписания, но общий подход к решению данной задачи рекомендуется следующий:

- подписание производить в рамках предопределенного процесса;
- в начале процесса серверным действием устанавливается запрет редактирования подписываемого объекта;
- в ходе процесса, после фиксации исполнителем этапа в форме предопределенного сообщения того факта, что документ им подписывается, (соответствующий пункт выбора может называться «Согласовано», «Нет замечаний», «Подписать» и т.п.) серверным действием осуществляется подписание.

Могут использоваться различные варианты подписания, например:

- присвоение атрибуту значения, содержащего имя текущего пользователя;
- присвоение атрибуту значения, содержащего текущую дату;
- присвоение атрибутам значений, содержащих имя текущего пользователя и текущую дату;
- присвоение атрибутам значений, содержащих текущую дату, должность подписывающего, фамилию имя и отчество подписывающего;
- создание объекта «Подпись», содержащего необходимую атрибутивную информацию и связывание данного объекта с подписываемым объектом (рекомендуется использовать подчиненную связь);
- связывание соответствующего объекта сотрудника с подписываемым объектом, с установкой даты подписания в виде атрибута проекта (используется подчиненная связь)

При принятии решения об использовании того или иного варианта подписания, важно учесть требования, предъявляемые к оформлению каждой подписи. При определении наиболее правильного для конкретной задачи варианта подписания рекомендуется принимать во внимание следующее.

Можно условно разделить подписи на именные и групповые.

Под именными подписями будем понимать подписи, имеющие собственное название, обычно заранее наносимые на бланк бумажного документа, зачастую вместе с именем и должностью подписывающего, такая подпись в общем случае может иметь несколько отдельных полей:

- название (смысл) подписи – «Разработал», «Проверил», «Выполнил нормоконтроль», «Утвердил» и т.д.;
- должность подписывающего – «Конструктор 1 категории», «Руководитель группы», «Инженер бюро нормоконтроля», «Директор»;
- фамилия;
- дата подписания.

Для каждой из подобных подписей могут быть использованы, например, следующие атрибуты: «Должность_разработал», «ФИО_разработал», «Дата_разработал».

Под групповой подписью будем понимать различные согласования, список которых формируется индивидуально для конкретного документа. Например, список сотрудников, которых необходимо ознакомить с приказом и т.п. В общем же случае, при реализации в Lotsia PDM таких подписей может быть использован один строковый атрибут, куда информация о подписывающих и датах подписания заносится через

разделитель, в виде списка. Чтобы в форме объекта подписи отображались с новой строки, в качестве разделителя используется стандартный набор символов: ~г~п .

При реализации подписей в виде отдельных объектов нет необходимости создания для каждой именной подписи собственного набора атрибутов, однако, перенос значений атрибутов в документ архива путем синхронизации атрибутов выполнить не удастся.

Для обеспечения возможности автоматического указания должности подписывающего и другой информации, рекомендуется использовать проект – справочник «[Структура предприятия](#)».

Возможны и другие подходы к подписанию документов, в том числе и не связанные с использованием документооборота, однако, в этом случае пользователь редактирует атрибуты согласуемого объекта, вопросы защиты данных решаются индивидуальным образом для различных случаев.

6 Сбор и просмотр виз и замечаний пользователей

Возможны следующие варианты:

1. Для сбора виз используются переменные с типом значения «Одиночное».
2. Для сбора виз используются переменные с типом значения «Массив».

Пример шаблона процесса согласования см. Рисунок 2.

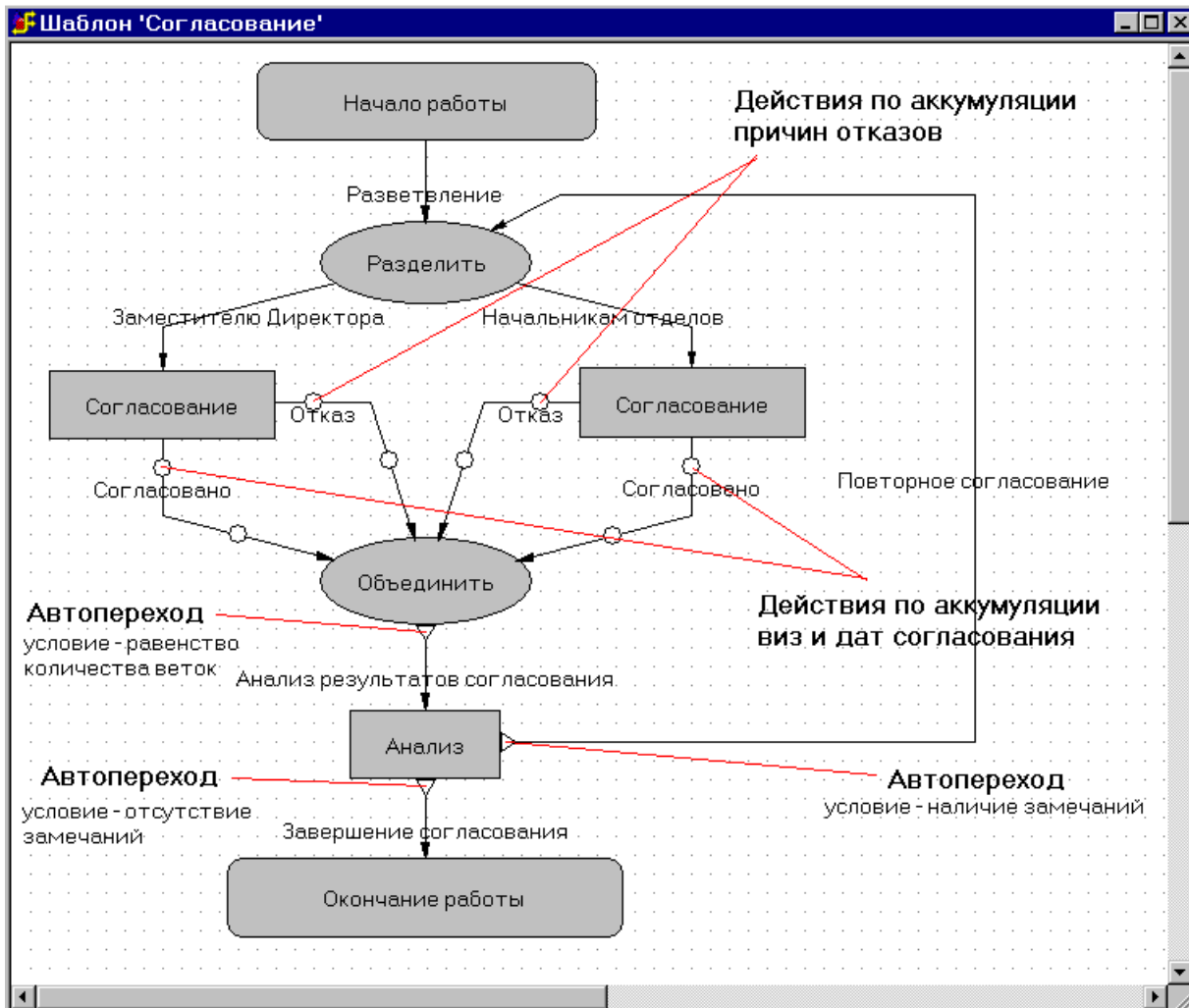


Рисунок 2 Пример шаблона процесса согласования

Рассмотрим первый вариант. Для организации процесса используется разветвитель. Переход на разветвитель может происходить от начала работы. Переход ручной. Чтобы избежать отправки сообщения самому себе, в форме начала работы можно настроить массив для задания списка исполнителей (эту переменную типа «массив» следует определить заранее). Задача разветвителя делается скрытой, а ветка – автоматической. Создаются локальные строковые переменные для сохранения текста замечаний и подписи текущего пользователя (последняя не обязательно).

Далее, если форма перехода от задачи к объединителю отображается, то текст из сообщения, с помощью действия над переменными, сохраняется в локальную переменную. Можно в форме перехода удалить предопределенное поле *msg_text* и вместо него отобразить локальную переменную. Если форма перехода не отображается, то текст набирается пользователем в форме задачи в поле этой переменной.

При выполнении перехода на объединитель, с помощью серверного действия аккумулируются значения локальных переменных с текстами замечаний пользователей в

глобальную переменную или в сообщение ближайшей задачи (объединителя). Можно объединитель сделать скрытым с помощью соответствующего действия (это предпочтительно). Тогда переход из него надо сделать автоматическим и обязательным условием надо сделать равенство количества входящих и исходящих веток. Если объединитель скрытый, то аккумулировать локальные тексты в сообщение объединителя нет смысла. Аккумулируйте их в глобальную переменную. На переходе от объединителя значение этой переменной можно, с помощью действия над переменными, передать в текст задачи следующей после объединителя.

Приведем пример действия над переменными, которое в сообщение следующей задачи Glob_text записывает замечания, вводимые пользователями в локальную переменную User_text:

```
Glob_text +
Задача.Имя_текущего_исполнителя + ':' + '~r~n' +
If (User_text=", 'Текст замечаний отсутствуют', User_text) + '~r~n' +
'===== ' + '~r~n'
```

Здесь набор символов '~r~n' означает перевод строки.

Таким образом, в задаче после объединителя исполнитель получит во входящем сообщении набор строк в виде, соответствующем приведенному нами действию. Например:

```
Иванов:
Текст замечаний1
=====
Петров:
Замечания отсутствуют
=====
Сидоров:
Текст замечаний2
=====
```

Рассмотрим второй вариант. Принцип организации процесса такой же, как и в первом варианте. Но сообщения пользователей аккумулируются в переменную с типом значения «Массив».

На этапе после объединителя следует настроить форму массива для просмотра (не редактирования) замечаний пользователей (Рисунок 3).

Массив значений переменной

№ п.п.	Пользователь	Дата	Замечания
1.	ГИП	12.06.2003	Замечаний нет
2.	Начальник отдела № 2	13.06.2003	Не хватает данных
3.	Начальник отдела № 3	13.06.2003	Замечаний нет

Ok Отмена

Рисунок 3 Пример формы массива

7 Динамическое управление ветвлением процесса

Разветвитель позволяет осуществлять разветвление в двух вариантах: по исполнителям и с повтором ветки. В первом варианте ветвление будет происходить по каждому основному исполнителю следующей задачи по данной ветке.

Ветвление с повтором ветки требует некоторых пояснений. Во-первых, в этом случае, количество повторов ветки определяется выражением. Во-вторых, если выражение ветвления не задано или возвращает нулевое значение, то ветка повторяется один раз. В-третьих, ветка будет повторяться заданное количество раз для каждого исполнителя задачи. Здесь, подчеркнем, что и список исполнителей задачи может быть назначен динамически, через массив переменной типа «Пользователь». Но, при переходе к задаче, все повторяющиеся значения исполнителей игнорируются, и задача отправляется исполнителю только один раз. Зачастую требуется одному исполнителю направить несколько задач в рамках одного разветвителя. Здесь, нам на помощь и приходит выражение для повтора ветки. Исполнитель может быть один, но количество повторов может быть любым.

Для наглядности рассмотрим пример ветвления с повтором ветки. На первом рисунке (Рисунок 4) изображен традиционный способ ветвления. Если для каждой ветки количество повторов равно единице, то каждый исполнитель задачи получит по одному предопределенному сообщению.

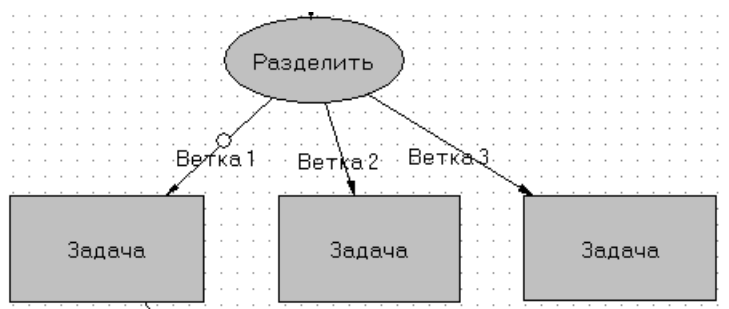


Рисунок 4

На втором рисунке (Рисунок 5) может быть изображен как традиционный способ ветвления без повтора ветки, так и с повтором ветки (равно, как и на первом рисунке). Из самого рисунка этого не видно (ветвление по исполнителям мы не рассматриваем). Но мы будем считать, что изображено ветвление с повтором ветки. В чем основное отличие? Первое – компактность шаблона. Второе – более сложная настройка, но более гибкое управление процессом ветвления, что и является предметом рассмотрения данного раздела.

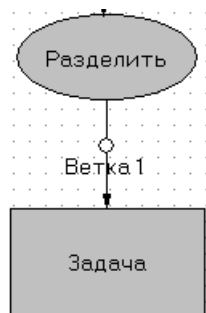


Рисунок 5

Поставим перед собой следующую задачу: осуществлять разветвление столько раз, сколько отмечено строк в форме массива. Исполнители ветвления будут задаваться в той же форме массива.

Решение задачи. Нам потребуются следующие переменные:

- массив типа «Пользователь» – исполнители ветвления по умолчанию. Назовем ее `branch_users`;
- массив типа «Пользователь» – реальные исполнители ветвления. Назовем ее `branch_users_1` и зададим одно любое значение по умолчанию;
- массив типа «Строка» – для задания темы задач. Назовем ее `ls_titles`;
- массив типа «Строка» – для ввода примечания. Назовем ее `ls_subj`;
- массив типа «Число» – для пометки тех задач, которые следует исполнять. Назовем ее `li_flags`;
- одиночная типа «Число» – счетчик веток. Назовем ее `counter`.

Вид шаблона, достаточный для демонстрации решения задачи – см. Рисунок 6

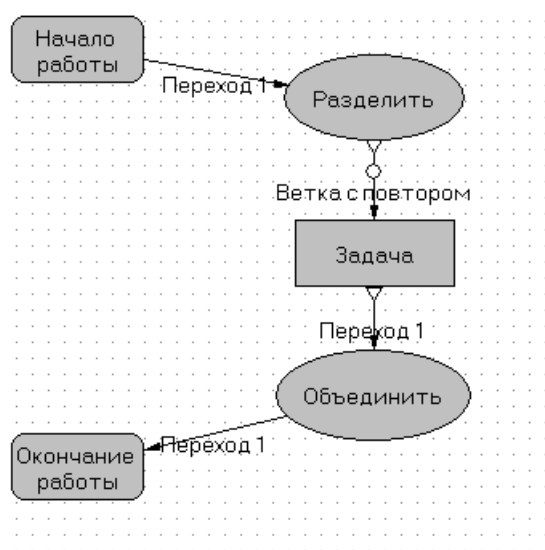


Рисунок 6

На разветвителе настроим форму массива для ввода информации (Рисунок 7).

№	Темы[...]	Примечания[...]	Исполнители ветвления[...]	Включ.[...]
1.	1-я тема	Проверить	Отдел кадров	<input checked="" type="checkbox"/>
2.	2-я тема	Согласовать	Отдел снабжения	<input type="checkbox"/>
3.	3-я тема	Согласовать	Пользователь №2	<input checked="" type="checkbox"/>
4.	4-я тема	Разослать информацию	Группа 3	<input type="checkbox"/>

Рисунок 7 Форма массива. Пример

Для переменной `li_flags` настроим стиль редактирования «Флажок». Включенный флажок возвращает значение 1, выключенный – 0.

На переходе от разветвителя настроим условия перехода и зададим выражение ветвления (Рисунок 8).

Рисунок 8

Выражение повтора ветки определяется количеством флажков, установленных в поле «Включ.[...]», то есть размерность числового массива, состоящего из элементов с значением, большим нуля.

Условие перехода на ветвление задает необходимость включения хотя бы одного флажка. Такое условие задавать не обязательно, но оно имеет демонстрационную цель.

Создадим на переходе (ветке) действия над переменными. Первое действие состоит из трех строк:

1. Установка значения счетчика веток *counter*. Содержание действия: *counter + 1*.
2. Установка темы следующей задачи. Содержание действия:

*Задача.Тема + ' № ' + string (Counter) + ' (' +
f_GetSArrEl(ls_Titles, f_GetNArrEl(f_GetArrElInd(li_flags , '@ArrEL > 0'), Counter)) + ')'*

Здесь, поясним, что тема следующей задачи формируется сцеплением текущей темы задачи с символом «№», номером ветки и с значением переменной *ls_titles*, соответствующей индексам строк с включенными флажками (*ls_flags* > 0). Это определяется следующим образом. Сначала определяются индексы строк со включенными флажками: *f_GetArrElInd(li_flags , '@ArrEL > 0')*. Полученный массив индексов является первым аргументом функции *f_GetNArrEl*. Вторым аргументом является переменная *counter*. Таким образом, формируется массив номеров тем, которые надо выполнить. Этот массив, в свою очередь, является вторым (числовым) аргументом функции *f_GetSArrEl*. Первым аргументом является переменная-массив *ls_Titles*.

3. Установка текста следующей задачи. Содержание действия:

f_GetSArrEl(ls_subj, f_GetNArrEl(f_GetArrElInd(li_flags , '@ArrEL > 0'), Counter))

Действие полностью аналогично предыдущему, за исключением того, что извлекаются значения переменной-массива *ls_subj*.

Второе действие предназначено для формирования массива исполнителей ветвления и построено аналогично действиям по формированию темы и текста задачи, но первой строкой действия мы удаляем заданное нам значение по умолчанию. Значение по умолчанию нам нужно было для того, чтобы при попытке разветвления был хотя бы один исполнитель следующей задачи.

Аргумент	Значение
branch_users_1[...]	f_ExcludeArrEl(branch_users_1 , 'ALL')
branch_users_1[...]	f_GetNArrEl(Branch_Users , f_GetNArrEl(f_GetArrElInd(li_flags , '@ArrEL > 0'), Counter))

Рисунок 9

Отметим, что переменная *branch_users_1* является исполнителем задачи после разветвителя.

8 Реализация напоминаний

8.1 Вариант 1. Автоматическая генерация свободных сообщений из невыполненной задачи

Задача: требуется реализовать автоматическую генерацию свободных сообщений из невыполненной задачи, чтобы обеспечить напоминание пользователям.

Решение: используется следующая функциональная особенность Lotsia PDM PLUS: «Действия по событию «После выполнения задачи» ВСЕГДА выполняются перед проверкой возможности выполнения автоматического перехода из задачи сервером автоматических этапов, если они расположены ПЕРЕД встроенным действием «Подготовка переходов»».

Особенность решения: в карте работы никаких переходов не выполняется, генерируемые сообщения не связаны с картой работы и контролируемой задачей, содержание отчета «Контроль исполнения» не изменяется. При получении сообщения и необходимости обращения к задаче, пользователь самостоятельно должен найти данную задачу.

Реализация:

1. На контролируемой задаче делается автоматический переход «сам на себя» с заведомо невыполнимым условием. Пример такого условия: $1=2$.
2. На контролируемой задаче перед встроенным действием «Подготовка переходов» добавляется клиентское действие над объектами, которое отправляет с помощью функции MailSend свободное сообщение заданному адресату с заданными параметрами от имени пользователя, под которым сервер автоматических этапов подсоединен к базе данных. У этого действия необходимо установить условие выполнения, что задача не открыта пользователем. Такое условие можно сформировать с использованием какой-либо переменной, значение которой изменяется действием по событию «После открытия задачи» и сбрасывается действием по событию «Перед закрытием задачи». Таким образом, если значение переменной не изменено, то задача не открыта. Также может потребоваться с помощью обратного условия обеспечить выполнение других действий (при их наличии), расположенных в событии «После выполнения задачи» перед встроенным действием «Подготовка переходов» только, если задача открыта пользователем.
3. Очевидно, что не имеет смысла генерировать сообщения при каждой проверке очереди сервером автоматических этапов. Поэтому на всех переходах к контролируемой задаче добавляется действие, задающее интервал и/или дату проверки задачи сервером автоматических этапов.

Работает это следующим образом: сервер автоматических этапов в заданное действием время проверяет автоматический переход. Его условие заведомо невыполнимое, но перед проверкой выполняются действия, расположенные перед встроенным действием «Подготовка переходов». Будут выполнены действия, условия которых выполняются, в том числе, действие отправляющее сообщение.

8.2 Вариант 2. Автоматическое повторное поступление невыполненной задачи

Задача: требуется реализовать автоматическое поступление невыполненной задачи, чтобы обеспечить напоминание исполнителю задачи.

Решение: в контролируемой задаче используется автоматический переход «сам на себя».

Особенность решения: при выполнении условия перехода, контролируемая задача поступает пользователю вновь. В отчете «Контроль исполнения» появится новая соответствующая запись. При получении задачи пользователю нет необходимости предпринимать дополнительных действий для ее поиска.

Реализация (рассматривается минимально достаточный вариант):

1. На контролируемой задаче делается автоматический переход «сам на себя». Необходимость условия перехода определяется конкретикой шаблона работы, в котором реализуется напоминание.
2. На всех переходах к контролируемой задаче, включая переход «сам на себя», добавляются действия, определяющие дату первой проверки и интервал проверки задачи сервером автоматических этапов. Очевидно, что до наступления определенного срока не имеет смысла проверять переход «сам на себя», нагружая при этом сервер автоматических этапов. Поэтому первой датой проверки рекомендуется поставить либо дату срока исполнения, либо некоторую дату, полученную расчетным путем.

Работает это следующим образом: сервер автоматических этапов в определенное действие время выполняет автоматический переход «сам на себя». Пользователь получит задачу и если не выполнит ее до следующего, определенного действием, времени проверки, сервер автоматических этапов вновь выполнит переход «сам на себя».

Варианты развития реализации:

На переходе «сам на себя» возможно выполнение действий, формирующих значения различных переменных, например, счетчика количества напоминаний, текста напоминания. Также возможно выполнение действий над объектами, например, для записи информации в атрибуты каких-либо объектов или отправки свободных сообщений. Одновременно с выполнением перехода возможно выполнение информационной рассылки контролирующим пользователям. Но в этом случае необходимо установить условие выполнения такой рассылки, чтобы заблокировать выполнение рассылки при выполнении других переходов. Реализовать это можно следующим образом:

1. На переходе «сам на себя» ПЕРЕД встроенным действием <Ветвление переходов> добавляется действие, формирующее признак выполнения именно этого перехода. Используется переменная шаблона работы. Например, flag=1. Следует иметь в виду, что начальное значение переменной flag не должно быть равно 1.
2. На переходе к информационной рассылке устанавливается условие, используя ту же переменную и установленное ей значение согласно предыдущему пункту (flag = 1).
3. На переходе к информационной рассылке добавляется действие, сбрасывающее признак выполнения перехода. Например, flag=0.

9 Распределение строк массивов между исполнителями

При реализации бизнес-процессов часто встречается ситуация, когда одновременно формируется массив заданий (задач, документов и др.), каждому элементу которого назначается отдельный исполнитель. В таком случае, типичной будет ситуация, когда одному и тому же исполнителю в рамках одного процесса сопоставлено несколько заданий (задач, документов и др.). Случай, когда по каждой строке массива необходимо сформировать отдельную задачу, рассмотрен в разделе «Динамическое управление ветвлением процесса».

Здесь рассмотрим ситуацию, когда каждый исполнитель должен получить единственное сообщение, в котором перечислены все сопоставленные ему задания (задачи, документы и др.). Типичный пример подобной ситуации – рассылка информации об изменениях, проведенных в комплекте документации абонентам указанных документов. Список абонентов для каждого из документов различен, но абоненты для различных документов большей частью повторяются.

Список рассылки будет состоять минимум из двух колонок – Абонент (получатель рассылки) `user_isr` (массив, локальный, тип данных – пользователь) и обозначения документов `obozn_doc` (массив, локальный, тип данных – строка). Могут быть и другие колонки. Если выполнить рассылку отдельно по каждой строке (см. «Динамическое управление ветвлением процесса»), каждый абонент получит большое количество сообщений (по числу заданий, которым он сопоставлен).

Для обеспечения формирования единственного сообщения каждому получателю, при настройке шаблона работы (см. Рисунок 10) рекомендуется поступить следующим образом:

- массив документов поместить в форму задачи абонента;
- массив абонентов назначить исполнителем данной задачи;
- использовать разветвление по исполнителям при переходе к данной задаче;
- на переходе к задаче абонента формировать локальные массивы по числу требуемых колонок списка, соответствующие конкретному абоненту.



Рисунок 10 Рассылка. Фрагмент шаблона работы

Формирование локального массива (или массивов, по числу колонок) производится следующим действием над переменными:

- на переходе к задаче Абонента (используется разветвление по исполнителям) добавим действие над переменными;
- определим конкретного исполнителя данной задачи – при выполнении перехода уже известно, какому исполнителю `user_local` направляется задача:

`user_local` | Переход.Задача_приемник.ID_текущего_исполнителя

- определим массив номеров строк `m_num_del`, не относящихся к данному исполнителю:

```
m_num_del[...] [f_GetArrElInd( user_isp , '@ArrEL<> '+user_local )
```

- удалим из локального массива обозначений документов, направляемого данному абоненту, не относящиеся к данному абоненту строки и получим в форме задачи абонента только соответствующие абоненту строки массива `obozn_doc`:

```
obozn_doc [...] [f_ExcludeArrEl( obozn_doc , m_num_del )
```

- при большем числе колонок в списке рассылки повторим указанную строку для остальных колонок.

Если после получения абонентами рассылки требуется, например, получение общего результата обработки документов абонентами, то есть обратное объединение разделенных локальных массивов, рекомендуется использовать глобальные массивы для аккумуляции в них данных из локальных разделенных массивов (см. «Динамическое управление ветвлением процесса»).

10 Выбор пользователей

Задача выбора пользователей является актуальной в большинстве случаев (см. также «Назначение исполнителей», «Подписание документов»), поэтому рассмотрим некоторые аспекты решения данной задачи отдельно.

10.1 Определение пользователя-начальника отдела (группы, сектора и т.п.)

Постановка задачи: пользователи разбиты на группы в соответствии с организационной структурой предприятия, например, на отделы, группы, сектора... Требуется определить руководителя структурной единицы для какой-либо группы.

Реализация: логично предположить, что существует группа сотрудников отдела и группа начальников отделов (групп, секторов и т.п.). Нам требуется найти такого пользователя, который входит одновременно в две группы: группа отдела и группа начальников отделов (групп, секторов).

Создадим две переменные массива типа «Пользователь». Одна переменная со значением группы «Начальники отделов» и с именем Dep. Другая переменная со значением группы «Отдел» и с именем St. Для определения общего пользователя следует произвести следующие операции:

1. Развернуть группы Dep и St.

f_ExpandGroups(Dep, 1) и f_ExpandGroups (St, 1)

2. Сравнить развернутые составы группы Dep и группы St.

f_DifferenceOfArrs (f_ExpandGroups(Dep, 1), f_ExpandGroups (St, 1))

3. Сравнить результат сравнения из п.2 с массивом Dep.

f_DifferenceOfArrs (f_ExpandGroups(Dep, 1), f_DifferenceOfArrs (f_ExpandGroups(Dep, 1), f_ExpandGroups (St, 1)))

Создадим действие над переменными с содержанием, соответствующим п.3:

f_DifferenceOfArrs (f_ExpandGroups(Dep, 1), f_DifferenceOfArrs (f_ExpandGroups(Dep, 1), f_ExpandGroups (St, 1)))

10.2 Использование проекта «Структура предприятия»

Наибольшая гибкость в выборе сотрудников может быть достигнута при ведении отдельного проекта «Структура предприятия» (Рисунок 11).

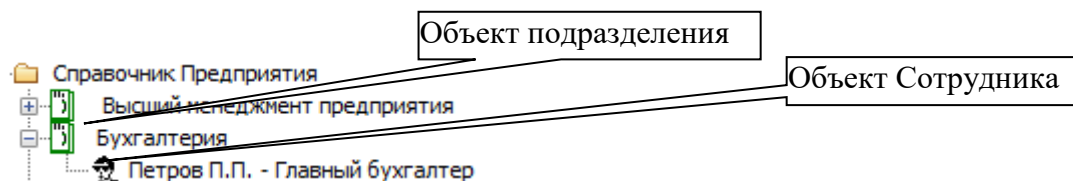


Рисунок 11. Структура Предприятия. Фрагмент

Объекту сотрудника сопоставляется пользователь БД. Сопоставление осуществляется путем записи ID пользователя в атрибут объекта «Сотрудник».

Для объекта сотрудника устанавливается необходимый набор атрибутов, в том числе: «Фамилия», «Имя», «Отчество», «Должность основная», «Телефон» и т.д.

Выбор объекта сотрудника из данного проекта возможен с применением [различных способов](#). Однако, можно указать несколько моментов, облегчающих поиск объектов-сотрудников в следующих типовых случаях:

- выбор всех сотрудников данного подразделения, пола, должности, возраста и т.п. Такой выбор удобно осуществлять заданием условий отбора по значению конкретных атрибутов объекта сотрудника, то есть целесообразно хранить подобную информацию в соответствующих атрибутах сотрудника: «Номер основного подразделения», «Пол», «Основная должность», Дата рождения и т.д.
- выбор начальника данного сотрудника или всех подчиненных данного сотрудника. Тут возможны различные варианты решения.

При слабой формализации взаимоотношений, можно рекомендовать использование связей соответствующих типов между объектами сотрудников. Например, использовать типы связей: «Прямая подчиненность», «Подчиненность в ситуации №1» и т.д., что позволяет произвольным образом настраивать отбор подчиненных и начальников в различных ситуациях.

Если организация функционирует на основе четких правил и подчиненность определена должностью сотрудника, более удобным может оказаться отбор сотрудников по атрибутивной информации. Можно использовать несколько атрибутов, например: «Направление деятельности», «Специализация», «Подразделение», «Категория» и т.п. Ту же информацию можно хранить в единственном строковом атрибуте стандартного формата – «Код основной должности сотрудника». Используя строковые функции, из кода должности можно получать отдельные позиции и задавать их значение в условиях отбора объектов сотрудников.

10.3 Множественный выбор исполнителей (адресатов)

Постановка задачи: существует справочник предприятия. Подразделения и сотрудники хранятся в виде двухуровневого дерева проекта. Первый уровень – подразделения, второй уровень – сотрудники. Требуется реализовать множественный выбор сотрудников-исполнителей (адресатов) из иерархического списка подразделений, открывающегося в отдельном окне, и сформировать на основании сделанного выбора список пользователей-получателей. Выбранные исполнители должны исключаться из списка выбора. Реализовать данную функцию можно различными способами:

Реализация 1: (Рекомендуется)

Использование действий над объектами с окнами выбора объектов. В настройках окна выбора из дерева включите флажок «Множественный выбор», в условиях построения дерева или списка выбора ввести выражение, исключающее выбранных исполнителей. Например, если список ID объектов выбранных исполнителей (разделитель – запятая) передать в строковую переменную действия `str_iskl`, выражение может быть таким – см. Рисунок 12.

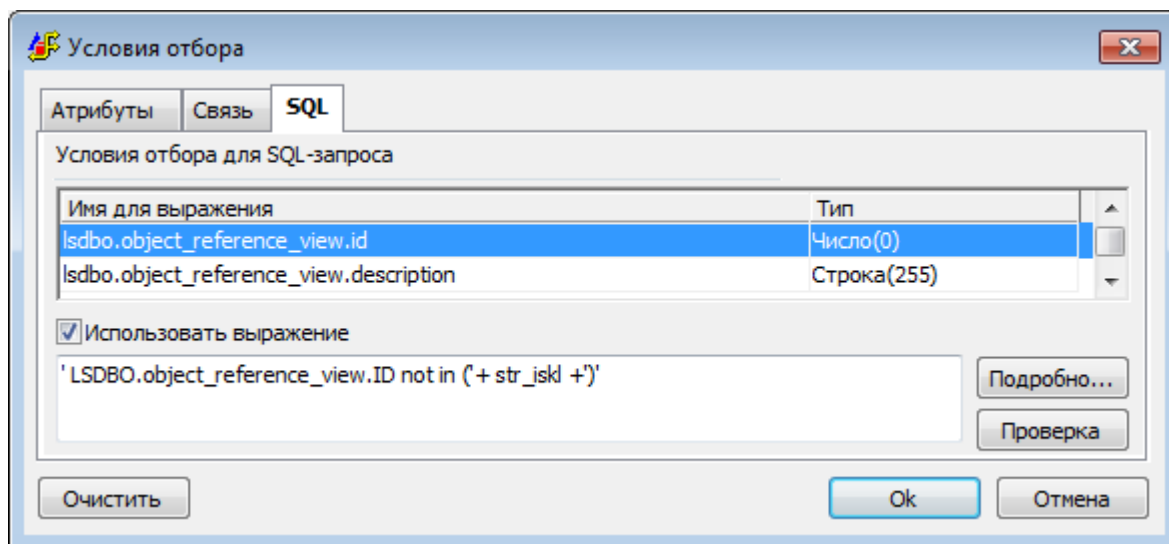


Рисунок 12. Условия отбора, исключающие ранее выбранные объекты. Пример

При работе с массивами (при настройке средств документооборота) для формирования из массивов строк через разделитель и развертывания строки через разделитель в массив удобно использовать функции `f_ArrayToString` и `f_StringToArray`. Для изменения типа данных элементов массива (например, со строки на число) удобно использовать агрегатную функцию – см. раздел. «Особенности использования агрегатных функций для массивов значений».

Реализация 2: (Подразумевается, что использование средств документооборота, сколько-нибудь значимых преимуществ перед рассмотренной выше реализацией не имеет, за исключением других возможностей настройки интерфейса. Более трудоемка при настройке в сравнении с реализацией 1.)

В форме задачи две формы переменных-массивов. Первая – форма для выбора (`forma_select`), вторая – форма, в которую возвращается выбор (`forma`).

Действие над переменными «Заполнить массив выбора» – после открытия окна формы массива `forma_select`. В действии три строки:

Строка 1 (запомним уже добавленных исполнителей):

```
arg_users = if (f_GetArrUpperBound ( users ) = 0, f_ExecAgregateEXP( f_StringToArray (
'0', '' ), 'number (@ArrEL)' ), users )
```

Строка 2:

```
ls_tmp = f_EmbeddedSQLSelect ( this,
'select obj.description, ~~, ~~'
from lsdbo.object_reference_view obj
where obj.type_id = 100000003044037 /*Отдел*/
union all
select obj1.description , obj2.description, vs.value
from lsdbo.tree_link_view tl, lsdbo.object_reference_view obj1, lsdbo.object_reference_view
obj2, LSDBO.Attrib_Value_S_v vs
where tl.parent_id = obj1.id
and tl.link_id = obj2.id
and tl.link_type_id = 1
and obj1.type_id = 100000003044037 /*Отдел*/
and obj2.type_id = 100000003144037 /*Сотрудник*/
and vs.object_id = obj2.id
and vs.attr_id = 100000006544037 /*ID пользователя*/
and vn.value not in (:arg_users ) /* Исклучим уже выбранных исполнителей*/
```

```

order by 1'
, 'arg_users,numberlist' , 'forma_select' )
+ f_RetrieveForm ( this , 'arg_users' , 'forma_select' )
+ f_FormDataToArray ( this , 'forma_select' )

```

Здесь, forma_select – форма массива, открывающегося в отдельном окне.

Атрибут ID пользователя строковый. Если нужно переделать на числовой, то запрос меняется на следующий:

```

ls_tmp = f_EmbeddedSQLSelect ( this ,
'select obj.description, ~~, -1
from lsdbo.object_reference_view obj
where obj.type_id = 100000003044037 /*Отдел*/
union all
select obj1.description , obj2.description, vn.value
from lsdbo.tree_link_view tl, lsdbo.object_reference_view obj1, lsdbo.object_reference_view
obj2, LSDBO.Attrib_Value_N_v vn
where tl.parent_id = obj1.id
and tl.link_id = obj2.id
and tl.link_type_id = 1
and obj1.type_id = 100000003044037 /*Отдел*/
and obj2.type_id = 100000003144037 /*Сотрудник*/
and vn.object_id = obj2.id
and vn.attrib_id = 100000006544037 /*ID пользователя*/
and vn.value not in (:arg_users ) /* Исключим уже выбранных исполнителей*/
order by 1'
, 'arg_users,numberlist' , 'forma_select' )
+ f_RetrieveForm ( this , 'arg_users' , 'forma_select' )
+ f_FormDataToArray ( this , 'forma_select' )

```

Представленный пример SQL-запроса должен быть модифицирован с точки зрения идентификаторов типов объектов и атрибута с ИД пользователя. Также могут быть изменены и условия отбора применительно к конкретной ситуации, например, для выбора сотрудников конкретного отдела.

Строка 3 (устанавливаем заголовок окна выбора):

```
ls_tmp = f_ModifyWin ( this , 'Window' , 'Title="Выберите исполнителей"' )
```

Настройка формы массива forma_select (открывается в отдельном окне, сетка, запрещено добавление, удаление, изменение элементов, разрешено выделять несколько строк):

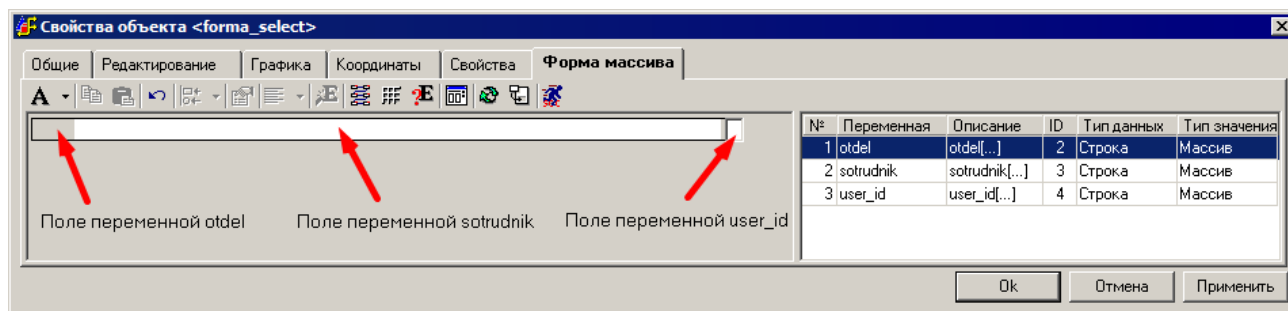


Рисунок 13 Форма массива forma_select

Типы данных и порядок следования переменных виден на картинке. Переменная с ИД пользователя (user_id) в нашем примере строковая, в соответствии с запросом. Если атрибут, хранящий код пользователя числовой, то переменную user_id следует создать как числовую. Все переменные с типом значения «массив».

В области заголовка формы удалены все элементы и высота области равна 0. В форме выполнена группировка по полю otdel. Высоты областей заголовка и итогов группы 1 тоже равны 0. Поле sotrudnik наложено на поле otdel с небольшим сдвигом. Для всех полей установлено свойство защищенности, равное 1. Для поля otdel установлено условие свойства видимости if (sotrudnik = " or IsNull (sotrudnik), 1, 0). Для поля sotrudnik установлено обратное условие свойства видимости if (sotrudnik = " or IsNull (sotrudnik), 0, 1). Для поля user_id установлено свойство видимости, равное 0.

Действие над переменными «Вернуть выбор в форму» – перед закрытием окна формы массива forma_select. В действии 2 строки.

Строка 1:

Вернуть описание сотрудника:

```
isp = f_DifferenceOfArrs ( sotrudnik , f_ExcludeArrEl ( sotrudnik , f_GetSelRowsForm ( this , 'forma_select' ) ) )
```

Строка 2:

Вернуть пользователя, сопоставленного сотруднику:

```
users = f_ExecAgregateEXP ( f_DifferenceOfArrs ( user_id , f_ExcludeArrEl ( user_id , f_GetSelRowsForm ( this , 'forma_select' ) ) ) , 'number (@ArrEL)' )
```

Если ИД пользователя числовой, то функция f_ExecAgregateEXP не нужна и строка 2 примет следующий вид:

```
users = f_DifferenceOfArrs ( user_id , f_ExcludeArrEl ( user_id , f_GetSelRowsForm ( this , 'forma_select' ) ) )
```

Кроме того, рекомендуется предусмотреть исключение из возвращаемых в форму массивов элементов с некорректными идентификаторами пользователей (пустым или равным -1).

Настройка формы массива forma (внедренный):

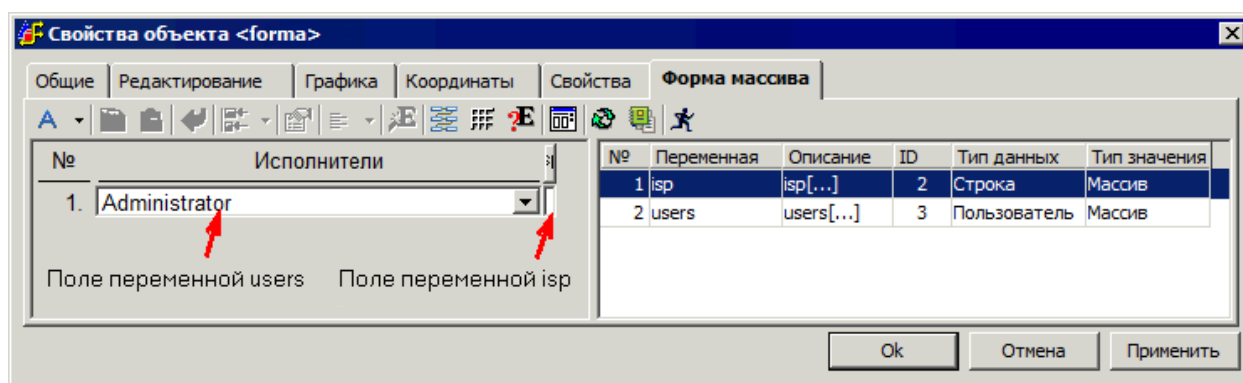


Рисунок 14 Форма массива forma

Фактически, в форме нужна только одна переменная-пользователь (users). Но для контроля можно оставить и переменную с описанием сотрудника (isp) или, например, вместо описания сотрудника возвращать в форму должность. В форме для нее и ее заголовка установлено свойство видимости, равное 0 и минимальную ширину. Если же

поле будет убрано из формы, то нужно убрать и строку 1 из действия «Вернуть выбор в форму».

После привязки к задаче, список действий выглядит следующим образом (на рисунке показаны только рассматриваемые действия, но их может быть и больше):

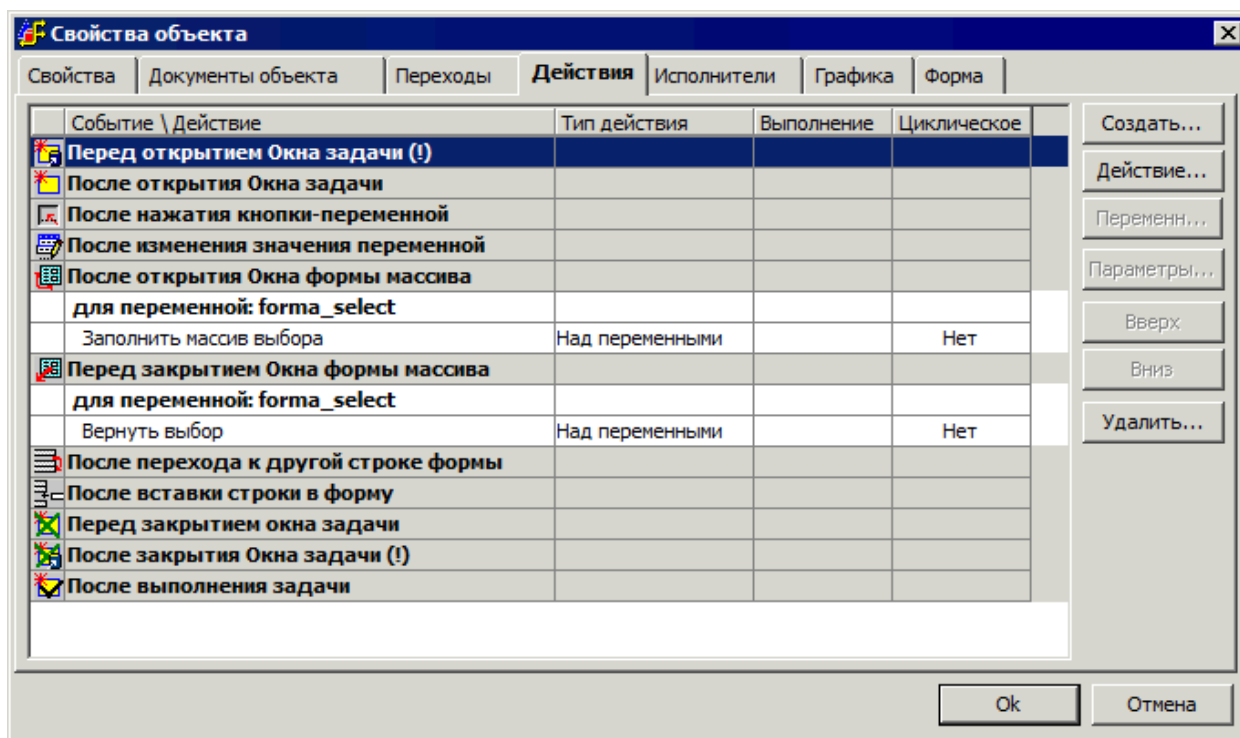


Рисунок 15 Привязка действий к событиям задачи

Настройка формы задачи в простейшем виде – из формы удалены все лишние поля и оставлены только поля рассматриваемых форм массивов:

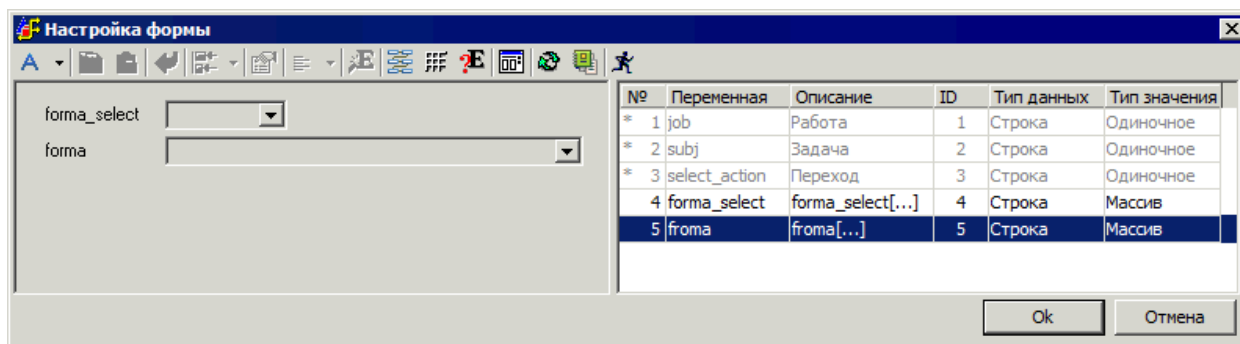


Рисунок 16 Пример настройки формы

После запуска задачи наша форма будет выглядеть так:

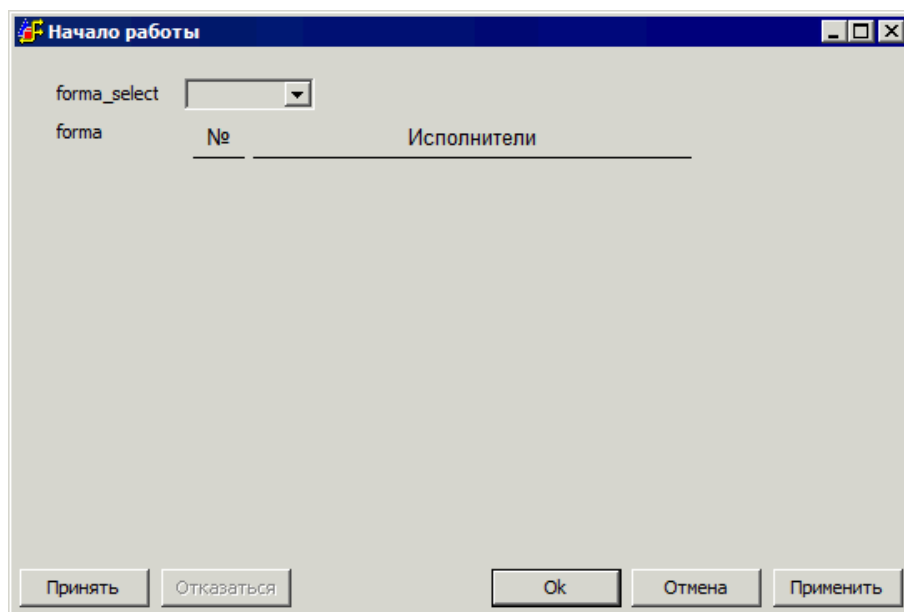


Рисунок 17 Форма задачи на этапе исполнения

Пользователь щелкает мышью по полю переменной `forma_select`. Открывается форма выбора. Выбираем одну или несколько строк и щелкаем кнопку «**Ok**» (в нашем примере в департаменте производства, отделе информационных технологий и в отделе продаж нет сотрудников).

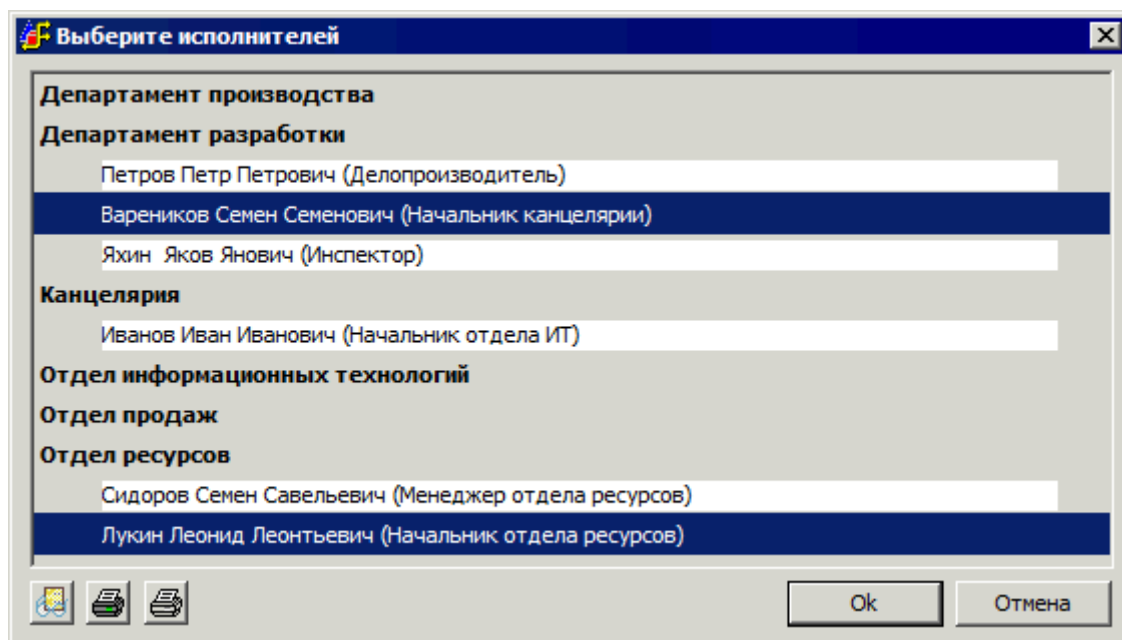


Рисунок 18 Окно выбора на этапе исполнения

Выбор вернется в форму задачи:

Начало работы

forma_select [dropdown]

forma

№	Исполнители
1.	Вареников Семен Семенович [dropdown]
2.	Лукин Леонид Леонтьевич [dropdown]

Принять Отказаться Ok Отмена Применить

Рисунок 19 Возвращенный список исполнителей

Более удачной реализацией, с точки зрения интерфейса, видится открытие формы выбора по нажатию кнопки (а не поля переменной `forma_select`). Для реализации потребуется настроить в форме кнопку. Тогда для поля переменной `forma_select` можно установить свойство видимости, равное 0. И тогда по событию «Нажатие кнопки-переменной» нужно будет выполнить действие над переменными следующего содержания:

```
f_Open_FreeForm ( this , 'forma_select' )
```

Очевидно, что нужно предусмотреть и возможность удаления выбранных исполнителей.

Для этого достаточно в списке исполнителей добавить переменную-кнопку и соответствующим образом модифицировать действие по возврату выбора, добавив в него еще одну возвращаемую колонку, например, с символом «X» и установив красный цвет шрифта кнопки.

```
button_del = f_ExecAgregateEXP( isp, '~'X~" )
```

Соответственно, к событию по нажатию этой кнопки следует привязать действие по удалению из массивов значений с индексами, равными номеру текущей строки.

11 Особенности использования внешних работ и информационных рассылок

Применение в шаблонах работ объектов «Информационная рассылка» позволяет направить получателям рассылки статичное сообщение с информацией, зафиксированной на определенный момент времени. Исключение составляет использование глобальных переменных, текущие значения которых автоматически применяются во всех объектах карты работы.

Однако в ряде случаев требуется создать сообщения информационной рассылки, поддерживающие обновление информации и включающие такие элементы интерфейса, как кнопки, переключатели и т.д. В этих случаях вместо объекта «Информационная рассылка» можно использовать объект «Внешняя работа». Однако следует учитывать, что переменные родительской работы передаются во внешнюю работу только один раз.

В шаблоне внешней работы используется только форма начала работы, в которой с помощью функции `f_ModifyWin` отключены стандартные кнопки. По событию «После открытия окна задачи» возможно выполнение действий, обновляющих данные в форме начала работы, используя SQL-запрос и/или действия над объектами или переменными (объекты или документы могут быть переданы во внешнюю работу из родительской). Также возможно использование в действиях системных переменных. В этой ситуации, при обращении к внешней работе каждый раз производится ее запуск, но сама работа не сохраняется в списке карт работ, поскольку после получения необходимой информации пользователь закрывает форму нажатием кнопки «Отмена» или системной кнопкой закрытия Окна.

12 Циклическая одновременная обработка двух массивов

В некоторых случаях требуется выполнить действия над элементами одного массива, используя в качестве аргумента также массив значений. Практическим примером данной ситуации может служить задача единовременной установки прав доступа к нескольким объектам для нескольких пользователей. Объекты представлены массивом идентификаторов объектов «Obj_id», пользователи – массивом пользователей «Users».

Права устанавливаем действием над объектом (само действие над объектом рассматривать нет необходимости), на вход которому подаем один из элементов массива пользователей и один из элементов массива объектов. Задача заключается в том, чтобы, циклически перебирая элементы массивов, выполнить действие со всеми возможными парами элементов массивов Obj_id и Users.

Действие выполняется циклически, извлекается и передается на вход очередной элемент из массива Users:

```
f_GetNArrEl (Users, Ceiling (f_GetCycleCounter(this)/f_GetArrUpperBound(Obj_id)))
```

Извлекается и передается на вход очередной элемент из массива Obj_id:

```
f_GetNArrEl(Obj_id, 1+(f_GetCycleCounter(this)-Int(f_GetCycleCounter(this)/  
f_GetArrUpperBound(Obj_id))*f_GetArrUpperBound(Obj_id)))
```

Условие выполнения действия:

```
f_GetCycleCounter(this)<=f_GetArrUpperBound(Obj_id)*f_GetArrUpperBound(Users)
```

При выполнении перебираются все возможные пары элементов массивов Obj_id и Users. Данный пример является одним из возможных вариантов решения рассмотренной задачи.

13 Особенности использования агрегатных функций для массивов значений

В [Редакторе выражений](#) допускается использовать агрегатные функции. Например, Max, Min, Percent и т.п. В случае использования этих функций в вычисляемых полях форм проблем не возникает. Но использование агрегатных функций в действиях над переменными с массивом значений имеет свои особенности. А именно: к массиву применяется специальная функция f_ExecAgregateEXP. Синтаксис этой функции следующий:

f_ExecAgregateEXP(<ArrVariable> , 'Max (@ArrEL for All)')

Функция возвращает массив, тип данных которого зависит от агрегатной функции, являющейся вторым аргументом. Агрегатная функция должна быть оформлена как строка, то есть заключена в кавычки. Функция выполняет над первым аргументом-массивом выражение с агрегатными функциями и возвращает массив значений полученных в результате данного выполнения. Число элементов возвращаемого массива равно числу элементов массива-аргумента. Значения возвращаются следующим образом:

1. Используются агрегатные функции типа Max или Min, возвращающие одно значение, например, 'Max (@ArrEL for All)'. Здесь, поскольку число элементов возвращаемого массива равно числу элементов массива-аргумента, будет возвращен массив, в котором значения всех элементов будут одинаковы (равны значению максимального элемента массива-аргумента). Тип итогового массива будет тот же, что и у массива-аргумента. Чтобы получить, например, одно значение максимального элемента некоторого числового массива n1 достаточно составить такое выражение: f_GetNArrEl (f_ExecAgregateEXP(n1, 'Max (@ArrEL for All)'), 1).
2. Используются агрегатные функции типа Percent, возвращающие ряд значений, например, 'Percent (@ArrEL for All)'. Здесь, n-ый элемент итогового массива будет содержать процентное отношение текущего значения элемента массива-аргумента к сумме всех значений элементов этого массива. В этом случае, значения элементов итогового массива, скорее всего, будут различаться.
3. Вторым аргумент может быть и не агрегатным и выполнять какие-то операции над первым аргументом, например, функция
f_ExecAgregateEXP(<Number Arr>, '~'[~] + String(@ArrEl * 2) + ~'[~]~')
из числового массива:
1, 2, 3, 4
вернет строковый массив
'[2]', '[4]', '[6]', '[8]'

14 Операции над документами работы

В шаблоне работы могут задаваться документы – объекты или документы архива. Они могут быть как с одиночным значением, так и со значением типа «массив». В ходе выполнения работы может понадобиться очистить значение документа или перебрать массив и часть документов (или все документы) удалить из массива.

Реализация: действие над объектами. В действии могут настраиваться формы шагов, в которых будет отображаться документ, с которым ведется работа, а также поле выбора дальнейших действий. Например, «Очистить» или «Перейти к следующему документу». На тех шагах, которые соответствуют выбору «Очистить», следует использовать функцию сброса значения переменной «Null». На других шагах могут выполняться какие-либо другие функции. Если в поле с документом выбрать другой документ, то первоначальный документ будет заменен на выбранный.

После создания действия над объектами следует правильно его включить в шаблон. Добавьте действие. Добавьте переменную действия (типа «объект» или «версия документа»). В колонке «Значение» указывается документ работы, который передается в действие. Если указан массив, то действие выполнится столько раз, сколько элементов находится в массиве. В колонке «Возврат» указывается тот же документ работы (хотя может быть указан и другой). Смотрите Рисунок 20.

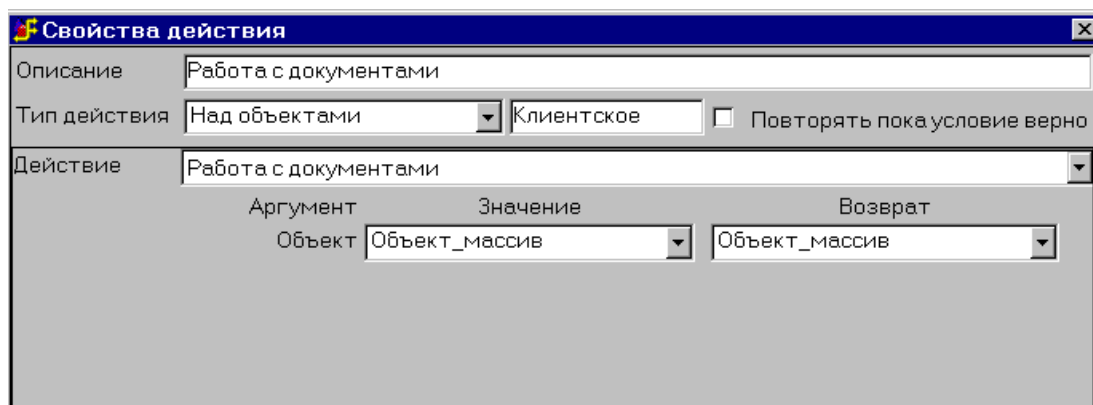


Рисунок 20 Пример действия по прикреплению объекта к работе

Эта комбинация работает следующим образом. При запуске действия, в переменную «Объект» (Рисунок 20) на вход подается первый элемент массива. Далее действие обрабатывается в соответствии со своими настройками. Если элемент обнуляется (функцией Null), то обратно возвращается пустое значение и переданный элемент из массива удаляется. Если элемент сохранился или изменился, то он возвращается обратно в массив. Другими словами, на вход переменной «Объект» последовательно передаются элементы массива. Обратно в массив также последовательно возвращаются значения переменной «Объект», но уже после выполнения действия.

Таким же образом можно организовать выборочное перемещение документов из одного массива в другой. Для этого достаточно на вход действия подавать элементы одного массива, а результат возвращать в другой массив. Это может использоваться, например, для решения задачи группировки согласованных и несогласованных документов по разным массивам.

15 Импорт документов-файлов predeterminedного процесса в архив

В некоторых случаях удобно выполнять импорт приложенных документов-файлов в архив автоматически, в ходе выполнения работы. В этом случае, при настройке шаблона работы используется действие над объектами, производящее импорт документа-файла в архив. В действии над объектом импорт выполняется с помощью функции DocImport, например:

DocImport (a_Object , file_imp , file_name , biblioteka)

В приведенном примере функции используются следующие переменные действия над объектом:

- a_Object – объект, в который осуществляется импорт документа;
- file_imp – полное имя импортируемого файла, включая путь. Строка не должна содержать кавычек, то есть должна быть обработана соответствующим образом;
- file_name – описание создаваемого документа архива;
- biblioteka – библиотека, в которую осуществляется импорт документа.

Если требуется, функция DocImport может содержать и другие параметры, см. [Редактор действий](#).

Для выполнения импорта следует в каждой итерации цикла в качестве входного параметра действия над объектом сформировать строковое значение, содержащее очередное полное имя файла для соответствующего документа шаблона. Например, для документа с описанием «Документ_файл_массив» и с таким же именем для выражения, в качестве входного значения действия следует указать (см. Рисунок 21):

f_GetSArrEl (f_GetTemplDocFiles (this , 'Документ_файл_массив'), f_GetCycleCounter (this))

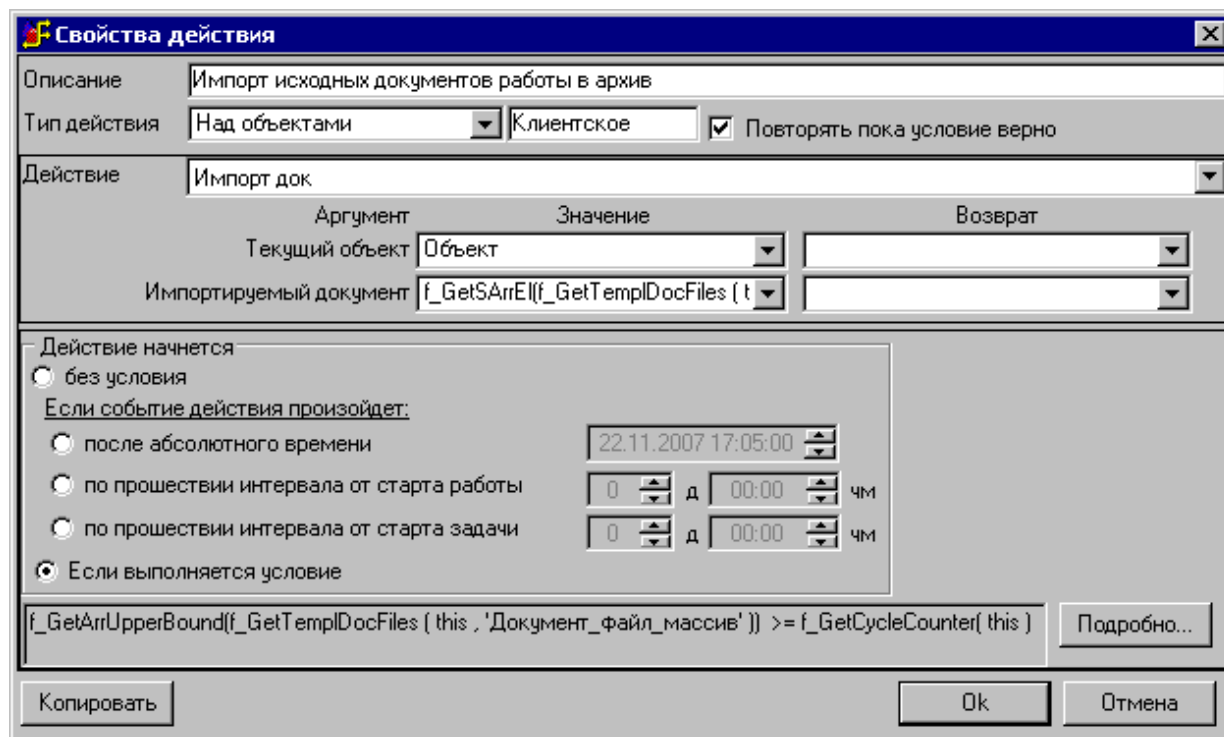


Рисунок 21 Импорт документов-файлов шаблона в архив. Пример подключения действия над объектом

Действие должно быть циклическим, если документ-файл шаблона работы задан с типом значения «Массив». Условие выполнения действия:

```
f_GetArrUpperBound (f_GetTemplDocFiles (this, 'Документ_файл_массив')) >= f_GetCycleCounter (this)
```

Если документ-файл шаблона работы задан с типом значения «Одиночное», то цикличность и условие выполнения действия можно отключить (но не обязательно).

16 Защита от импорта документов в определенные объекты

Запретить импорт документов в определенные объекты требуется для того, чтобы пользователь случайно не поместил документ в объект, который не предусматривает наличие документа. Пользователь может это сделать, например, через сохранение из интегрированных приложений, ошибочно выбрав неверный целевой объект. Например, вместо чертежа пользователь может ошибочно выбрать папку. Впоследствии, доступ к документу будет затруднен, так как у объектов-папок может не быть формы документов архива.

Задачу можно решить с помощью назначений прав доступа, но, во-первых, это далеко не всегда можно удобно реализовать, а во-вторых, права доступа могут ограничить пользователя в выполнении других необходимых операций. Кроме того, в данном случае требуется всего лишь защита от ошибочного действия.

Реализовать такую защиту легко можно с помощью бизнес-правил. Необходимо создать бизнес-правило типа «Действие» для контроля действия «Изменение документа» (Рисунок 22) и привязать его ко всем типам объектов (вкладка «Привязка»), в которые следует запретить импортировать документы.

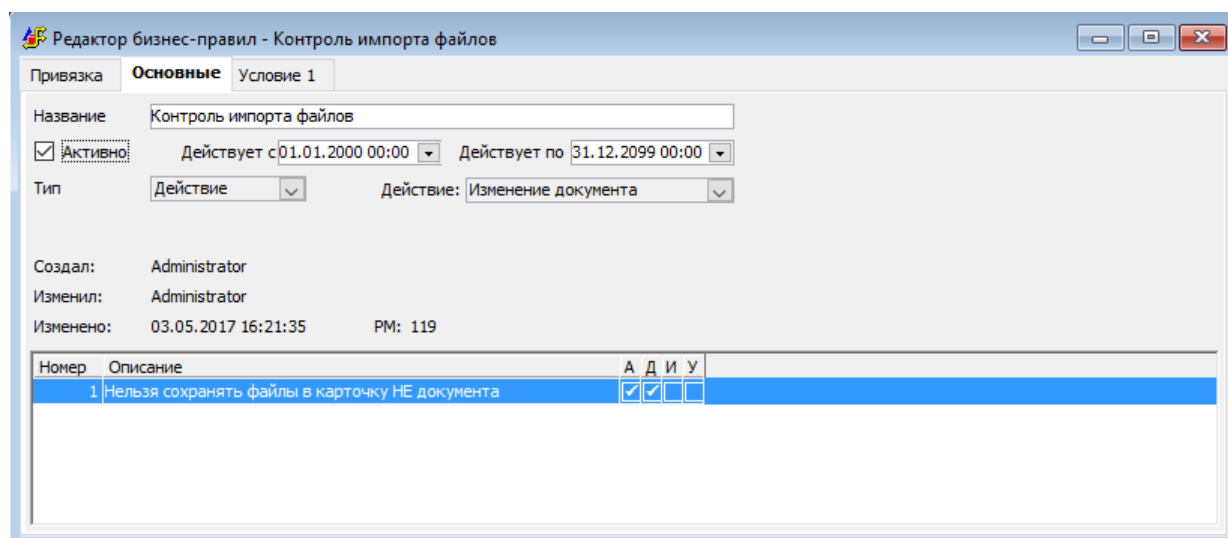


Рисунок 22 Бизнес-правило для контроля импорта документов

Затем нужно создать условие и включить в области «Когда проверять» флажок «Добавление», а в области «Что проверять» добавить любое заведомо невыполнимое условие (Рисунок 23). В нашем примере «****» – это такое название атрибута.

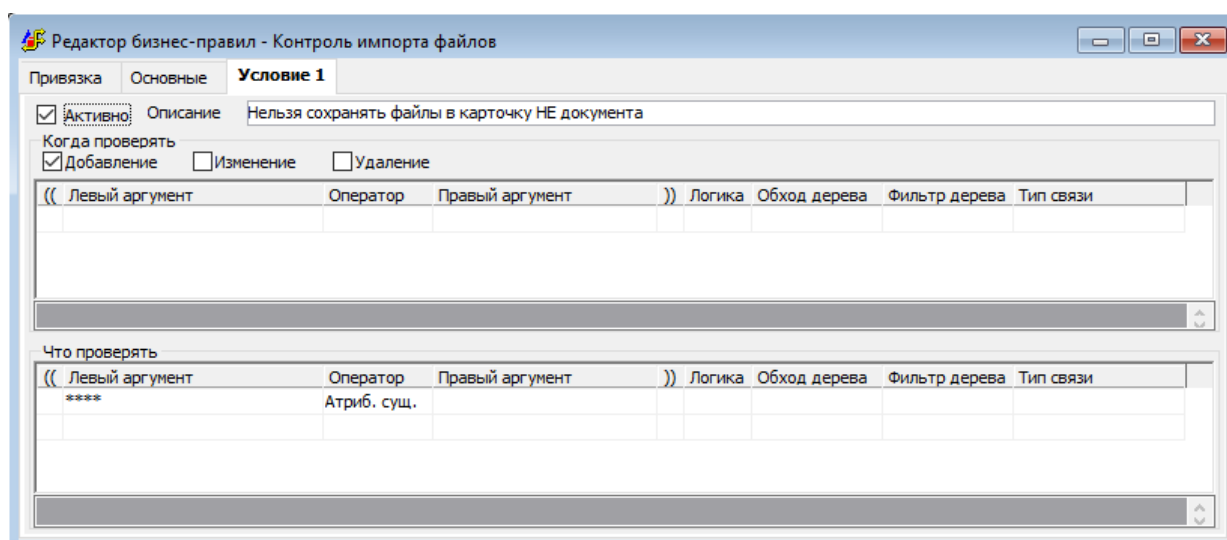


Рисунок 23 Условие бизнес-правила для контроля импорта документов

Название бизнес-правило и условие нужно назвать так, чтобы сообщение, выдаваемое пользователю, было понятным. В нашем примере сообщение будет содержать следующую информацию: «Правило 'Контроль импорта файлов' не выполняется: Нельзя сохранять файлы в карточку HE документа».

Если исключить из привязки какие-либо типы объектов, то пользователь сможет импортировать документы в исключенные типы объектов.

Бизнес-правило можно модифицировать не только за счет привязки к типам объектов, но и указывая другое условие. Например, можно указать определенные значения атрибута «Статус», запрещающие импорт документов в объект. Также, можно включить флажки проверки для изменения и/или удаления документа.

17 Последовательное добавление данных в значение атрибута с проверкой уникальности и сортировкой

Постановка задачи: требуется в текущее значение атрибута добавлять новые значения через разделитель, при этом должна быть возможность сортировки значений, находящихся между разделителями.

Для полноценной реализации подобного решения потребуется использование механизмов Workflow и действий над объектами.

Порядок реализации возможен следующий:

Имеется строковый атрибут, содержащий некоторое количество значений, разделенных символом-разделителем.

По нажатию кнопки в атрибутивной форме объекта выполняется действие, запускающее работу. При открытии формы первого этапа выполняется считывание и конвертация строки со списком значений с разделителем в массив значений.

Далее пользователь добавляет в массив элементы, после чего действием над переменными с использованием функции `f_DistinctArr()` из массива исключаются неуникальные элементы. Далее, циклическим действием над переменными значения элементов массива и разделителя передаются в строковую переменную, значение которой действием над объектом заносится в атрибут объекта.

При необходимости выполнения полноценной сортировки значений потребуется предварительно реализовать сортировку массива с переиндексацией.

Упрощенный вариант:

В некоторых частных случаях подобная задача может быть решена написанием единственного действия над объектами. Действие считывает существующее значение атрибута, пользователь в форме действия вводит значение очередного добавляемого элемента, действие формирует суммарную общую строку, и после завершения добавления пользователем элементов выполняется присвоение атрибуту сформированного значения.

Один из таких случаев – формирование значения атрибута «Формат». Документ может быть оформлен на листах различного формата и требуется инструмент для ввода этих форматов в значение атрибута (учитывая уникальность) и сортировку. Для хранения списка форматов конкретных документов используем атрибут «Формат (список)» с типом значения «Единичное».

По ГОСТ 2.106 различные форматы одного документа указываются в поле «Примечание» спецификации в порядке возрастания – следовательно, для использования значений атрибута «Формат (список)» при формировании спецификаций требуется сортировать добавляемые форматы по возрастанию, то есть сортировка не по алфавиту.

Формат должен указываться в списке однократно, то есть должна проверяться уникальность вводимого значения.

Проверка уникальности производится проверкой входимости добавляемой подстроки (`Str1`) в имеющуюся строку (`Str0`) значения атрибута. Могут использоваться функция `Pos` или оператор `like`. Проверка уникальности осложняется тем, что, например, «А4» и «А4х3» должны в результате проверки рассматриваться как различные значения, то есть при наличии в строке значения «А4х3» добавление значения «А4» должно выполняться.

Имеющееся значение «А4х3» в строке `Str1` окружено символами – разделителями, если оно только не расположено в начале или в конце строки. Сформируем промежуточную переменную `Str0_1`, которой присвоим значение `'rr'+ Str0+ 'rr'`, где `rr` – используемые символы разделители. Аналогично сформируем строку `Str1_1`:

`Str1_1= Set ('rr'+ Str1+ 'rr')`

Будем проверять входимость `Str1_1` в `Str0_1`, таким образом, проверка уникальности будет выполняться правильно.

Сортировка значений для частного случая, когда возможные значения элементов определены, а их количество ограничено, также может выполняться внутри действия над объектом. В порядке сортировки используется последовательная проверка входимости каждого возможного значения формата в строку Str0_1. Если вхождение есть, проверяемый формат добавляется через разделитель в переменную, при отсутствии вхождения значение остается прежним (например, для формата A4 может использоваться выражение `If (pos (Str0_1, 'A4') > 0, Str0_3+'rr'+ 'A4', Str0_3)`).

Последовательно проверив аналогичным образом все возможные форматы, заносим сформированное значение Str0_3 в атрибут объекта.

Последовательность проверки значений должна соответствовать требуемой последовательности сортировки.

Пример практической реализации подобного действия имеется в предопределенных настройках системы – «Машиностроение» и «Проектные организации».

18 Управление формами этапов работ с помощью действий над переменными

18.1 Формирование выпадающего списка значений в форме задачи

Имеется форма задачи. Одно из полей формы должно редактироваться из выпадающего списка. Список значений должен представлять собой динамически изменяемый список значений, например, какого-либо атрибута. Соответственно, требуется автоматически обновлять список значений поля формы.

Рассмотрим решение задачи. При настройке формы задачи для требуемой переменной задаем стиль редактирования «Выпадающий список». Каждое значение такого списка характеризуется двумя параметрами: описанием значения (то что видит на экране и выбирает пользователь) и реальным значением (значение, соответствующее выбору пользователя). Для строковых переменных оба параметра могут совпадать.

Создадим действие над переменными (Рисунок 24) с привязкой к событию «после открытия Окна задачи».

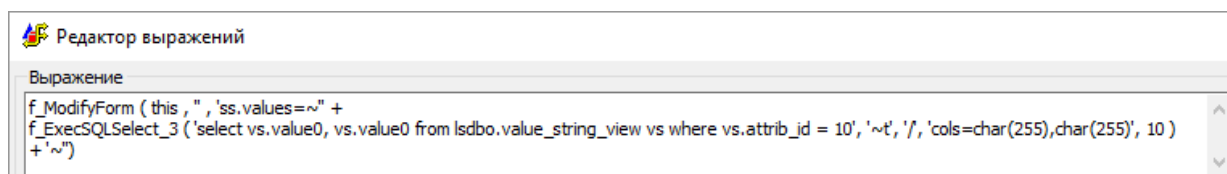


Рисунок 24

В этом действии «ss» – это строковая переменная, для которой мы создаем список значений. f_ExecSQLSelect_3 – функция, применяющая заданный SQL-запрос к текущей базе данных. В нашем случае, SQL-запрос два раза возвращает колонку value0 таблицы value_string для атрибута с ID=10. В нашей базе данных – это атрибут «Статус».

Результат выполнения действия – см. Рисунок 25.

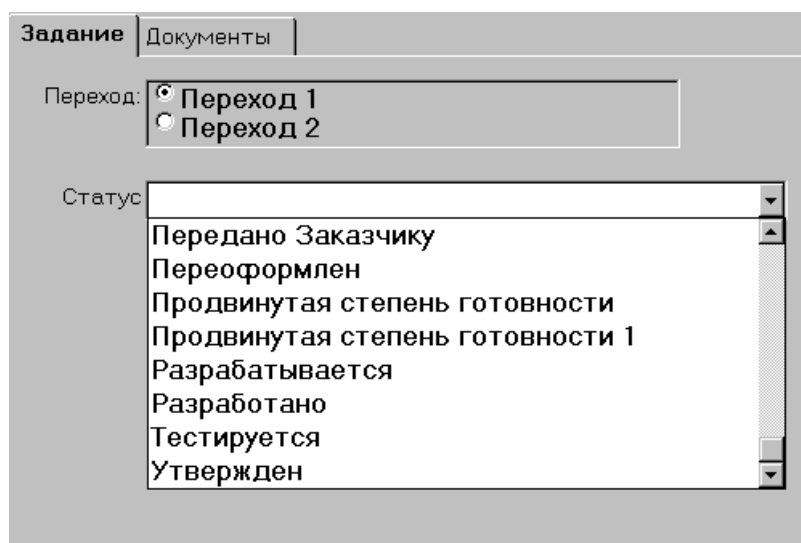


Рисунок 25

19 Расчет количества определенных символов (групп символов) в строке

Постановка задачи: требуется определить, сколько раз в указанной строке встречается определенный символ или группа символов (подстрока). Например, сколько в строке встречается переносов строк (~r~n) или сколько символов «о» в слове «обороноспособность».

Реализация: рассмотрим на примере расчета количества символов «о» в строке «обороноспособность».

1. Используя функцию `f_GlobalReplace` заменим в строке «обороноспособность» все символы «о» на пустые строки без учета регистра символов:

```
f_GlobalReplace ( 'обороноспособность', 'o', "", 1 )
```

2. Вычтем из длины строки «обороноспособность» длину строки, полученной в п. 1:

```
len ( 'обороноспособность' ) - len ( f_GlobalReplace ( 'обороноспособность', 'o', "", 1 ) )
```

Результат – 7.

Напомним, что вместо строк-констант обычно используются имена переменных или полей форм, в зависимости от контекста применения данного способа расчета.

20 Автоматический перебор и обработка объектов

Рассмотрим типовую задачу, когда необходимо сходным образом обработать некоторое количество объектов.

Обработка объектов может быть различной, и заключаться, например, в установке/удалении атрибутов, связей, импорте документов, копировании объектов, установке прав и так далее. Сама обработка с помощью соответствующих функций редактора действий обычно достаточно очевидна.

Для того, чтобы осуществить подобную обработку автоматически, в ходе выполнения действия, в действии над объектом следует организовать цикл. Средства для организации цикла рассмотрим подробнее. Инструментом для организации цикла может служить функция GoTo, с проверкой условий выхода из цикла. Для проверки условия обычно используется выражение If(). Проверка может выполняться как вначале, так и в конце цикла, но обычно удобнее вначале, так как в любом случае перед выполнением цикла проверка нужна.

Отбор (поиск) обрабатываемых объектов может осуществляться внутри цикла, либо предварительно.

Поиск внутри цикла обычно нежелателен, в первую очередь с точки зрения ресурсоемкости и скорости выполнения: необходимо в каждом цикле выполнять поиск объекта в БД, а также запись результата выполнения обработки в БД (Update), чтобы уже обработанный объект не был найден повторно. Инструментом выполнения поиска внутри цикла может быть функция f_ExecSQLSelect_3 и соответствующий SQL-запрос, возвращающий единственное значение id объекта (например - минимальное), либо, например, использование условий отбора для объектов и функции VarCheckValues для их проверки и присвоения значения переменной обрабатываемого объекта.

Предварительный (то есть до выполнения цикла) отбор может осуществляться различными способами (например, функцией f_ExecSQLSelect_3 с соответствующим SQL-запросом, окном выбора, выбором файлов). Результатом такого отбора обычно является строка id объектов, которые предстоит обработать, указанных через разделитель. Ограничением тут может быть максимально допустимая длина строки (64 000 символов).

Таким образом, задача автоматического перебора объектов в основном сводится к циклической обработке строки id объектов внутри действия. В качестве инструментов такой обработки используем строковые функции (в данном примере использованы функции: left, right, pos, len, их описание имеется в документации и контекстной справке по системе).

Рассмотрим пример: есть переменная str_id, содержащая значения id объектов через разделитель, который может быть любым, для определенности используем символ «;» (точка с запятой). Требуется автоматически обработать объекты, идентификаторы которых содержатся в строке str_id.

Шаг, с которого начинается цикл, имеет метку «Start», шаг, на который переходим по окончании цикла, помечен «End_start».

Вначале проверим содержимое строки str_id: если она пустая – завершим цикл, если нет – продолжим, это также является условием выхода из цикла:

```
GoTo ( If( str_id = " " , 'End_start' , " " ) )
```

Если строка str_id содержит разделитель, «вырежем» из нее значение ID слева до разделителя, в противном случае, значение строки соответствует единственному ID. Присвоим значение переменной обрабатываемого объекта obj:

```
obj = SetByID ( Number ( If( Pos( str_id , ';' ) > 0, left( str_id, pos(str_id , ';' )-1) , str_id)))
```

Переменной str_id присвоим значение правее разделителя (это в случае, если текущее значение str_id содержит разделитель, в противном случае, присвоим пустое значение):

```
str_id = Set ( If( Pos( str_id , ';' ) >0, right( str_id, len (str_id)- pos(str_id , ';') ) , " ) )
```

Выполним проверку присвоения значения переменной obj:

```
GoTo (if(isnull(obj), 'start'," ) )
```

Далее осуществим обработку объекта obj. Как говорилось выше – обработка может быть произвольной и в данном примере не рассматривается.

Далее осуществляем безусловный переход на начало цикла:

```
GoTo ('start')
```

Отметим, что данная (и сходная с ней) задача может быть решена также с использованием скриптов. В этом случае можно обработать проект целиком, независимо от количества уровней в дереве. За консультацией и примерами подобных скриптов вы можете обратиться в службу технической поддержки по адресу techhelp@lotsia.com.

21 Рекомендации по SQL-отбору объектов по атрибутам и атрибутов объекта

Значения атрибутов можно извлечь обращением к представлениям (view) `lsdbo.attrb_value_s_v`, `lsdbo.attrb_value_n_v` и `lsdbo.attrb_value_t_v` для строковых, числовых и датавременных атрибутов соответственно.

Для поиска объекта по значению атрибута рекомендуется использовать запрос следующей структуры (представление `...s_v`, `...n_v` или `...t_v` используется в соответствии с типом данных атрибута):

```
select object_id
from lsdbo.attrb_value_s_v
where lsdbo.attrb_value_s_v.attrb_id = <код_атрибута>
and lsdbo.attrb_value_s_v.value_aid = <код_атрибута>
and lsdbo.attrb_value_s_v.value = ...
```

В большинстве случаев выборка по значению строкового атрибута идет по начальным символам, поэтому рекомендуется в последнем условии использовать колонку `value0`:

```
and lsdbo.attrb_value_s_v.value0 = ...
```

В колонке `value0` для строковых атрибутов хранятся первые 255 символов значения. В колонке `value` – остальные символы (при их наличии).

Для получения значения атрибута известного объекта рекомендуется использовать запрос следующей структуры (представление `...s_v`, `...n_v` или `...t_v` используется в соответствии с типом данных атрибута):

```
select lsdbo.attrb_value_s_v.value
from lsdbo.attrb_value_s_v
where lsdbo.attrb_value_s_v.object_id = <код_объекта>
and lsdbo.attrb_value_s_v.treelink_id = <0 для атрибутов объекта или код связи из представления tree_link_view>
and lsdbo.attrb_value_s_v.attrb_id = <код_атрибута>
```

22 Удаление группы атрибутов

Иногда необходимо удалить у объекта или объектов несколько атрибутов, при этом удаляемые атрибуты объединены в отдельную группу.

Например, есть группа атрибутов под названием «Подписи», куда включены все атрибуты подписей такие как: название подписи, ФИО подписывающего, дата подписи, ID пользователя и т.д. Для согласования документов (группы документов) используется шаблон работы, в атрибуты записывается информация, но в процессе согласования может наступить момент, когда необходимо удалить все ранее проставленные подписи, то есть когда документ идет на повторное согласование.

С технической точки зрения, данная задача имеет много общего с задачей, рассмотренной выше, в разделе «Автоматический перебор и обработка объектов». Отметим, что данную задачу можно решить различными способами:

- составить действие, где в переменных явным образом определены все возможные атрибуты подписей. В действии последовательно удалять все эти атрибуты функцией `AttribDel`, предварительно проверяя их наличие. Либо, использовать во всех случаях для удаления функцию `AttribSet` с установкой атрибуту `Null` – значения;
- составить действие, где подписи удаляются в циклическом режиме.

Использование первого способа достаточно очевидно, при использовании второго способа верны соображения, рассмотренные в разделе «Автоматический перебор и обработка объектов», с поправкой на то, что обрабатывать будем не различные объекты, а различные атрибуты одного объекта. Получить строку с `id` атрибутов, входящих в группу атрибутов и имеющихся у данного объекта, можно следующим запросом:

```
SELECT atr.id  
FROM lsdbo.attrib_view atr, lsdbo.attrib_value_view anp  
WHERE anp.object_id = <ID объекта> and atr.group_id = <ID группы атрибутов> and  
anp.attrib_id = atr.id
```

Запрос может быть использован в функции `f_ExecSQLSelect_3` для получения строки, содержащей `id` удаляемых атрибутов через разделитель, далее обработка строки не отличается от рассмотренной в разделе «Автоматический перебор и обработка объектов».

23 Нумерация групп в формах

Зачастую возникают ситуации, когда в формах, допускающих группировку, требуется проставить порядковый номер в заголовке группы. Решение этой задачи состоит в том, чтобы с помощью одного вычисляемого поля «пометить» начало новой группы единицей, а с помощью другого – подсчитать и отобразить нарастающий итог этих пометок. Оба вычисляемых поля размещаются в области заголовка группы.

Приведем содержание вычисляемых полей. Выражение вычисляемого поля, «помечающего» начало группы выглядит так:

`If (GetRow() = First(GetRow() for group 1), 1, 0)`

Дадим этому полю имя, например, GR. На это имя будет ссылаться второе вычисляемое поле.

Выражение вычисляемого поля, подсчитывающего нарастающий итог пометок и отображающего порядковый номер группы, выглядит так:

`CumulativeSum (GR for all)`

Чтобы не нарушать оформление, поле GR можно сделать скрытым.

24 Вставка изображений в формы

Система позволяет вставлять в формы изображения. Изображения могут представлять собой как самостоятельные элементы оформления, так и подложки для кнопок или других полей форм. Можно использовать изображения общепринятых растровых форматов (bmp, jpg, gif и др.). Изображения могут вставляться в любые формы и в любые области форм, допускающие использование вычисляемых полей.

Для вставки изображения следует создать в форме вычисляемое поле, в выражении которого использовать функцию Bitmap (<bitmap filename>).

Выражение <bitmap filename> должно возвращать строку, содержащую полное имя файла изображения, включающее путь. Может указываться как абсолютный путь (в том числе сетевой), так и путь относительно рабочего каталога. Например, если в рабочем каталоге имеется папка для изображений \Pic, в которой расположен файл изображения Lotsia.gif, то при указании относительного пути выражение вычисляемого поля может быть, например, таким:

```
bitmap ('Pic\Lotsia.gif')
```

Результат вставки в форму изображения см. на рисунке (Рисунок 26).

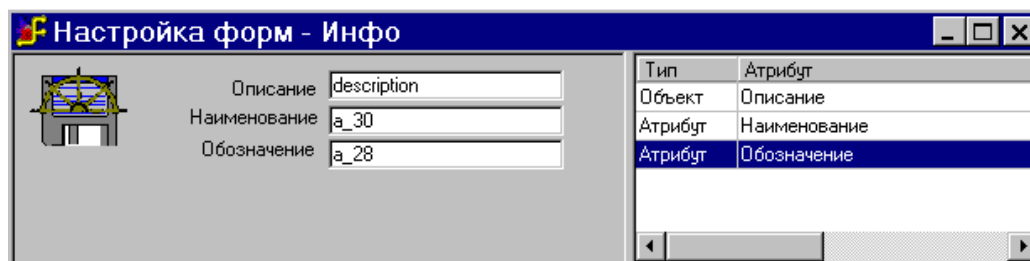


Рисунок 26 Пример настройки формы со статическим изображением

Для вставки в форму изображения, которое изменяется в зависимости от заданных параметров, выражение <bitmap filename> должно по разным условиям возвращать строку с разными полными именами файлов. Например, если в папке \Pic расположены файлы, содержащие фотографии сотрудников предприятия, а имена указанных файлов в формате jpg соответствуют значению атрибута «Табельный номер» для объекта сотрудника, выражение вычисляемого поля может быть, например, таким:

```
bitmap ('Pic\' + a100000006544037 + '.jpg')
```

где a100000006544037 – имя колонки атрибута «Табельный номер».

В форме будет отображаться фотография, соответствующая значению атрибута текущего объекта «Табельный номер».

Отметим, что автоматическое масштабирование размера вычисляемого поля под размер изображения не производится, поэтому при использовании различных изображений через одно вычисляемое поле рекомендуется изображения в файлах привести к одному размеру.

Для обновления или рассылки графических файлов, при их хранении на рабочих станциях, может использоваться функция [автоматического обновления](#). Если для хранения указанных файлов используется папка \Pic в рабочем каталоге программы, необходимо в общедоступной папке с обновлением создать аналогичную папку, куда и помещать новые и обновленные файлы. При очередном автоматическом обновлении клиентской части программы будет выполнено копирование содержимого каталога \Pic в каталог программы на рабочих станциях.

Возможен показ изображения и с использованием документа архива, например, выражение:

```
bitmap (f_GetVerPathName (
```



```

number (f_ExecSQLSelect_3 (
'select max (ver.id)
from lsdbo.file_ver_v ver, lsdbo.file_v doc
where
ver.file_id = doc.id
and doc.file_type_id = 110000002
and doc.object_reference_id = ' + id
, ", ", 'cols=number', 60))))

```

Где: 110000002 – ID типа документа, используемого в качестве изображения (используйте ваш соответствующий ID из настройки типов документов);

Id – идентификатор текущего объекта, соответствующую колонку необходимо добавить в форму.

Отобразит в форме объекта изображение, соответствующее версии с максимальным ID документа архива указанного типа, для данного объекта.

25 Некоторые практические приемы настройки отчетов

Рассматривать приемы настройки отчетов будем на примере следующей настройки, посвященной инвентарному учету компьютеров, расположенных в помещениях, на разных этажах здания – см. Рисунок 27. Выделенные подчеркиванием слова соответствуют типам объектов настройки.

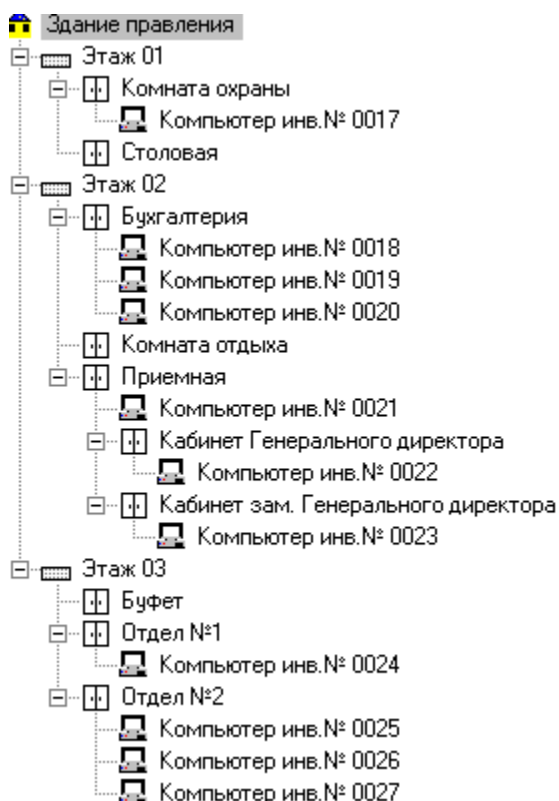


Рисунок 27 Учебный проект «Учет компьютеров»

Пусть имеются следующие присвоенные атрибуты объектов у потомков: «Номер инвентарный» – (строковое значение, формат – порядковый номер, дополненный впереди нулями до стандартной длины, например, 4 символа: 0001, 0002 и т.д.), «Дата постановки на инвентарный учет» – (дата, формат dd.mm.yyyy, задан для атрибута).

Все объекты имеют названия, соответствующие описаниям объектов (см. Рисунок 27).

25.1 Пример создания отчета

Общая последовательность создания отчета описана в документе [«Руководство пользователя Lotsia PDM»](#).

Создадим настраиваемый отчет «Журнал учета компьютеров» со следующими параметрами:

Тип отбора – «Отбор по условиям отчета»

Глубина поиска объектов – «Все уровни»

Строка для выделенного объекта обязательна – «Нет»

Отчет имеет единственный раздел, без названия, направление поиска потомков – вниз по связи «Дерево проектов», то есть название, связь и направление поиска для раздела, заданные «по умолчанию» при его создании нет необходимости изменять. В данном разделе отбираем потомков типа «Компьютер».

Колонки отчета: «Дата постановки на инвентарный учет (Потомок)», «Номер инвентарный (Потомок)» и «Описание объекта (Потомок)», см. Рисунок 28.

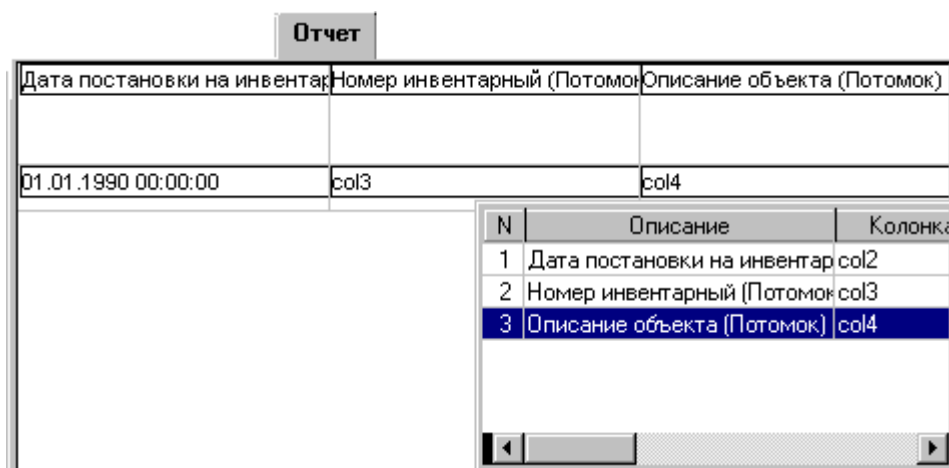


Рисунок 28 Добавленные колонки отчета

Названия колонок отчета, их графическое оформление и положение настраивается аналогично [настройке форм](#).

Зададим правильный формат отображения даты – см. Рисунок 29.

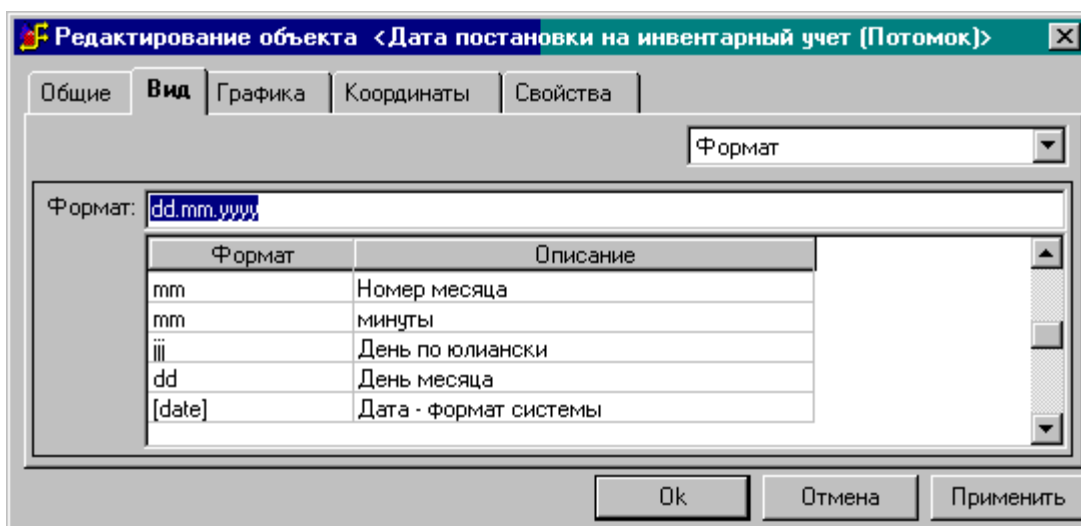


Рисунок 29 Задание формата отображения даты

Поместив курсор в верхнюю часть области отчета, из контекстного меню выберем «Свойства областей». Изначально в отчете имеются области:

- Заголовок отчета;
- Область данных;
- Область итогов;
- Нижний колонтитул;

Чтобы нагляднее отобразить области на экране, зададим различные цвета для областей отчета, для области нижнего колонтитула установим высоту около 100 единиц.

Обратите внимание, что область итогов пока не отображена на экране. Для того, чтобы отобразить область итогов в отчете, необходимо создать сумму по какой-либо их колонок отчета. Суммировать можно числовые колонки. Добавим в область данных отчета вычисляемое поле (по умолчанию оно имеет числовой формат) и создадим для него сумму.

После создания суммы в отчете отобразится и Область итогов. В области итогов создадим вычисляемое поле для подсчета количества потомков, его выражение может быть, например, следующим:

'Всего компьютеров: '+ Count(col2 for All)+' шт.'

где:

Count(col2 for All) – функция, определяющая количество заполненных строк в колонке (в данном случае – col2) для всего отчета (то есть for All).

В кавычках указаны постоянные строковые значения.

Область итогов отображается в отчете до тех пор, пока в ней присутствует хотя бы один элемент отчета (это может быть колонка, текстовое или вычисляемое поле).

Для перемещения полей, расположенных в области заголовка отчета, используйте клавиши со стрелками, для изменения размеров полей – клавиши со стрелками при нажатой клавише «Shift». Для остальных областей пользуйтесь мышью или клавиатурой.

Для отображения номера текущей страницы отчета и их общего количества, в области нижнего колонтитула создадим вычисляемое поле и введите в него следующее выражение:

'Стр. '+Page()+' из '+ PageCount()

В области заголовка отчета создадим вычисляемое поле, введем, например, выражение такого вида:

'Журнал инвентарного учета компьютеров', поместим поле с заголовком на передний план, размеры и положения заголовка определим с помощью клавиатуры, зададим графическое оформление – см. Рисунок 30.

Отчет

Журнал инвентарного учета компьютеров			
Дата постановки на инвентарный учет (Потомок)	Номер инвентарный (Потомок)	Описание объекта (Потомок)	
01.01.1990	col3	col4	0
Всего компьютеров: 1 шт.			0
N	Описание	Колонка	Уровень
1	Дата постановки на инвентар	col2	Потомок
2	Номер инвентарный (Потомок)	col3	Потомок
3	Описание объекта (Потомок)	col4	Потомок
4	Исполнитель (Потомок)	col5	Потомок

Стр. 1 из 1

Рисунок 30 Отчет с выделенными областями

Результат выполнения отчета по объекту «Здание правления», (см. Рисунок 27) может быть таким – см. Рисунок 31.

Отчет 'Журнал учета компьютеров'			
Журнал инвентарного учета компьютеров			
Дата постановки на инвентарный учет (Потомок)	Номер инвентарный (Потомок)	Описание объекта (Потомок)	
23.05.2004	0024	Компьютер инв. № 0024	0
23.05.2004	0025	Компьютер инв. № 0025	0
23.05.2004	0026	Компьютер инв. № 0026	0
23.05.2004	0027	Компьютер инв. № 0027	0
Всего компьютеров: 11 шт.			0
Стр. 3 из 3			
Обновить			

Рисунок 31 Выполненный отчет

25.2 Сортировка строк отчета

Отметим, что установить порядок сортировки можно как в выполненном отчете, так и в отчете, при его настройке.

Сортировка выполняется в алфавитном порядке, независимо от формата колонки сортировки. Для [описанного отчета](#) мы использовали данные, изначально имеющие формат, обеспечивающий правильную сортировку.

В реальном отчете, для обеспечения правильной сортировки, например, по инвентарному номеру, обычно необходимо выполнить дополнительную настройку.

Например, для определенности, пусть инвентарный номер имеет следующий формат:

X/V, где:

X – порядковый номер, число, количество разрядов различное

/ – фиксированный разделитель

V – буквенный индекс переменной длины

Сортировку необходимо выполнить в порядке возрастания номеров, то есть

Например, так: 2/ке, 17/в, 18/тмп, 19/W14, 21/цмк

а не: 17/в, 18/тмп, 19/W14, 2/ке, 21/цмк.

В подобных случаях используется следующий типовой прием – в области данных создается вычисляемое поле, где обрабатывается содержимое колонки (колонок) по которым необходимо выполнить сортировку, с целью приведения данных к строкам вида, соответственно: 0002, 0017, 0018, 0019, 0021, либо к числовому формату. По указанному вычисляемому полю и производится сортировка строк отчета. Для удобства использования удобно присвоить данному вычисляемому полю название (обеспечивается путем создания текстового поля с требуемыми [свойствами](#)) например, «Порядковый номер». В готовом отчете данное поле можно не показывать (скрыв его с помощью задания соответствующих свойств, или ширину колонки равной 0). В выражения подобных вычисляемых полей обычно используют строковые функции, например, в рассмотренном случае выражение может быть таким:

string (number (Mid (col3 , 1 , Pos(col3 , '/')-1)), '0000') где:

col3 – колонка «Номер инвентарный»;

Pos (col3 , '/') – возвращает номер позиции разделителя «/»;

Mid – строковая функция, получает из строки подстроку заданной длины, начиная с указанной позиции;

Number – функция, преобразующая строку в число;

String – функция, преобразующая число в строку заданного формата (0000)

Описание функций – см. [Редактор выражений](#).

Для преобразования в числовой формат, можно использовать выражения:

number (Mid (col3 , 1 , Pos(col3 , '/')-1)), либо

number (Left (col3 , Pos(col3 , '/')-1))

25.3 Нумерация строк отчета

Для нумерации строк отчета можно использовать вычисляемое поле, содержащее функцию GetRow(), которая возвращает порядковый номер строки, или использовать вычисляемое поле, аналогичное примененному нами для [сортировки](#).

Добавим в отчет нумерацию строк.

25.4 Группировка строк отчета

Использование группировки подразумевает, что предварительно строки отчета были отсортированы по той же колонке отчета, по которой выполняется группировка.

Для [рассмотренного отчета](#) сгруппируем строки по датам постановки на инвентарный учет.

Предварительно зададим сортировку строк, указав колонки [сортировки](#) в следующем порядке: «Дата постановки на инвентарный учет», «Порядковый номер». Сортировку зададим по возрастанию (установлена по умолчанию).

При задании группировки, в окне «Группировка» (см. Рисунок 32) укажем высоту 150 единиц для добавляемых областей отчета, а также для визуализации этих областей пометим их различными цветами.

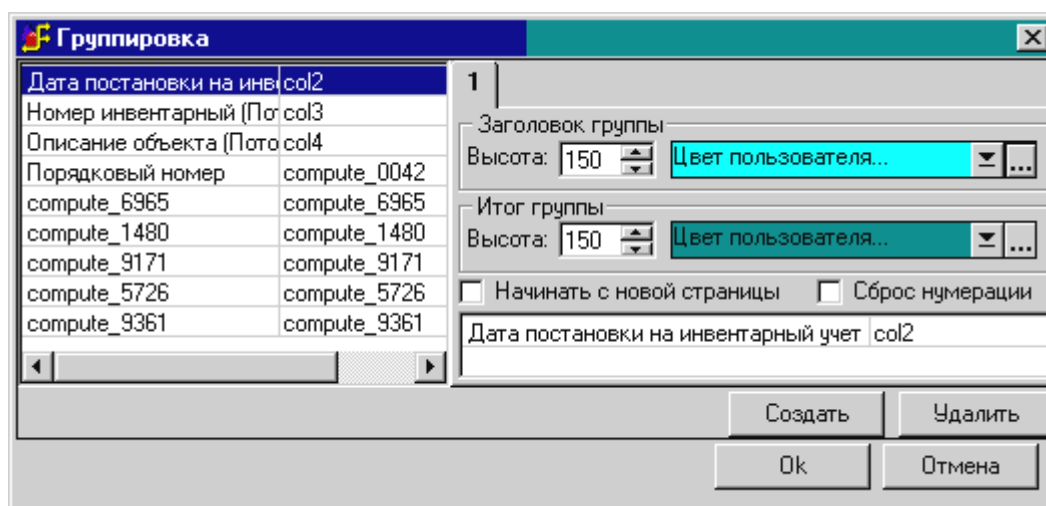


Рисунок 32 Задание параметров группы

В области заголовка группы (см. Рисунок 33) создадим вычисляемое поле, введя в него выражение:

col2 где:

col2 – колонка «Дата постановки на инвентарный учет»

В области итогов группы (см. Рисунок 33) также создадим вычисляемое поле, введя в него выражение:

'За ' + date (col2) + ' присвоено' + '~r~n' + Count(col3 for group 1) + ' инв. ном.' где:

col2 – колонка «Дата постановки на инвентарный учет»

'~r~n' – специальная строка, обеспечивает жесткий перенос текста

col3 – колонка «Номер инвентарный»

Count (col3 for group 1) – функция, определяющая количество строк с непустыми колонками col3 в группе.

Отчет

Журнал инвентарного учета компьютеров

Дата постановки на инвентарный учет (Потомок)	Номер инвентарный (Потомок)	Описание объекта (Потомок)	Поряд												
	01.01.1990														
01.01.1990	col3														
За 01.01.1990 присвоено 1 инвентарных номеров															
Всего компьютеров: 1 шт.															
		<table><tr><th>N</th><th>Описание</th><th>Кол</th></tr><tr><td>1</td><td>Дата постановки на инвентар</td><td>col2</td></tr><tr><td>2</td><td>Номер инвентарный (Потомок</td><td>col3</td></tr><tr><td>3</td><td>Описание объекта (Потомок)</td><td>col4</td></tr></table>	N	Описание	Кол	1	Дата постановки на инвентар	col2	2	Номер инвентарный (Потомок	col3	3	Описание объекта (Потомок)	col4	
N	Описание	Кол													
1	Дата постановки на инвентар	col2													
2	Номер инвентарный (Потомок	col3													
3	Описание объекта (Потомок)	col4													

Рисунок 33 Настроенный отчет с группировкой. Пример

После выполнения данный отчет выглядит следующим образом – см. Рисунок 34:

Дата постановки на инвентарный учет (Потомок)	Номер инвентарный (Потомок)	Описание объекта (Потомок)
	04.05.2004	
04.05.2004	2/ке	Компьютер инв. № :
04.05.2004	17/в	Компьютер инв. № :
За 04.05.2004 присвоено 2 инв. ном.		
	10.05.2004	
10.05.2004	18/тмп	Компьютер инв. № :
10.05.2004	19/М14	Компьютер инв. № :
За 10.05.2004 присвоено 2 инв. ном.		

Рисунок 34 Фрагмент выполненного отчета с группировкой. Пример

Отметим, что в качестве колонки для группировки можно использовать и различные системные колонки. Например, при использовании системной колонки «Раздел отчета (номер)», группировка строк отчета может выполняться в соответствии с разделами отчета, заданными на вкладке «Раздел».

25.5 Обработка повторяющихся строк отчетов

В рассматриваемом выше [примере \(модели данных\)](#) отсутствуют повторяющиеся строки отчета (любой из компьютеров может быть расположен только в одном помещении). Подобная единичная входимость объектов учета в реальной настройке часто не соблюдается.

Для иллюстрации множественной входимости, рассмотрим в качестве объектов учета, например, объекты документов-актов проверки противопожарного состояния помещений. Будем считать, что такие акты могут оформляться по результатам разовых проверок нескольких произвольно выбранных помещений здания.

Тип объекта учета – «Акт проверки», внешний вид дерева проекта – см. Рисунок 35.

Атрибуты объектов учета, используемые в отчете – «Регистрационный номер» и «Дата регистрации». В остальном модель данных аналогична [рассмотренной выше](#).

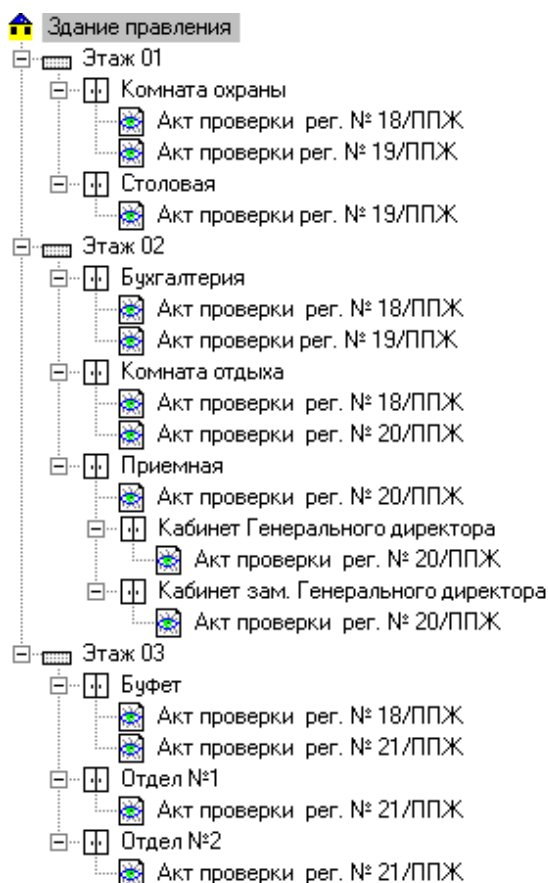


Рисунок 35 Проект «Учет актов проверки противопожарного состояния помещений»

Создадим отчет «Книга регистрации актов проверок», аналогичный рассмотренному выше отчету «[Журнал регистрации компьютеров](#)».

Для создания нового отчета можно выполнить копирование отчета «Журнал регистрации компьютеров» и далее отредактировать его элементы:

- на вкладке «Разделы» укажите тип объекта – «Акт проверки».

- на вкладке «Отчет» измените атрибуты в свойствах колонок (выделите строку с колонкой в области колонок, в контекстном меню выберите «Свойства», в окне свойств укажите требуемый атрибут).
- в области отчета отредактируйте содержимое текстовых полей с названиями колонок, а также соответственно измените строки в вычисляемых полях.
- сортировку и группировку отчета следует сохранить как в исходном отчете.

Отчет может выглядеть так – см. Рисунок 36:

Отчет														
Книга регистрации актов проверок														
Дата регистрации (Потомок)	Регистрационный номер (Потомок)	Описание объекта (Потомок)												
	01.01.1990													
01.01.1990	col3	col4												
За 01.01.1990 присвоено 1 рег. ном.														
Всего актов проверок: 1 шт.														
<table border="1"> <thead> <tr> <th>N</th><th>Описание</th><th>Колонка</th></tr> </thead> <tbody> <tr> <td>1</td><td>Дата регистрации (Потомок)</td><td>col2</td></tr> <tr> <td>2</td><td>Регистрационный номер (Потомок)</td><td>col3</td></tr> <tr> <td>3</td><td>Описание объекта (Потомок)</td><td>col4</td></tr> </tbody> </table>			N	Описание	Колонка	1	Дата регистрации (Потомок)	col2	2	Регистрационный номер (Потомок)	col3	3	Описание объекта (Потомок)	col4
N	Описание	Колонка												
1	Дата регистрации (Потомок)	col2												
2	Регистрационный номер (Потомок)	col3												
3	Описание объекта (Потомок)	col4												

Рисунок 36 Настроенный отчет «Книга регистрации актов проверок»

Выполнив данный отчет по объекту «Здание правления» убедимся, что строки потомков в отчете повторяются, и количество актов определяется неверно, см. Рисунок 37.

Книга регистрации актов проверок		
Дата регистрации (Потомок)	Регистрационный номер (Потомок)	Описание объекта (Потомок)
	01.01.2004	
01.01.2004	18/ЛПЖ	Акт проверки рег. № 18/ЛПЖ
01.01.2004	18/ЛПЖ	Акт проверки рег. № 18/ЛПЖ
01.01.2004	18/ЛПЖ	Акт проверки рег. № 18/ЛПЖ
01.01.2004	18/ЛПЖ	Акт проверки рег. № 18/ЛПЖ
01.01.2004	19/ЛПЖ	Акт проверки рег. № 19/ЛПЖ
01.01.2004	19/ЛПЖ	Акт проверки рег. № 19/ЛПЖ
01.01.2004	19/ЛПЖ	Акт проверки рег. № 19/ЛПЖ
За 01.01.2004 присвоено 7 рег. ном.		

Рисунок 37 Фрагмент выполненного отчета с повторяющимися строками

Для того, чтобы не показывать повторяющиеся значения в столбцах отчета, применим инструмент [«Подавление повторяющихся значений»](#).

Подавление повторяющихся значений не изменяет количества потомков в отчете, а лишь воздействует на отображение данных в экранной форме. Поэтому применение данного средства не влияет на количество элементов, определяемое в итогах отчета или группы.

Для того, чтобы потомок использовался для определения итогов отчета или группы единожды, следует использовать опцию Distinct в функциях, определяющих итоги отчета (группы). Например, выражение вычисляемого поля для определения количества потомков в группе может иметь следующий вид:

'За ' + date (col2) + ' присвоено' + '~r~n' + Count (col3 for group 1 distinct col4) + ' рег. ном.'

где:

col2 – колонка «Дата постановки на инвентарный учет»

'~r~n' – специальная строка, обеспечивает жесткий перенос текста

col3 – колонка «Номер инвентарный»

col4 – колонка, по которой контролируется уникальность объекта (в данном случае все объекты имеют различные описания, поэтому можно использовать колонку «Описание объекта», что и сделано).

Для того, чтобы быть абсолютно уверенным в правильности подведения итогов по уникальным объектам, рекомендуется добавлять в отчет специальную системную колонку «Объект (код)», в которой будет отображаться ID объекта – потомка. Ввиду того, что уникальность ID контролируется системой автоматически, использование этой колонки для контроля уникальности предпочтительно в большинстве случаев.

Отметим, что если данная колонка присутствует в отчете (хотя бы в области колонок, в области отчета ее можно не показывать), то при двойном щелчке левой клавишей мыши по строке потомка в выполненном отчете, в отдельном окне будет открыт объект с данным ID.

При использовании опции «distinct» отчет выполняется верно – см. Рисунок 38.

Дата регистрации (Потомок)	Регистрационный номер (Потомок)	Описание объекта (Потомок)
	01.01.2004	
01.01.2004	18/ППЖ	Акт проверки рег. № 18/ППЖ
	19/ППЖ	Акт проверки рег. № 19/ППЖ
За 01.01.2004 присвоено 2 рег. ном.		

Рисунок 38 Фрагмент выполненного отчета, в котором применено подавление повторяющихся значений и использовано определение итогов по уникальным значениям

Удалить из отчета повторяющиеся строки можно также с помощью [фильтрации](#). Выражение фильтра может быть следующим:

`isobject_id <> isobject_id [-1] or GetRow ()=1`

первая часть выражения (до `or`) сравнивает значение `isobject_id` (`isobject_id` – имя колонки «Объект (код)») в данной строке отчета и в предыдущей. Строка войдет в отчет в случае, если значения не повторяются.

Для первой строки любого отчета данное условие не выполняется, так как для первой строки значения `isobject_id [-1]` не существует

Чтобы первая строка автоматически не исчезала из отчета, в выражение фильтра добавлена вторая часть – функция `GetRow ()` возвращает номер текущей строки.

Предварительно строки должны быть отсортированы по той же самой колонке, в данном случае по колонке «Объект (код)».

Использование фильтрации, в отличие от подавления повторяющихся значений, влияет на итоги по отчету или группе.

Отметим, что имеется возможность фильтровать как выполненный, так и настраиваемый отчет. В некоторых случаях, например, при использовании в выражении фильтра значений, содержащихся в различных строках отчета (как в приведенном выражении), результат, полученный при фильтрации выполненного отчета, может отличаться от того, который будет получен при использовании аналогичного фильтра при настройке отчета. В подобных случаях следует использовать фильтрацию при настройке отчета.

25.6 Быстрое удаление всех типов объектов из вкладки «Разделы» отчета

Если во вкладке «Разделы» Окна настройки отчета задано много типов объектов и требуется их все удалить, то наиболее простой способ – удалить раздел отчета. Затем раздел можно вновь создать.

25.7 Использование аргументов отчета

Отчет, в котором глубина поиска объектов не «Вся база данных» и не «Пользовательский», имеет обязательный аргумент – «Объект» (это текущий объект, относительно которого в соответствии с заданным направлением поиска ищутся потомки).

Если в выполненном отчете перейти на вкладку «Аргументы», значение аргумента можно изменить. После обновления отчета он будет сформирован с использованием текущих значений аргумента.

В отчете можно использовать и дополнительные аргументы.

Для рассмотренных выше отчетов в качестве дополнительных аргументов можно использовать даты начала и окончания периода. Если значения таких аргументов использовать при фильтрации строк отчета, можно формировать отчет за заданный период.

Добавим в отчет 2 аргумента – начало и конец периода – см. Рисунок 39.

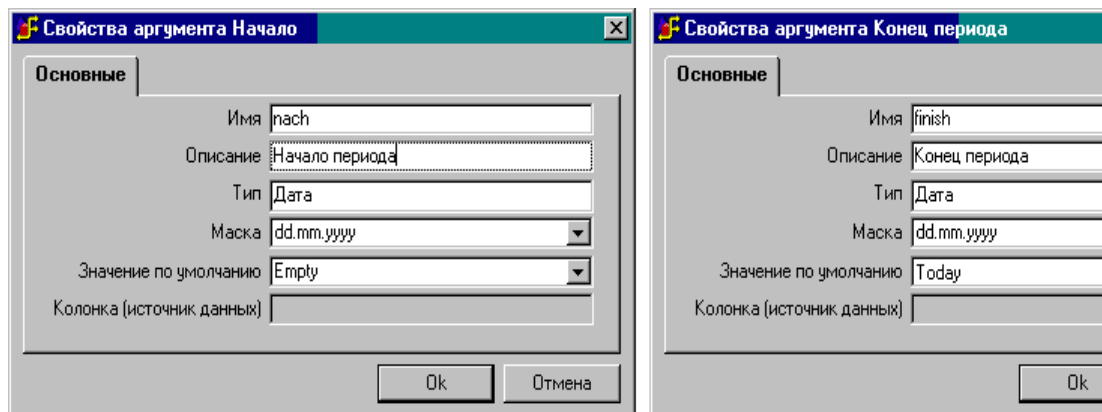


Рисунок 39 Свойства аргументов отчета

Добавленные аргументы можно использовать в вычисляемых полях отчета, в том числе, в форме аргументов отчета, а также в выражении фильтра и/или в условиях дополнительного отбора. Настройка и применение условий дополнительного отбора описаны в разделах «Дополнительные условия отбора в SQL», «Динамическое формирование дополнительных условий отбора в SQL на основе аргументов. Использование аргументов в конструкции IN» документа.

Дополним имеющееся выражение фильтра следующим:

and col2 >= nach and col2 <= finish , где col2 – «Дата регистрации».

Выражение вычисляемого поля заголовка также отредактируем:

'Книга регистрации актов проверок'~r~n'+' за период с '+' nach '+' по '+' finish

При выполнении отчета пользователь сможет указывать различные даты начала/окончания периода, при этом попадающие в отчет строки соответствуют заданному периоду.

25.8 Ввод аргументов непосредственно в окне отчета

При настройке отчета может быть указано местоположение формы с аргументами (Рисунок 40).

Рисунок 40 Указание места расположения формы с аргументами отчета

По умолчанию установлено значение «Отдельно», что соответствует расположению формы аргументов на отдельной вкладке. При выборе другого значения, например, «Сверху», форма с аргументами будет расположена в окне отчета, что обеспечивает ввод значений аргументов непосредственно в окне отчета. При печати отчета форма аргументов не печатается.

25.9 Особенности запуска и выполнения отчетов с аргументами

Остановимся чуть подробнее на практически важных моментах запуска отчетов. По умолчанию, при запуске отчета сразу происходит и его выполнение, которое может занимать ощутимое время. Если подразумевается, что выполнение отчета требует предварительного ввода аргументов, то выполнять отчет сразу при запуске, до ввода аргументов нецелесообразно. Для предотвращения немедленного выполнения отчета при его запуске, в окне настройки отчета на вкладке «Свойства» отключите флажок «Выполнять при запуске» (Рисунок 41).

После ввода значений аргументов отчет должен быть выполнен, для чего используются клавиатурные клавиши «Enter» или «F5» или кнопка «**Обновить**» в окне отчета. По умолчанию, кнопка «**Обновить**» расположена в правом нижнем углу отчета. В ряде случаев удобнее, чтобы отчет обновлялся по нажатию кнопки, расположенной, например, в форме аргументов. Для отключения отображения стандартной кнопки обновления отчета, в окне настройки отчета на вкладке «Свойства» отключите флажок «Показывать кнопку «Обновить»». Создайте собственную кнопку в требуемом месте отчета, в ее свойствах в поле «Щелчок по кнопке» выберите «Обновить» (Рисунок 41).

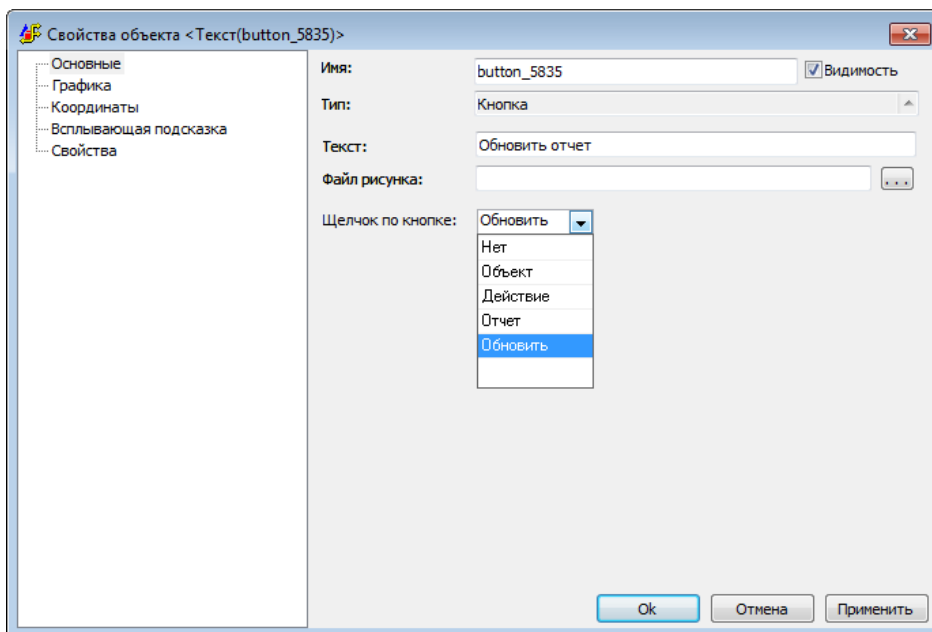


Рисунок 41 Указание свойств кнопки отчета

Кроме того, автоматическое обновление отчета по нажатию кнопки можно осуществить, когда по данной кнопке выполняется действие. В этом случае обновление отчета зависит от положения переключателя «Обновить отчет после выполнения действия» (Рисунок 42).

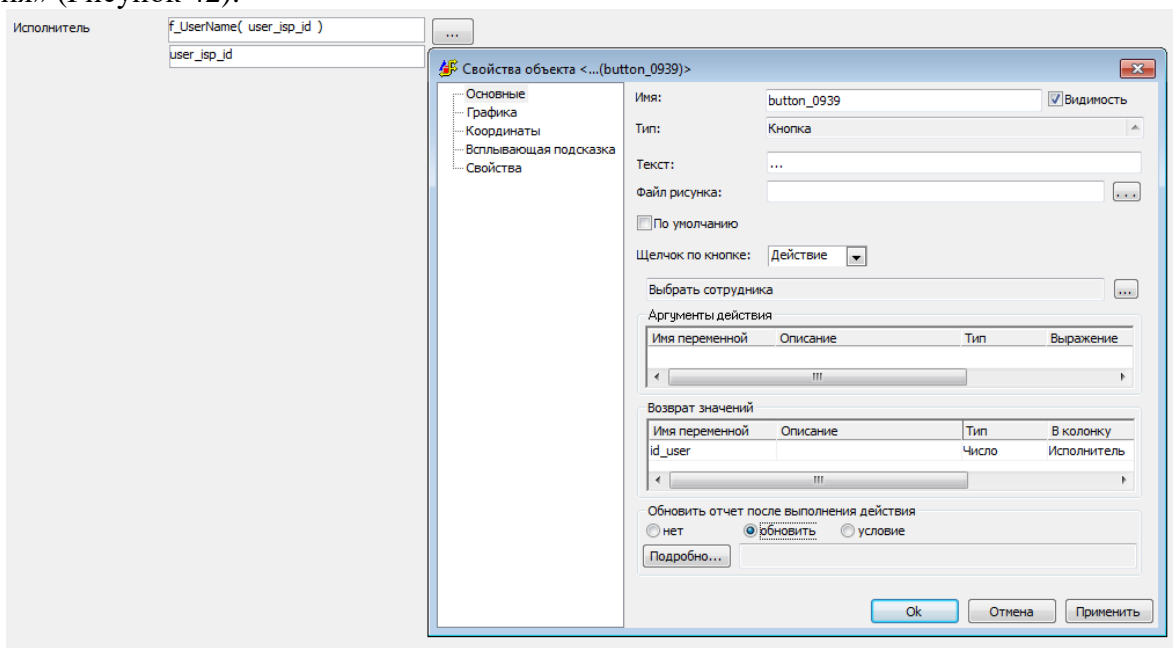


Рисунок 42 Выполнение действия по кнопке в форме аргумента отчета. Пример использования действия для формирования значения аргумента отчета

Отключение показа системной кнопки «Обновить» вызывает и отключение обновления отчета по нажатию клавиши «Enter». Возможно использование клавиши «Enter» для нажатия кнопки в форме аргументов отчета: если флажок «По умолчанию» в свойствах кнопки для формы аргументов отчета (Рисунок 42) включен, то щелчок по этой кнопке будет отрабатываться по нажатию клавиши «Enter». Если указанный флажок включен в свойствах нескольких кнопок, то по нажатию клавиши «Enter» осуществит щелчок по первой попавшейся в форме кнопке. Таким образом, если в свойствах одной из кнопок, кроме включенного флажка «По умолчанию» определено обновление отчета

после выполнения действия – по нажатию клавиши «Enter» выполниться нажатие соответствующей кнопки, и после завершения действия, сопоставленного данной кнопке (данное действие может, например, сформировать значение аргумента отчета) будет обновлен отчет.

Итак, помимо широко применяемого запуска отчета действием, с использованием формы действия для ввода значений аргументов отчета и последующим автоматическим выполнением отчета с переданными из действия значениями аргументов, может быть реализован и следующий механизм:

Запуск отчета без его немедленного выполнения. В открывшемся окне отчета может быть размещена форма аргументов, ввод (редактирование) аргументов в которой возможен различными способами (в том числе, выполнением действий). После выполнения по событию в отчете действия (см. также раздел «») возможно обновление отчета, либо, обновление отчета может осуществляться специальной кнопкой, расположенной в удобном пользователю месте.

25.10 Использование информации из различных объектов при формировании строки отчета. Применение источников данных

В качестве модели данных продолжим использовать проект «Учет компьютеров», см. Рисунок 27, добавив в него информацию о сотрудниках, обслуживающих каждый компьютер (тип объекта – «Сотрудник»). Считаем, что для выполнения обслуживания, за каждым компьютером закреплен один или несколько сотрудников отдела вычислительной техники см. Рисунок 43.

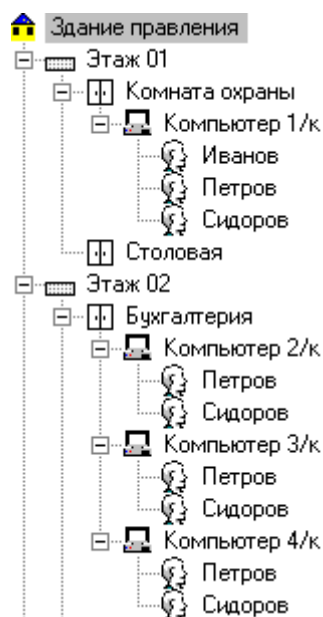


Рисунок 43 Учебный проект «Обслуживание компьютеров». Фрагмент

Для того, чтобы получить сведения о компьютерах и обслуживающем персонале, достаточно сформировать один отчет согласно следующему описанию:

- название отчета – «Обслуживание компьютеров»;
- глубина поиска – все уровни;
- отбор по условиям отчета;
- переключатель «Строка для выделенного объекта отчета обязательна» установите в положение «Нет»;

- направление поиска – вниз по связи «Дерево проектов», в отчет включаются потомки типа «Сотрудник»;
 - отчет состоит из двух колонок: описания родительского объекта (1) и описания потомка (2) (под родительским объектом понимается родитель потомка, в нашем случае это объекты типа «Компьютер»);
 - сортировка и подавление повторяющихся значений – по колонке (1).
- Результат выполнения отчета по объекту «Здание правления» – см. Рисунок 44.

Описание объекта (Родитель)	Описание объекта (Потомок)
Компьютер 1/к	Иванов
	Сидоров
	Петров
Компьютер 10/к	Иванов
Компьютер 11/к	Иванов
Компьютер 2/к	Петров
	Сидоров
Компьютер 3/к	Сидоров
	Петров

Рисунок 44 Результат выполнения отчета «Обслуживание компьютеров». Фрагмент

Нетрудно видеть, что в данный отчет невозможно включить информацию о помещении, в котором установлен каждый компьютер.

Для того, чтобы включить в отчет информацию о связанных с компьютерами объектах верхнего уровня (помещениях), потребуется создать два отчета.

Будем использовать отчет с источником данных, состоящий соответственно, из отчета-источника и отчета-приемника.

Создадим отчет-приемник – «Обслуживание компьютеров по помещениям». Данный отчет будет выполняться по выделенному объекту-помещению, с глубиной поиска «Все уровни», отбор по условиям отчета, переключатель «Строка для выделенного объекта отчета обязательна» установите в положение «Нет», направлением поиска – вниз по связи «Дерево проектов», в отчет включаются потомки типа «Сотрудник». Добавьте в отчет следующие колонки (в учебных целях, для наглядности, сохраним автоматические названия колонок, дополнив их наименованием их содержимого):

1. «Описание объекта (Выделенный объект) – Помещение»;
2. «Описание объекта (Родитель) – Компьютер» (под родительским объектом понимается родитель потомка, то есть «Компьютер»)
3. «Описание объекта (Потомок) – Сотрудник»

Отсортируем строки отчета по первой и второй колонкам, подавите в тех же колонках повторяющиеся значения.

Выполнив данный отчет, используя в качестве текущего объект помещения, убедимся в автономной работоспособности отчета (см. Рисунок 45).

Описание объекта (Выделенный объект) - Помещение	Описание объекта (Родитель) - Компьютер	Описание объекта (Потомок) - Сотрудник
Бухгалтерия	Компьютер 1/к	Петров
		Сидоров
		Иванов
	Компьютер 2/к	Петров
		Сидоров
	Компьютер 3/к	Сидоров
		Петров
	Компьютер 4/к	Петров
		Сидоров

Рисунок 45 Отчет «Обслуживание компьютеров по помещениям», выполненный по объекту «Бухгалтерия»

Однако, в случае, если текущее помещение включает в себя другие помещения, (например, Приемная включает в себя еще и кабинеты руководителей, см. Рисунок 27) данный отчет будет дважды показывать компьютеры, расположенные в кабинетах руководства. Чтобы избежать указанной ситуации, ограничим глубину поиска потомков (объектов типа «Сотрудник»). Нужные нам потомки располагаются на втором уровне от выделенного объекта, на вкладке «Свойства» такая глубина поиска установлена быть не может. Для того, чтобы задать поиск потомков на требуемую глубину, добавим в отчет системную колонку «Уровень», и выполним фильтрацию отчета, выражение фильтра будет следующим:

$level = 2$

Добавим в наш отчет указанный фильтр.

Чтобы сформировать общий отчет по всем помещениям здания, создадим отчет – источник «Помещения здания». Данный отчет предназначен для нахождения всех объектов типа «Помещение», входящих в выделенный объект (например, типа «Здание», или «Этаж»).

Глубина поиска – «Все уровни», отбор по условиям отчета, переключатель «Строка для выделенного объекта отчета обязательна» установим в положение «При отсутствии потомков», направление поиска – вниз по связи «Дерево проектов», в отчет включаются потомки типа «Помещение».

Добавьте в отчет системную колонку «Объект (Код)». После выполнения соответствующей настройки, данная колонка будет передаваться в отчет приемник, и найденные потомки данного отчета будут использоваться в отчете – приемнике в качестве текущих объектов. Данная колонка несет основную функциональную нагрузку, она может быть единственной в данном отчете.

Так как информация в колонке «Объект (Код)» для восприятия человеком не очень удобна, добавим в отчет колонку, содержащую информацию, пригодную для визуальной идентификации найденного объекта-потомка, например, его описание.

Добавим также колонку с информацией о выделенном объекте, например, его описание. После выполнения соответствующей настройки, данная колонка может также быть передана в отчет приемник.

Выполним данный отчет по объекту «Здание правления», убедимся в его правильной автономной работе, см. Рисунок 46.

Объект (код)	Описание объекта (Потомок)	Описание объекта (Выделенный объект)
100000008000037	Комната охраны	Здание правления
100000008100037	Столовая	Здание правления
100000008200037	Комната отдыха	Здание правления
100000008300037	Приемная	Здание правления
100000008400037	Кабинет Генерального директора	Здание правления
100000008500037	Кабинет зам. Генерального директора	Здание правления
100000008600037	Бухгалтерия	Здание правления
100000008700037	Буфет	Здание правления
100000008800037	Отдел №1	Здание правления
100000008900037	Отдел №2	Здание правления

Рисунок 46 Отчет «Помещения здания», автономно выполненный по объекту «Здание правления»

Убедившись, что оба отчета автономно работают правильно, подключим отчет «Помещения здания» в качестве источника данных к отчету «Обслуживание компьютеров по помещениям».

В свойствах аргумента «Объект», в поле «Колонка (источник данных)», укажем «Объект (код)».

Добавим аргумент отчета «desc_a_obj» (см. также раздел «Использование аргументов отчета»), см. Рисунок 47. Данный аргумент будем использовать в вычисляемых полях отчета. Добавим в область данных отчета вычисляемое поле, сделаем колонку данного вычисляемого поля первой в отчете. В выражении вычисляемого поля укажем аргумент desc_a_obj.

Рисунок 47 Добавление аргумента для передачи из отчета источника описания выделенного объекта

Подавим повторяющиеся значения в добавленной колонке (порядок подавления повторяющихся значений важен – см. Рисунок 48).

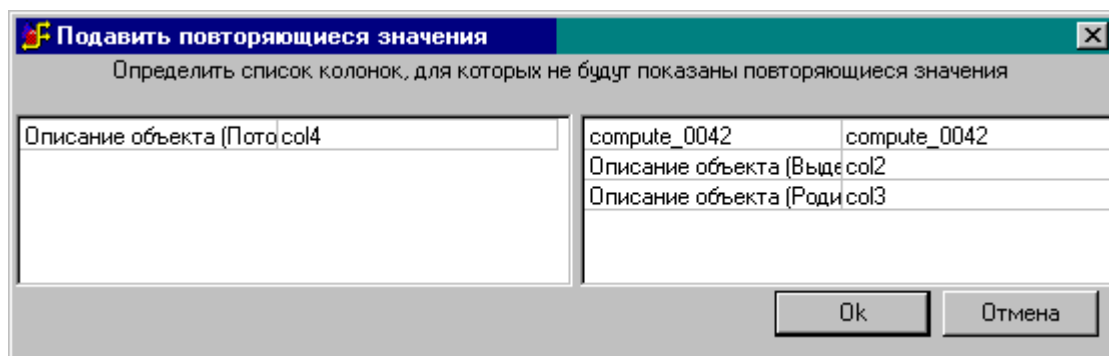


Рисунок 48 Последовательность указания списка колонок для подавления повторяющихся значений

Проверим правильность работы отчета – см. Рисунок 49.

	Описание объекта (Выделенный объект) - Помещение	Описание объекта (Родитель) - Компьютер	Описание объекта (Потомок) - Сотрудник
Здание правления	Бухгалтерия	Компьютер 1/к	Иванов
			Сидоров
			Петров
		Компьютер 2/к	Сидоров
			Петров
		Компьютер 3/к	Петров
			Сидоров
		Компьютер 4/к	Сидоров
			Петров
	Кабинет Генерального директ	Компьютер 6/к	Сидоров
	Кабинет зам. Генерального ди	Компьютер 7/к	Сидоров
	Комната охраны	Компьютер 1/к	Иванов
			Сидоров
			Петров

Рисунок 49 Отчет «Обслуживание компьютеров по помещениям», выполненный по объекту «Здание правления», с использованием отчета «Помещения здания» в качестве источника данных. Фрагмент

25.10.1 Несколько замечаний по использованию отчетов с источниками данных

При использовании отчета с вычисляемыми полями в качестве источника, значения вычисляемых полей можно передавать в отчеты – приемники, аналогично колонкам отчета.

Чтобы использовать в отчете с источником данных аргументы, значения которых будут задаваться пользователем, см. раздел «Использование аргументов отчета», следует:

- требуемые аргументы добавлять сначала в отчет-источник;
- в отчете-источнике создать вычисляемое поле, в выражении которого указать колонку данного аргумента;

- в отчете приемнике создать соответствующий аргумент, указав в поле «Колонка (источник данных)» колонку вышеуказанного вычисляемого поля отчета-источника;
- после выполнения вышеперечисленного, значения аргумента можно использовать в выражениях вычисляемых полей или условий фильтрации отчета-приемника.

При передаче аргументов из отчета-источника в отчет-приемник с использованием вычисляемых полей необходимо также учитывать, что формат аргумента отчета приемника должен соответствовать формату данных вычисляемого поля отчета-источника. Формат данных вычисляемого поля может принимать одно из следующих значений:

- Дата/время (даже если фактически задавалось значение в формате «Дата»)
- Строка
- Число

Один и тот же отчет-источник могут использовать различные отчеты-приемники, отчет-приемник, в свою очередь, может использоваться в качестве отчета-источника для следующего отчета-приемника, и так далее, без ограничения глубины вложенности отчетов.

26 Отображение имени пользователя/группы для значения атрибута с соответствующим идентификатором

Зачастую в атрибутах объектов хранятся не имена пользователей, а их идентификаторы. Но в формах требуется отобразить имена, иначе форма будет нечитаемой. В этом случае можно поступить двумя способами. Первый способ – создать вычисляемое поле с функцией, возвращающей имя пользователя/группы (f_UserName/f_GroupName). Этот способ менее удобен тем, что нужно создавать вычисляемое поле, прятать колонку с идентификатором пользователя/группы, проверять, что содержит идентификатор – пользователя или группу. Второй способ проще: нужно в свойствах колонки с идентификатором пользователя/группы на вкладке «Вид» выбрать в поле «Стиль» значение «Выпадающее окно данных». Затем в поле «Окно данных» нужно выбрать «Пользователи», в поле «Колонка (экран)» – «Name», а в поле «Колонка (данные)» – «Id». Флажок «Показывать стрелку» оставьте выключенным. После сохранения изменений, в поле будет отображаться имя пользователя или группы.

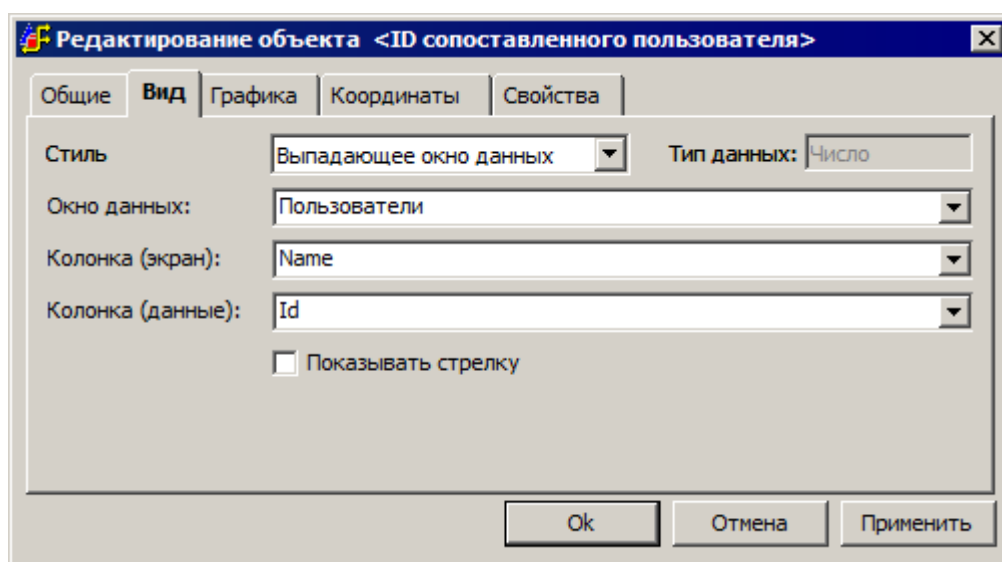


Рисунок 50 Настройка колонки для отображения имени пользователя/группы

27 Организация встреч с использованием календаря

Предварительно рекомендуется ознакомиться с разделом [«Организация встреч» Общей части руководств](#).

Рассмотрим практический пример настройки организации встреч. Создадим шаблон работы и настроим форму для ввода приглашения с информацией о встрече (Рисунок 51).

Рисунок 51 Пример формы для ввода приглашения с информацией о встрече

Для создания встречи используем действие над переменными, содержащее функцию `f_CalEvt_Set`:

`f_CalEvt_Set (EventID, subject, content, datetime (event_date, time ('00:00:00')), time (string(time_start + ':00')), duration, users, author)`

где

`EventID` – строковая переменная, содержащая возвращаемый ID события.

`subject` – поле «Тема».

`content` – поле «Примечание».

`event_date` – поле «Дата».

`time_start` – поле «Время».

`duration` – числовая переменная, продолжительность встречи в минутах.

`users` – строковая переменная, содержащая список ID пользователей – приглашенных, разделенных точкой с запятой («;»). Формируется предыдущим циклическим действием, преобразующим массив пользователей в строку.

`author` – переменная типа «Пользователь» или число. ID инициатора встречи. Может быть либо текущим пользователем, либо вынесено в форму приглашения. Так, к примеру, секретарю может понадобиться создавать приглашения за своего руководителя. Отметим, что в этом случае секретаря тоже понадобится включить в список в переменную `users` для того, чтобы запланированные встречи руководителя отображались в календаре секретаря.

Результат выполнения такого действия: созданное событие типа «Встреча», помещенное в календари приглашенных и автора.

28 Рекомендации по оптимизации работы с информационными объектами

Далее, в настоящем разделе, сведены воедино некоторые часто используемые продвинутые приемы работы с системой Lotsia PDM.

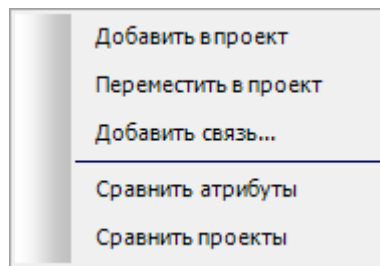
28.1 Быстрый выбор корневых объектов

1. Быстро выбрать корневой объект можно несколькими способами:
 - а. в Окне «Выбор объекта» перейти во вкладку «Последние объекты» – если требуемый корневой объект регулярно Вами выбирается, то его, как правило, можно найти в списке объектов этой вкладки (на этой вкладке хранится до 40 последних объектов);
 - б. в Окне «Выбор объекта» перейти на вкладку «Доп.условия», установить переключатель в позицию **«Объекты, не входящие ни в один проект»** и щелкнуть по кнопке **«Выполнить»**. Система, обработав такой запрос, выдаст список объектов, среди которых будут все корневые. Остается только дважды щелкнуть левой кнопкой мыши по необходимому объекту и выбрать, при необходимости, тип связи. Вы также можете сохранить такой запрос для дальнейшего использования. Данная рекомендация не подходит для такой ситуации, когда требуется построить дерево от узлового объекта.
2. Окно проекта открыто и требуется открыть другое Окно проекта с корневым объектом, выбранным из объектов открытого Окна проекта. Для этого достаточно встать на объект, от которого требуется построить дерево и выбрать в строке меню пункт «Объект» > «Открыть дерева проекта». Новое Окно проекта откроется без запроса типа связи.
3. Создать подборку корневых объектов. Открытие дерева производится двойным щелчком мыши по объекту подборки.
4. В редакторе действий создать действие по открытию Окна проекта.

28.2 Добавление объектов в проект и перемещение объектов между проектами

Встречаются ситуации, когда требуется перетащить объект из одного места текущего дерева в другое, но в Окне проекта не уместятся старый и новый родитель этих объектов (длинное дерево). В этом случае, при традиционном подходе, пользователь «цепляет» мышью перетаскиваемый объект, затем клавишами перемещает курсор вверх или вниз до требуемого родителя и, манипулируя мышью, накладывает на него перемещаемый объект. Значительно упростить эту задачу можно следующим образом. Открыть еще одно аналогичное Окно проекта. Для этого в корне дерева выбрать в строке меню пункт «Объект» > «Открыть дерева проекта». После открытия нового Окна проекта расположить оба Окна рядом друг с другом. В одном Окне расположить дерево так, чтобы были видны перемещаемые объекты. В другом Окне расположить дерево так, чтобы был виден новый родительский объект. Далее следует, «цепляя» левой кнопкой мыши, перетаскивать объекты из одного Окна в другое. Напомним, что таким образом реализуется операция создания связи. Разрыв связи в этом случае не производится и выполняется отдельно (удаление объекта из проекта).

Для перемещения объекта (с автоматическим удалением исходной связи) объект следует перетаскивать с нажатием правой клавиши мыши. При отпускании клавиши



отобразится контекстное меню: , где следует выбрать «Переместить в проект». Данный способ перемещения применим и для объектов с единичной входимостью.

28.3 Удаление объектов из проекта

В ситуации, когда требуется произвести удаление всех или нескольких объектов из проекта, удобны следующие способы:

1. Для удаления всех объектов: поместить в подборку корневой объект проекта, а затем удалить его из базы данных (доступно только администратору). Для этого следует в подборке выделить удаляемый объект и выбрать в строке меню пункт «Правка» > «Удалить объекты из БД...». Получив подтверждение, система произведет удаление объекта. Здесь следует отметить, что если потомки первого уровня удаленного объекта не входят больше ни в один проект, то проект рассыплется. Поэтому, как правило, такая процедура имеет смысл, если все объекты проекта перемещены в другой проект, а вхождение в старый проект требуется удалить. Если же удаляемый объект все-таки нужен, то после удаления, его можно вновь создать. Данный способ не подходит, если на удаляемый из БД корневой объект ссылаются действия, отчеты и т.д. – это приведет к их неработоспособности.
2. Для удаления всех или нескольких объектов: – выделить объект, из которого нужно удалить объекты, перейти на вкладку «Объекты» (или любую другую вкладку с формой типа «Дочерние объекты»), затем выделить на этой вкладке всех потомков, вызвать контекстное меню и выбрать пункт «Удалить...».

28.4 Удаление объекта из проектов

В ситуации, когда требуется произвести удаление объекта из нескольких проектов, удобен следующий способ: выделите объект, который нужно удалить из проектов, перейдите на вкладку «Входимость» (или любую другую вкладку с формой типа «Входимость»), затем выделить на этой вкладке родителей, связь с которыми нужно разорвать, вызвать контекстное меню и выбрать пункт «Удалить...».

28.5 Задание значений атрибутов для большого числа существующих объектов

Зачастую возникает необходимость изменения или добавления атрибутов большому количеству объектов. Объекты могут быть разных типов и иметь разные атрибуты. Поэтому не всегда такое действие удобно реализовать с помощью действий над объектами. Облегчить такую задачу возможно через экспорт-импорт: создать отчет, содержащий системные поля, необходимые для импорта объектов (ID, Type ID) и включить в него колонки, позволяющие визуально идентифицировать объекты. Задать в разделах отчета необходимые типы объектов. Затем выполнить отчет и сохранить его как «Text» или «Text with headers». После этого открыть этот отчет в MS Excel и первым делом все столбцы с выгруженными кодами отформатировать как числовые без знаков

после запятой. Столбцы с датами отформатировать как даты. Если этого не сделать, то после сохранения файла в текстовом формате с разделителями табуляции коды могут сохраниться некорректно. Затем, отведя под каждый атрибут отдельный столбец, заполнить ячейки значениями. Сохранив файл в текстовом формате с разделителями табуляции, произвести импорт объектов, задав последовательность полей.

Время, затрачиваемое на создание такого отчета – не более 3 минут; на выполнение отчета, – в зависимости от размера базы данных, – обычно в пределах 5 минут; на импорт объектов – в зависимости от объема импорта – обычно, в пределах 10 минут. Ввод значений атрибутов занимает значительно большее время, но производить его в MS Excel при больших объемах данных значительно проще и быстрее.