

Министерство образования Республики Беларусь
Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Кафедра информатики

А.А. Волосевич

ТЕХНОЛОГИИ КОРПОРАТИВНОГО ЭЛЕКТРОННОГО ДЕЛОПРОИЗВОДСТВА

Курс лекций
для студентов специальности I-31 03 04 «Информатика»
всех форм обучения

Минск 2006

СОДЕРЖАНИЕ

1. УПРАВЛЕНИЕ ПРОЕКТАМИ	4
1.1. УПРАВЛЕНИЕ ПРОЕКТАМИ: ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ.	4
1.2. МЕТОД КРИТИЧЕСКОГО ПУТИ	7
1.3. МЕТОД PERT	11
1.4. СТОИМОСТНЫЙ АНАЛИЗ ПРОЕКТА	14
1.5. РЕСУРСНЫЙ АНАЛИЗ ПРОЕКТА	17
1.6. ОБЗОР ПРОГРАММНЫХ СРЕДСТВ ДЛЯ УПРАВЛЕНИЯ ПРОЕКТАМИ	20
1.6.1. Системы начального уровня.....	21
1.6.2. Профессиональные системы.....	22
1.7. БАЗОВЫЕ АСПЕКТЫ РАБОТЫ И НАСТРОЙКИ MS PROJECT.	26
1.8. ВИДЫ ЗАДАЧ. ХАРАКТЕРИСТИКИ ОТДЕЛЬНОЙ ЗАДАЧИ	29
1.9. РЕСУРСНОЕ ПЛАНИРОВАНИЕ В MS PROJECT	34
1.10. СТОИМОСТНЫЙ АНАЛИЗ ПРОЕКТА В MS PROJECT	41
1.11. АНАЛИЗ ПРОЕКТА ПО МЕТОДУ PERT	43
1.12. АНАЛИЗ РИСКОВ	45
1.13. ОТСЛЕЖИВАНИЕ ХОДА ВЫПОЛНЕНИЯ ПРОЕКТА	47
2. ЭЛЕКТРОННОЕ ДЕЛОПРОИЗВОДСТВО	49
2.1. ДОКУМЕНТ, ДОКУМЕНТООБОРОТ, ДЕЛОПРОИЗВОДСТВО	49
2.2. ЭЛЕКТРОННЫЙ ДОКУМЕНТ	51
2.3. ЭЛЕКТРОННЫЙ ДОКУМЕНТООБОРОТ	54
2.3.1. Системы автоматизации делопроизводства	56
2.3.2. Архивы документов.....	57
2.3.3. Системы ввода и системы обработки образов документов.....	57
2.3.4. Системы управления стоимостью хранения документов.....	58
2.3.5. Системы маршрутизации документов.....	58
2.3.6. Системы комплексной автоматизации бизнес-процессов.....	59
2.3.7. Уровни внедрения систем электронного документооборота	59
2.4. LOTUS NOTES/DOMINO – ОБЩАЯ ХАРАКТЕРИСТИКА	61
2.5. УПРАВЛЕНИЕ ДОСТУПОМ В LOTUS NOTES/DOMINO	63
2.6. СОЗДАНИЕ БАЗЫ ДОКУМЕНТОВ В LOTUS NOTES/DOMINO	65
2.7. СТРУКТУРА И СВОЙСТВА БАЗЫ ДОКУМЕНТОВ	69

2.8. СОЗДАНИЕ ФОРМ. СВОЙСТВА ФОРМЫ	76
2.9. ПОЛЯ ФОРМЫ	83
2.10. СОЗДАНИЕ ПРЕДСТАВЛЕНИЙ И ПАПОК	89
2.11. СОЗДАНИЕ ДЕЙСТВИЙ.....	100
2.12. ЯЗЫК ФОРМУЛ.....	104
2.12.1. <i>Функции для управления ходом выполнения в формуле</i>	<i>106</i>
2.12.2. <i>Функции для работы с полями</i>	<i>107</i>
2.12.3. <i>Функции для работы со списками и строками.....</i>	<i>108</i>
2.12.4. <i>Функции для выборки информации</i>	<i>110</i>
2.12.5. <i>Функции для осуществления диалога с пользователем</i>	<i>112</i>
2.13. ЯЗЫК LOTUS SCRIPT	113
2.14. DOMINO OBJECT MODEL.....	121
2.14.1. <i>Обработка событий в LotusScript</i>	<i>132</i>
2.15. СОЗДАНИЕ АГЕНТОВ ДЛЯ БАЗ LOTUS/DOMINO.....	134
2.16. ПУБЛИКАЦИЯ БАЗ LOTUS/DOMINO В WEB.....	137
2.16.1. <i>Страницы</i>	<i>137</i>
2.16.2. <i>Схемы.....</i>	<i>140</i>
2.16.3. <i>Фреймсеты</i>	<i>142</i>
ЛИТЕРАТУРА	146

1. УПРАВЛЕНИЕ ПРОЕКТАМИ

1.1. УПРАВЛЕНИЕ ПРОЕКТАМИ: ОСНОВНЫЕ ПОНЯТИЯ И ОПРЕДЕЛЕНИЯ.

Управление проектами – это особая область менеджмента. История управления проектами началась одновременно с историей человека разумного. В определенном смысле проектом можно считать и организацию охоты на мамонта, и постройку египетских пирамид, и экспедицию Магеллана. Так что попытка отыскать самый первый проект обречена на неудачу. Результаты одних проектов мы с вами видим до сих пор, а о других можем судить лишь по описаниям современников.

Рассмотрение управления проектами как отдельной научной дисциплины принято связывать с именем Генри Ганта (Henry Gantt). Гант является создателем методики *диаграмм Ганта*, разработанной для отслеживания хода строительства больших трансконтинентальных океанских лайнеров во время Первой мировой войны. Идея Ганта состояла в том, что главным ресурсом планирования является время, а основой принятия управленческих решений – сравнение запланированного и фактического состояния работ. На диаграммах Ганта по горизонтали обычно показывают интервалы времени, а по вертикали – работы, операции, оборудование. Горизонтальные отрезки отражают длительность выполнения работ. Выбрав текущий момент времени и получив оперативную информацию о ходе производства, можно сопоставить фактическое состояние дел и планировавшееся.

В 1956 году М. Уолкер из фирмы DuPont, исследуя возможности более эффективного использования принадлежащей фирме вычислительной машины Univac, объединил свои усилия с Д. Келли из группы планирования капитального строительства фирмы «Ремингтон Рэнд». Они попытались использовать ЭВМ для составления планов-графиков крупных комплексов работ по модернизации заводов фирмы DuPont. В результате был создан рациональный метод описания проекта с использованием ЭВМ – *метод критического пути* (Critical Path Method, CPM). Данный метод имеет три достоинства – позволяет получить графическое представление проекта, определяет ориентировочное время, требуемое для его выполнения, и показывает, какие действия критичны, а какие не столь важны для соблюдения всего графика работ.

Для задач, связанных с интеллектуальным трудом и другими вопросами, в которых стоимость оптимизируемого параметра не известна наверняка, используется *метод PERT-анализа* (Program Evaluation Review Technique). Он был разработан сотрудниками Военно-морского флота США в 1957 году для обеспечения создания ракеты «Полярис». Применяя PERT-анализ, они попытались сымитировать график выполнения работ по созданию ракеты путем построения логической сети взаимозависимых последовательных событий. На начальной стадии PERT-представление было сфокусировано на контроле временных характеристик графика и прогнозировании вероятности успешного завершения

программы. Но прежде чем PERT-представление было окончательно принято руководителями программ в промышленности, Военно-воздушные силы США внесли дополнение в методику, добавив к логической сети функцию ресурсной оценки. Таким образом, в 1962 году появилась PERT/Cost-методика (PERT-анализ с целью стоимостного прогнозирования), в то время как первоначально PERT-анализ был известен под названием PERT/Time (PERT-анализ для определения времени реализации проекта). Использование метода PERT позволило руководству программы точно знать, что требуется делать в каждый момент времени и кто именно должен это делать, а также какова вероятность своевременного завершения отдельных операций. Руководство программой оказалось настолько успешным, что проект удалось завершить на два года раньше запланированного срока. Благодаря такому впечатляющему началу, данный метод управления вскоре стал использоваться для планирования проектов во всех вооруженных силах США. Методика отлично себя зарекомендовала при координации работ, выполняемых различными подрядчиками в рамках крупных проектов по разработке новых видов вооружения.

После короткого исторического экскурса в методики управления проектами представим набор формальных определений, используемых в данной дисциплине.

Проект – это временное предприятие, предназначенное для создания уникальных продуктов или услуг. «Временное» означает, что у любого проекта есть начало и непременно наступает завершение, когда достигаются поставленные цели, либо возникает понимание, что эти цели не могут быть достигнуты. «Уникальных» означает, что создаваемые продукты или услуги существенно отличаются от других аналогичных продуктов и услуг.

В качестве примеров проектов можно привести строительство, разработку любой новой продукции, проведение ремонтных работ, внедрение информационной системы на предприятии, проведение избирательной кампании, съемки кинофильма и многое другое, что отвечает приведенному определению.

Проект состоит из процессов. *Процесс – это совокупность действий, приносящая результат.* Процессы проекта обычно выполняются людьми и распадаются на две основные группы: процессы управления проектом (общие для любого проекта) и процессы, ориентированные на продукт (специфичные для конкретного проекта и продукта).

Процессы управления проектами могут быть разбиты на шесть основных групп, реализующих различные функции управления:

1. *процессы инициации* – принятие решения о начале выполнения проекта;
2. *процессы планирования* – определение целей и критериев успеха проекта и разработка рабочих схем их достижения;
3. *процессы исполнения* – координация людей и других ресурсов для выполнения плана;
4. *процессы анализа* – определение соответствия плана и исполнения проекта поставленным целям и критериям успеха и принятие решений о необходимости применения корректирующих воздействий;

5. *процессы управления* – определение необходимых корректирующих воздействий, их согласование, утверждение и применение;
6. *процессы завершения* – формализация выполнения проекта и подведение его к упорядоченному финалу.

Процессы управления проектами накладываются друг на друга и происходят с разной интенсивностью на всех стадиях проекта. Кроме этого, процессы управления проектами связаны своими результатами – результат выполнения одного становится исходной информацией для другого.

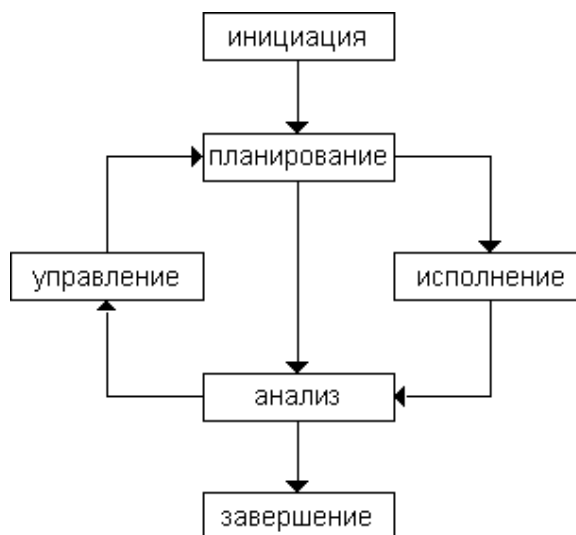


Рис. 1. Взаимосвязи процессов управления

Процессы, ориентированные на продукт, служат основой для построения проектного плана. Проектный план состоит из трех основных элементов: задач ресурсов и назначений.

Задача – это работа в рамках проекта для достижения определенного результата. В некотором смысле термины «задача» и «процесс, ориентированный на продукт» являются синонимами. Иногда задачи проекта объединяются в более крупные группы, называемые *фазами*. При этом в фазе обычно выделяется *завершающая задача (веха)*, которая обозначает результат фазы. Задача характеризуется такими параметрами как *длительность* – период рабочего времени, необходимый для выполнения задачи, и *трудозатраты* – время, которое потратил сотрудник на получение результата задачи.

Ресурсы – это сотрудники и оборудование, необходимые для выполнения задач проекта. Ресурсы обычно имеют свойство *стоимости*. Стоимость ресурсов бывает двух видов: повременная ставка и однократная стоимость за использование. Также ресурсы можно разделить на *невосполнимые* (например, физические материалы) и *восполнимые* (рабочая сила).

Назначение задает связь между работой и ресурсом.

Наглядной иллюстрацией основных характеристик проекта и связей между ними является *проектный треугольник* (рис. 2).



Рис. 2. Основные характеристики проекта

Время, стоимость и объем работ – взаимосвязанные величины. Для уменьшения времени, за которое выполняется проект, можно либо увеличить стоимость проекта (чтобы привлечь дополнительные трудовые ресурсы), либо уменьшить объем работ, которые необходимо выполнить. Таким образом, имеем следующее соотношение (k_1 и k_2 – некоторые коэффициенты):

$$t = \frac{k_1 V}{k_2 c}.$$

Без учета коэффициентов получаем следующую наглядную формулу, позволяющую оценить стоимость проекта:

$$c = \frac{V}{t}.$$

1.2. МЕТОД КРИТИЧЕСКОГО ПУТИ.

Рассмотрим некоторые математические методы, используемые в управлении проектами. Начнем с *метода критического пути* (СРМ).

Для математического описания, анализа и оптимизации проектов наиболее подходящими оказались *сетевые модели*, представляющие собой разновидность ориентированных графов. В сетевой модели роль вершин графа могут играть события, определяющие начало и окончание отдельных работ, а дуги в этом случае будут соответствовать работам. Такую сетевую модель принято называть *сетевой моделью с работами на дугах* (Activities on Arrows, AoA). В то же время возможно, что в сетевой модели роль вершин графа играют работы, а дуги отображают соответствие между окончанием одной работы и началом другой. Такую сетевую модель принято называть *сетевой моделью с работами в узлах* (Activities on Nodes, AoN).

Сетевая модель может быть представлена в нескольких видах. Наиболее распространенными формами представления являются сетевой график и табличная форма.

Рассмотрим следующий пример представления проекта в табл. 1.

Сетевая модель проекта в табличной форме

Номер работы	Срок выполнения	Каким работам предшествует
1	2	2, 3
2	3	8
3	4	6, 7, 10
4	5	6, 7, 10
5	4	9
6	6	8
7	4	–
8	2	–
9	5	11
10	1	11
11	2	–

В табл. 1 для каждой работы указаны работы-«последовательницы». Возможен вариант, когда для каждой работы указываются работы-«предшественницы».

Перейдем от табличной формы к сетевой модели с работами на дугах. Для этого вначале определим события (вершины графа). Начальное событие является исходным для работ, которым ничего не предшествует (их нет в третьей колонке таблицы). Завершающее событие – это окончание работ, у которых нет «последователей» (прочерк в третьей колонке таблицы).

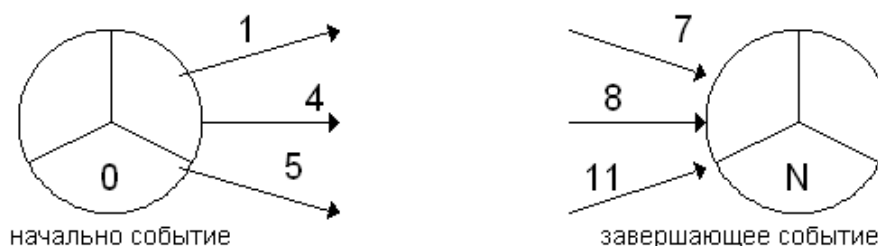


Рис. 3. Сетевая модель: начальное и завершающее события

Остальные события определяются на основе анализа строк таблицы (но пока не получают номера).

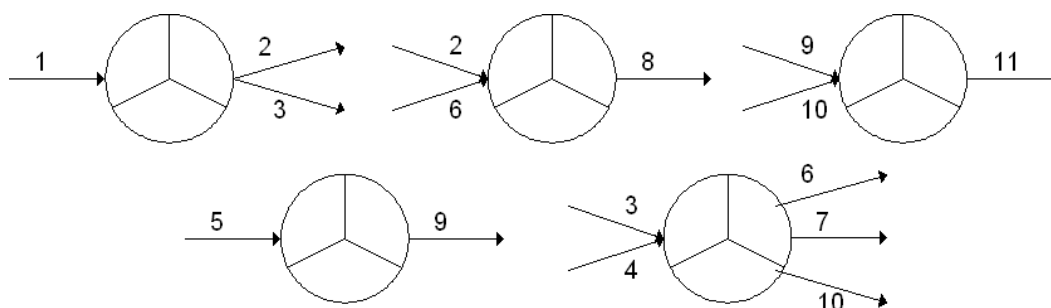


Рис. 4. Сетевая модель: набор событий

Для нумерации событий и построения сетевой модели используется следующий *алгоритм ранжирования*. К нулевому рангу относится начальное событие. Все работы, исходящие из начального события, вычеркиваются. На каж-

дом шаге алгоритма определяются события, которые относятся к очередному рангу. Это события, у которых вычеркнуты все входящие работы. Затем у всех событий без ранга вычеркиваются те входящие работы, которые являются исходящими у событий очередного ранга.

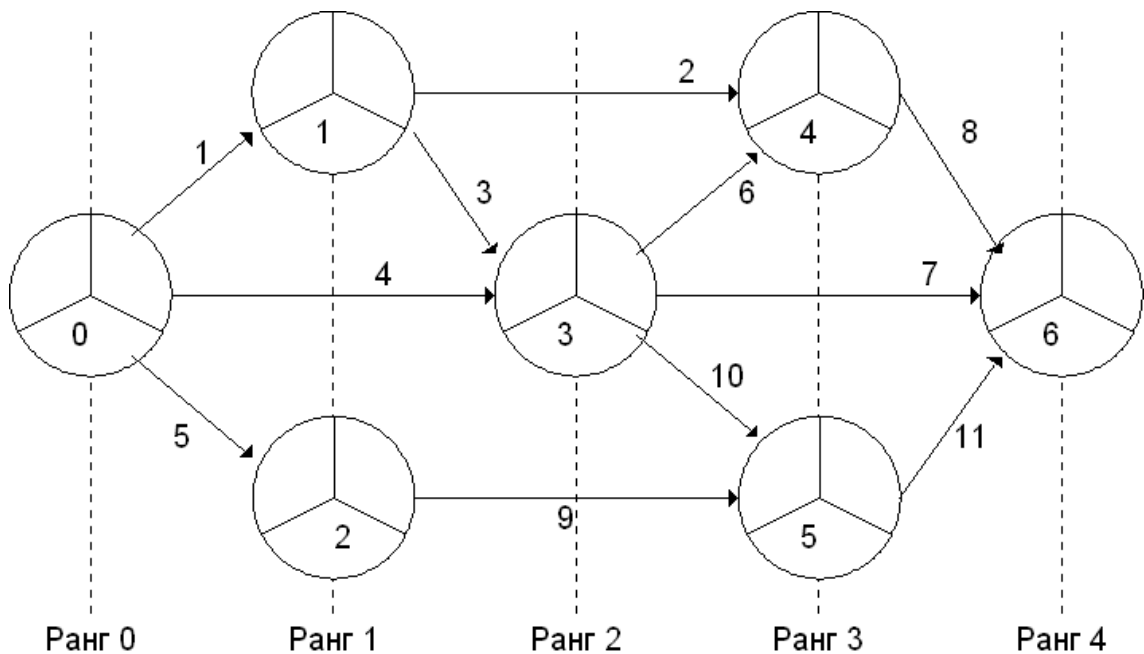


Рис. 5. Сетевая модель, полученная после алгоритма ранжирования

Рассмотрим на примере данной сетевой модели метод критического пути. Введем некоторые определения. Пусть дан путь от события K до события L . *Длиной пути* назовем сумму продолжительностей работ, которые составляют этот путь. *Критическое время* T_N – это длина самого длинного пути от начального события до завершающего. Смысл критического времени следующий: за время, меньшее, чем T_N , проект выполнить нельзя; работы, которые составляют путь, давший критическое время требуют особого внимания (задержка их выполнения увеличивает общий срок проекта).

Как найти критическое время и путь? Для этого будем использовать следующие величины. Через T_J^0 обозначим *наиболее ранний срок наступления события J* . Это самый длинный путь от начального события до события J . Данная величина вычисляется по следующей формуле:

$$T_J^0 = \begin{cases} 0, & J = 0, \\ \max_{I < J} \{T_I^0 + t_{IJ}\}, & J \neq 0. \end{cases}$$

Здесь t_u обозначает продолжительность работы между событиями I и J . Величина T_J^0 вычисляется для всех событий последовательно, начиная с начального события. Значение T_N^0 даст величину T_N .

Через T_J^1 обозначим *наиболее поздний допустимый срок наступления события J* . Если событие J наступит после T_J^1 , то срок исполнения проекта неизбежно увеличится. Данная величина вычисляется по формуле:

$$T_J^1 = \begin{cases} T_N, & J = N, \\ \min_{I>J} \{T_I^1 - t_{IJ}\}, & J \neq N. \end{cases}$$

Величина T_J^1 вычисляется для всех событий последовательно, начиная с завершающего события.

Имеет место следующее утверждение: **работа (I, J) является критической, тогда и только тогда, когда $T_J^0 - T_I^1 = t_{IJ}$.**

Вычислим величины T_J^0 и T_J^1 для нашей сетевой модели. Будем записывать величину T_J^0 в узле сети слева, а величину T_J^1 – справа.

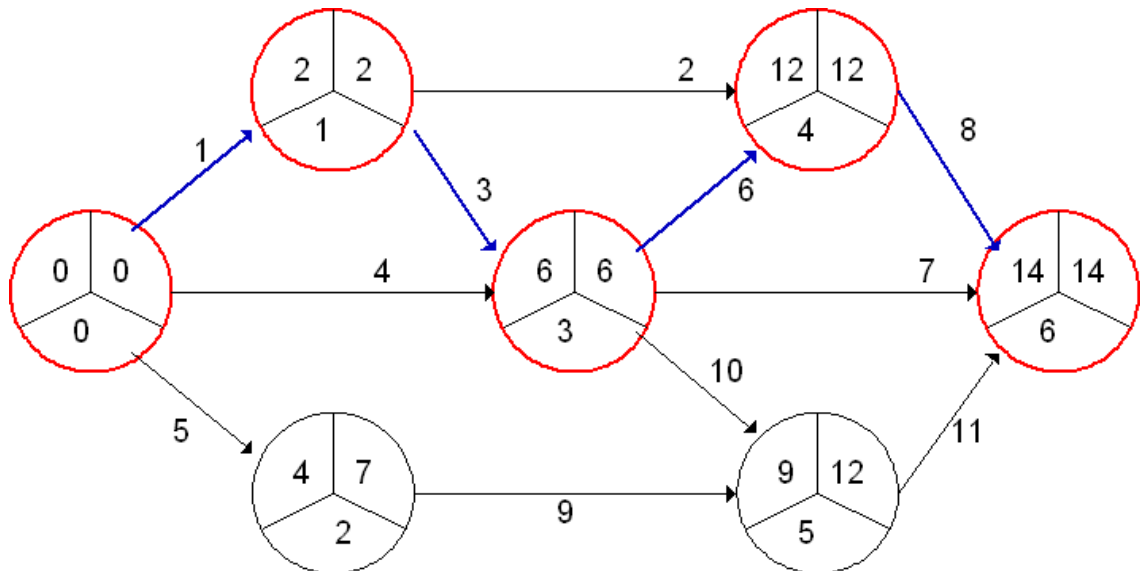


Рис. 6. Критический путь

Для критических работ $T_J^0 = T_J^1$ (обратное, вообще говоря, не верно). Таким образом, мы получили критический путь длиной 14, состоящий из работ 1, 3, 6, 8.

Для дальнейшего анализа удобными оказываются следующие величины. *Полный резерв времени* для работы (I, J) это величина $R_{IJ}^H = T_J^1 - T_I^0 - t_{IJ}$, которая показывает, на сколько можно задержать выполнение работы (I, J) , не изменяя критического времени. *Свободный резерв времени* $R_{IJ}^C = T_J^0 - T_I^0 - t_{IJ}$ для работы (I, J) показывает, на сколько можно задержать выполнение работы (I, J) , не сдвигая (не задерживая) всех остальных работ.

1.3. МЕТОД PERT.

Рассмотрим применение метода PERT в проектном планировании. В реальной жизни очень часто приходится сталкиваться с ситуациями, когда продолжительность работ не может быть определена точно, а лишь приблизительно. В принципе, могут иметь место два случая:

работы не являются новыми, и мы знаем приблизительно закон распределения продолжительности каждой из них;

работы совершенно новые для нас, и закон распределения продолжительности их выполнения нам неизвестен.

В первом случае, так как закон распределения продолжительности работы известен, то известны и два его параметра: *математическое ожидание* m продолжительности выполнения работы и *дисперсия* σ^2 продолжительности выполнения.

Во втором случае, когда точный закон распределения продолжительности работ неизвестен, предполагается, что это распределение подчиняется нормальному закону и описывается β -*функцией*, которая имеет следующие математическое ожидание и дисперсию:

$$m = \frac{O + 4M + P}{6},$$
$$\sigma^2 = \left(\frac{O - P}{6} \right)^2.$$

Здесь O обозначает оптимистическую оценку продолжительности работы, P – пессимистическую оценку, M – наиболее вероятную оценку продолжительности. Таким образом, в любом случае для оценки продолжительности любой работы мы будем иметь *ожидаемое время* (математическое ожидание) и *погрешность* (дисперсию) этого ожидания.

Процедура построения и разметки сетевого графика в случае со случайной продолжительностью работ ничем не отличается от той, что используется в случае с детерминированной продолжительностью работ. Однако продолжительность найденного критического пути также будет иметь две оценки – ожидаемую и погрешность. Ожидаемая продолжительность критического пути равна сумме ожидаемых продолжительностей критических работ, а погрешность продолжительности критического пути равна сумме дисперсий критических работ. В этом случае говорить о том, что комплекс работ будет завершен к какой-то определенной дате (т. е. будет иметь какую-то фиксированную продолжительность выполнения T), можно лишь с некоторой вероятностью $P(T)$. Данная вероятность рассчитывается на основе таблиц для случайной величины, распределенной по нормальному закону. При этом в качестве параметра выступает масштабированная величина T_N , равная $T_N = (T - m_k)/\sigma_k$, где m_k – ожидаемая продолжительность критического пути, а σ_k – квадратный корень из погрешности продолжительности критического пути.

Рассмотрим в качестве примера сетевую модель, представленную табл. 2.

Таблица 2

Сетевой график в случае со случайной продолжительностью работ

Номер работы	Оптимистическая оценка продолж.	Наиболее вероятная оценка продолжит.	Пессимистическая оценка продолж.	Каким работам предшествует
1	1	2	3	2, 3
2	2	3	4	8
3	2	4	6	6, 7, 10
4	2	5	8	6, 7, 10
5	2	4	5	9
6	4	6	7	8
7	2	4	10	–
8	1	2	3	–
9	2	5	8	11
10	1	1	3	11
11	1	2	4	–

Для каждой работы ожидаемое время и погрешность содержится в табл. 3.

Таблица 3

Ожидаемое время и погрешность

Номер работы	Оптимистическая оценка продолж.	Наиболее вероятная оценка продолж.	Пессимистическая оценка продолж.	Ожидаемое время	Погрешность
1	1	2	3	2	0,11
2	2	3	4	3	0,11
3	2	4	6	4	0,44
4	2	5	8	5	1
5	2	4	5	3,83	0,25
6	4	6	7	5,83	0,25
7	2	4	10	4,67	1,78
8	1	2	3	2	0,11
9	2	5	8	5	1
10	1	1	3	1,3	0,11
11	1	2	4	2,17	0,25

Построим сетевую модель и найдем критический путь, используя в качестве параметров ожидаемое время работы.

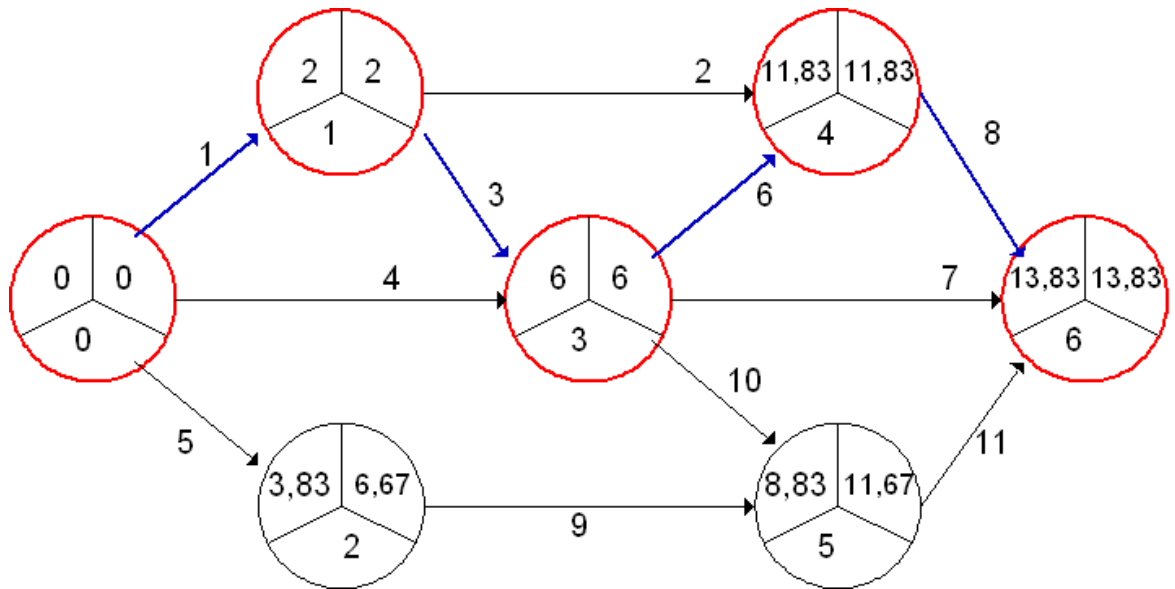


Рис. 7. Сетевая модель для метода PERT

Критический путь, состоящий из работ 1, 3, 6, 8, имеет ожидаемую продолжительность 13,83, а его суммарная погрешность равна $0,11 + 0,44 + 0,25 + 0,11 = 0,91$.

Однако полученная ожидаемая продолжительность критического пути не означает, что весь комплекс работ, описанный сетевым графиком, будет завершен именно в течение данного промежутка времени. Утверждать, что этот комплекс работ будет завершен именно в данный промежуток времени, можно только с вероятностью 0,5, так как:

$$P(13,83) = F((13,83 - 13,83) / 0,95) = F(0) = 0,5.$$

С таким же успехом можно определить вероятность завершения комплекса работ до любого директивного срока X . Пусть, например, $X = 15$. Тогда:

$$P(15) = F((15 - 13,83) / 0,95) = F(1,23) = 0,8907 \text{ (89\%)}$$

Кроме этого, можно решить и **обратную задачу**, то есть определить тот срок, к которому рассматриваемый комплекс работ может завершиться с некоторой заданной вероятностью P_D . Зная P_D , можно воспользоваться нормальным стандартным распределением (в форме таблиц или с помощью известной функциональной зависимости, описываемой интегралом нормального стандартного распределения) и найти x , при котором $F(x) = P_D$. Продолжительность критического пути T_D , соответствующая заданной вероятности P_D , будет равна

$$T_D = x\sigma_k + m_k$$

Так, для рассматриваемого здесь примера промежуток времени, в течение которого комплекс работ, описываемых сетевым графиком, будет завершен с вероятностью 0,95, равен:

$$P_D = 0,95 \Rightarrow F(x) = 0,95 \Rightarrow x = 1,645 \Rightarrow T_D = 1,645 * 0,95 + 13,85 = 15,41.$$

1.4. СТОИМОСТНЫЙ АНАЛИЗ ПРОЕКТА

Рассмотрим некоторые математические методики для стоимостного анализа проекта. Прежде всего, заметим, что любая работа проекта имеет прямую и косвенную стоимость. *Прямая стоимость работы* – это стоимость материалов, затраты на зарплату. Если необходимо сократить время выполнения работы, прямая стоимость работы обычно возрастает, так как работа требует *больших* ресурсов. *Косвенная стоимость работы* обычно не связана с конкретным проектом. Это налоги, стоимость аренды помещений и т. п. При увеличении длительности работы косвенная стоимость возрастает. Отсюда можно сделать важный вывод: **зависимость стоимости работы от продолжительности работы носит нелинейный характер.**

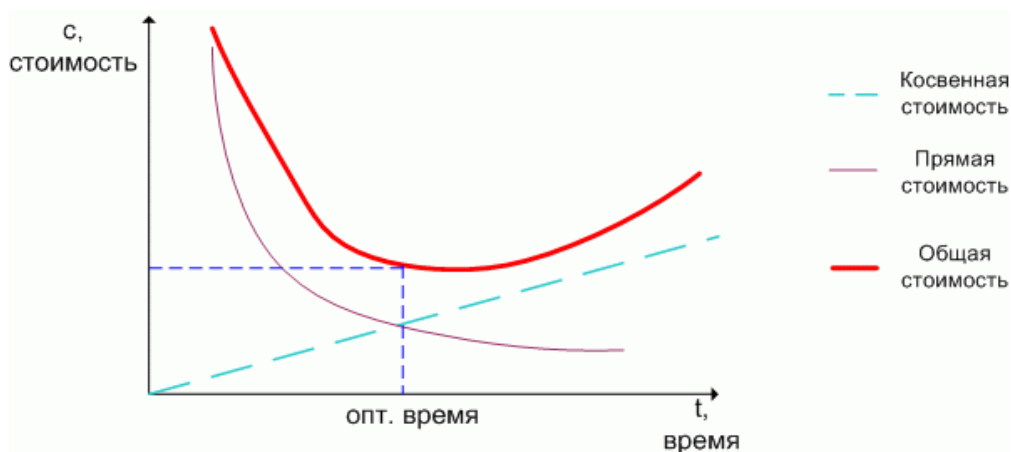


Рис. 8. Зависимость стоимости работы от времени

Однако полученную нелинейную зависимость обычно сознательно упрощают. Полагают, что любая работа имеет «нормальную» продолжительность d_i и «нормальную» стоимость c_i , а также «сжатую» продолжительность D_i и большую, «сжатую» стоимость C_i . Зависимость стоимости от длительности на отрезке $[D_i, d_i]$ предполагается линейной.

Можно сформулировать несколько задач планирования с использованием стоимости. В частности, рассмотрим задачу уменьшения сроков проекта за счет увеличения его стоимости. Предположим, что проект задан при помощи табл. 4.

Таблица 4

Описание проекта (длительность и стоимости работ)

Работа	Предшеств. работы	Нормальные сроки		Сжатые сроки		Приращение затрат
		продолж.	стоим.	продолж.	стоим.	
A	–	9	900	3	6300	900
B	–	7	2800	6	3300	500
C	A	10	7000	2	16600	1200
D	A	12	8400	6	13800	900
E	B	12	7200	4	12800	700
F	D, E	6	4900	6	4900	0
G	D, E	6	3000	4	6200	1600
H	G	14	4200	12	5200	500
I	C, F	8	3200	3	6700	700

Если использовать нормальные продолжительности работ, то получится следующая сетевая модель проекта.

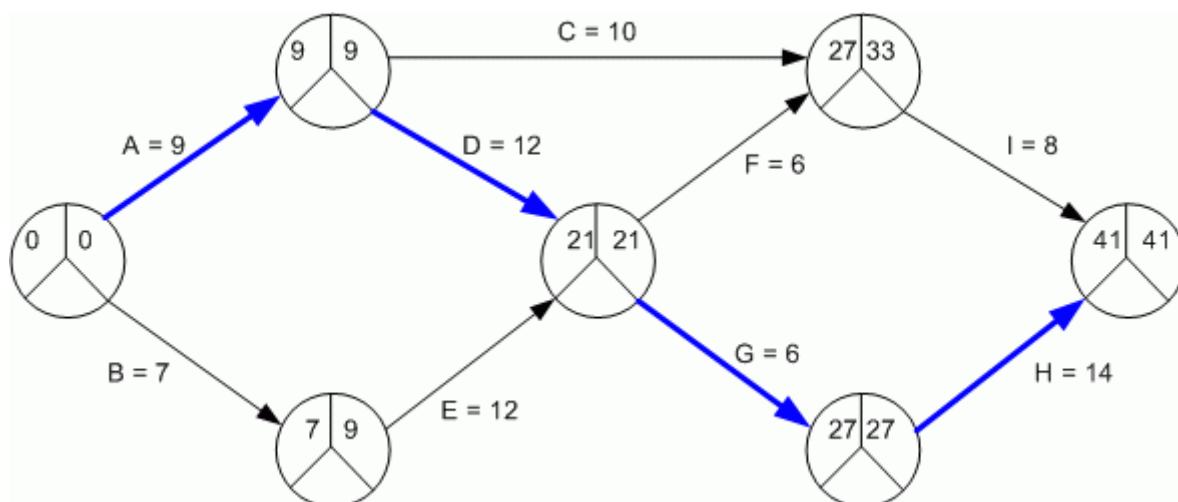


Рис. 9. Сетевая модель – нормальная продолжительность работ

Продолжительность проекта – 41 день. Общие затраты на проект составляют 41600 условных единиц.

При использовании сокращенных сроков работ получим следующую модель, общие затраты при этом 75800 условных единиц, а продолжительность – 26 дней.

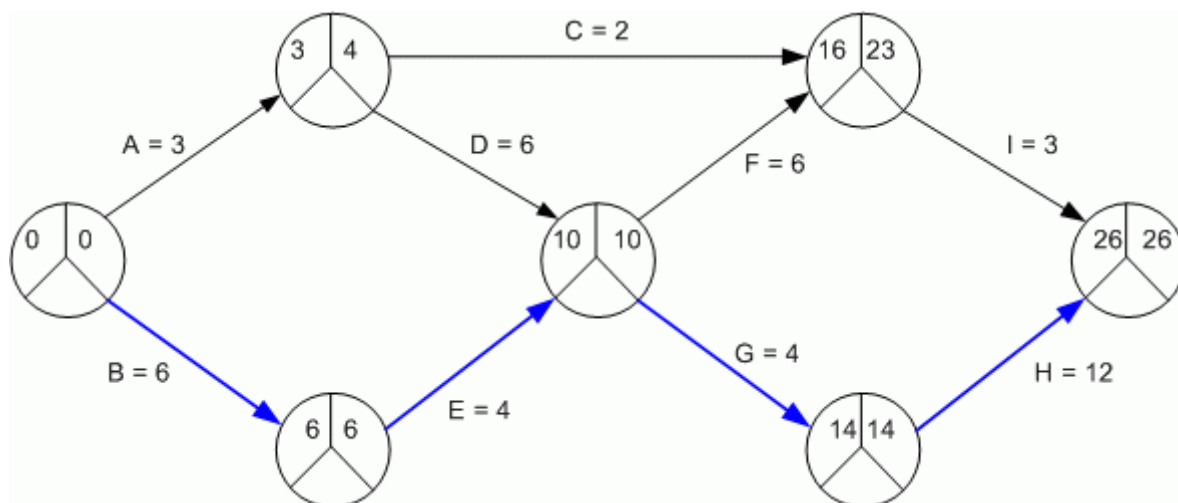


Рис. 10. Сетевая модель – сокращенные сроки работ

Новый сетевой план задает своеобразную «границу» возможной оптимизации исходной задачи. Обратите внимание: в новой модели другой критический путь.

Имея две сетевых модели далее можно поступить одним из двух способов:

1. В «сокращенной» модели растянуть выполнение не критических работ. При этом следует учитывать свободный резерв времени для работы и нормальную продолжительность работы. Для нашего примера получим табл. 5.

Новая сетевая модель

Работа	Свободный резерв	Нормальная продолж.	На сколько растягиваем	Сэкономили затрат
A	0	9	-	
C	11	10	8	9600
D	1	12	1	900
F	0	6	-	
I	7	8	5	3500

Общая экономия затрат составляет 14000 условных единиц.

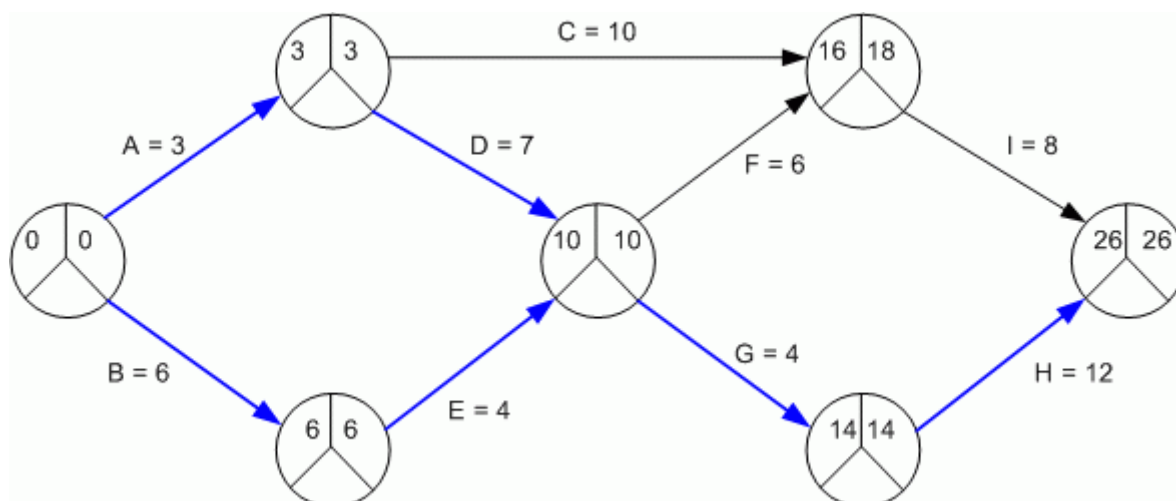


Рис. 11. Новая сетевая модель проекта

Обратите внимание на то, какие работы стали критическими.

2. В «нормальной» модели последовательно убыстрять выполнение отдельных работ. При этом убыстрять выполнение некритических работ не имеет смысла. Можно использовать такой эмпирический алгоритм.

Рассмотреть работы, которые лежат на критическом пути.

Выбрать ту работу, убыстрение которой наиболее дешево.

Уменьшить продолжительность данной работы на фиксированную величину (например, 1 день).

Пересчитать стоимость проекта и критический путь (он может измениться).

Повторить описанные шаги.

Признаком завершения алгоритма может служить достижение заданной продолжительности проекта или верхней границы стоимости проекта.

В нашем примере в нормальной модели критический путь составляют работы A, D, G, H. Дешевле всего начать ускорение с работы H. Ее можно ускорить на 2 дня (до сжатого срока в 12 дней) без изменения критического пути. Это потребует $2 \cdot 500 = 1000$ условных единиц. Далее можно ускорять работы A или D. Прodelайте самостоятельно несколько шагов алгоритма в качестве упражнения.

1.5. РЕСУРСНЫЙ АНАЛИЗ ПРОЕКТА

При планировании проекта одной из важных задач является сглаживание потребности в ресурсах. Любая работа может требовать для своего выполнения некоторых ресурсов. В простейшем случае существует только один однотипный ресурс для всех работ. На практике повсеместно приходится сталкиваться с ситуацией, когда потребность в том или ином виде физического ресурса в конкретный момент времени превышает имеющиеся возможности его обеспечения. Такие ситуации возникают в силу следующих причин:

- Стремление сократить время выполнения работы приводит к неправильному решению в отношении выделяемых на нее ресурсов. Это достаточно тривиальная ситуация, как правило, обусловленная невнимательным отношением к ограничениям по проекту. Нельзя назначить на выполнение работы, скажем, трех исполнителей, если в наличии только два. Такую ситуацию легко избежать при использовании компьютерных систем поддержки проектного управления, в которых запрограммирована процедура проверки на непротиворечивость условий проекта.

- Для каждой отдельной работы проекта ограничения по ресурсам соблюдены, но топология сетевой модели проекта оказывается причиной запараллеливания нескольких работ, предусматривающих использование одинаковых ресурсов. Это приводит к соответствующему увеличению суммарной потребности в них в определенные моменты времени. Возникает конфликтная ситуация: в рассматриваемый момент времени потребность в ресурсах превышает возможности, а значит, для каких-то из работ оказывается невозможным осуществить выполнение так, как это предполагается текущим планом. Данная ситуация, как правило, становится предметом тщательного анализа, поскольку требует своего разрешения на стадии планирования проекта. Конфликт должен и может быть разрешен с помощью перепланирования проекта. Целью такого перепланирования должно быть либо максимальное сокращение перерасхода ресурсов без увеличения общего времени выполнения проекта, либо приведение потребности в ресурсах в соответствие с установленными ограничениями (пусть даже за счет некоторого удлинения сроков выполнения проекта), либо комбинация этих двух подходов. В любом случае речь идет о сглаживании потребности в ресурсах. В первом случае как бы предполагается, что имеются четкие ограничения «по вертикали», то есть по срокам осуществления проекта. Во втором случае – что имеются четкие ограничения «по горизонтали», то есть по суммарной потребности в ресурсах. В третьем случае – что имеются четкие установки относительно общей стоимости проекта, а именно, что она должна быть минимальна.

Общие принципы сглаживания потребности в ресурсах очень просты.

Первый принцип основан на том, что, как правило, многие из параллельно запланированных работ, требующих одних и тех же ресурсов, имеют резервы времени их выполнения, предполагающие, что их осуществление может быть отложено на некоторое время без всякого влияния на общую продолжи-

тельность выполнения всего проекта в целом. Поэтому, распараллеливание работ приводит к сглаживанию потребности в ресурсах.

Второй принцип исходит из того, что продолжительность выполнения некоторых работ зависит от объема выделяемых для них ресурсов. Поэтому, если у таких работ имеются также и резервы времени, то можно безболезненно для проекта в целом пойти на снижение интенсивности выполнения этих работ, что приведет к сглаживанию потребности.

Однако, несмотря на простоту и понятность общих принципов, на которых строится сглаживание потребности проекта в ресурсах, расчетные алгоритмы оказываются очень и очень трудоемкими. Следует признать, что пока не разработано метода прямого поиска оптимального решения этой задачи, и на практике процедуры сглаживания связаны либо с полным перебором возможных вариантов топологии проектного плана (в этом случае оказывается возможным доказать оптимальность варианта плана), либо с применением некоторых эвристических правил выстраивания квазиоптимальной топологии (например, «наиболее короткая работа должна выполняться первой»). И в том, и в другом случае нельзя обойтись без специального программного обеспечения, не только из-за трудоемкости решения задачи, но из-за того, что при ее решении слишком высока вероятность допустить расчетную ошибку.

Рассмотрим следующий пример. Используем старые данные о проекте, но дополним их информацией о необходимых ресурсах и получим табл. 6.

Таблица 6

Сетевая модель проекта с данными о ресурсах

Номер работы	Срок выполнения	Каким работам предшествует	Ресурсы
1	2	2, 3	3
2	3	8	4
3	4	6, 7, 10	10
4	5	6, 7, 10	2
5	4	9	4
6	6	8	2
7	4	–	9
8	2	–	8
9	5	11	4
10	1	11	9
11	2	–	8

К анализу потребности в ресурсах приступают с построения графика Ганта проекта, на котором работы откладываются на временной шкале от ранних сроков начала их выполнения. Параллельно с графиком Ганта строится гистограмма изменения потребности во времени, ось абсцисс которой – это временная шкала выполнения проекта, а ось ординат – суммарная (по всем выполняемым в данный момент времени работам) потребность в ресурсах. Исходный график Ганта и гистограмма потребности в ресурсах представлены на схеме 7. На схе-

не выделен критический путь и обозначены резервы по времени для работ, не составляющих критический путь.

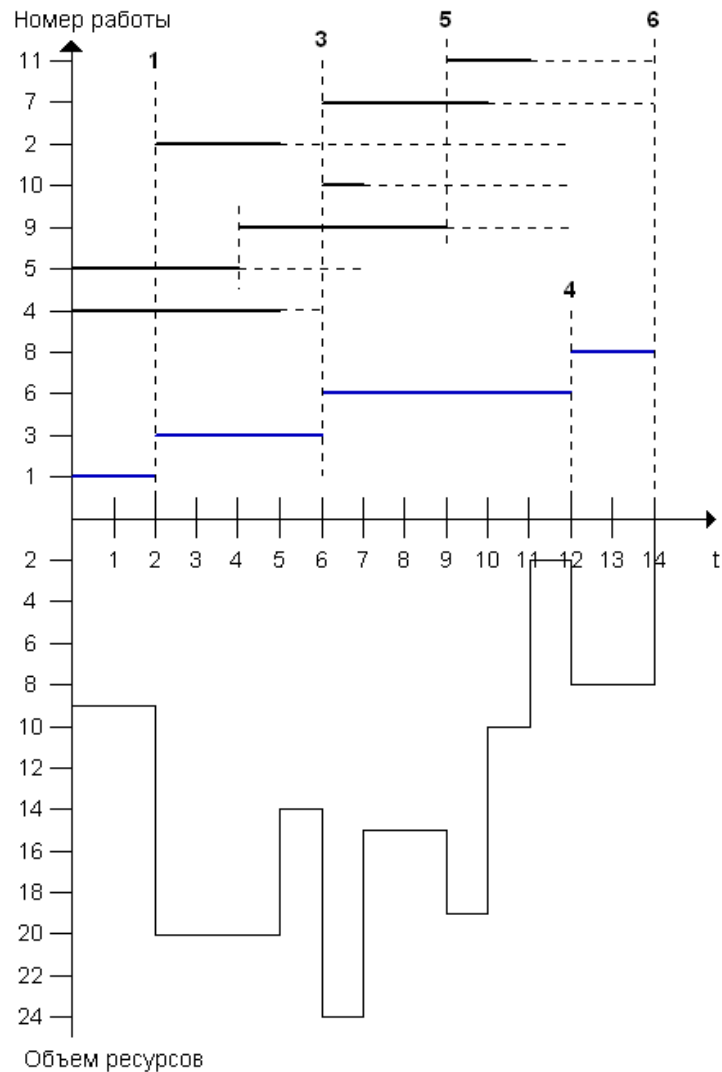


Рис. 12. Графика Ганта и гистограммы потребности в ресурсах

Простейшее выравнивание ресурсов в данном проекте можно осуществить вручную, сдвигая не критические работы в пределах их свободных резервов времени. Прodelайте данную работу самостоятельно.

Для выравнивания ресурсов не целесообразно использовать методы линейного программирования из-за большого количества уравнений и неизвестных в математической модели. Наибольшее распространение для решения такой задачи получили разнообразные эвристические методы из-за своей относительной простоты и вместе с тем неплохого качества получаемых решений (зачастую мало отличающихся от тех, которые можно было бы получить, применяя сложные методы оптимизации). Все эти методы основаны на принципе использования эвристик (определенных правил) перемещения ресурсов между работами и изменения календарных сроков выполнения работ. Один из алгоритмов, основанных на подобных эвристиках, приводится ниже.

Алгоритм приведения проекта в соответствие с ограничениями по одному ресурсу:

Шаг 1. Определяем список работ, которые могут начинаться в день D_i ($i=1, 2, 3, \dots, N$). Сначала рассматривается первый день. Переход к **Шагу 2**.

Шаг 2. Работы упорядочиваются в порядке возрастания их свободных резервов времени. Переход к **Шагу 3**.

Шаг 3. Из упорядоченного списка выбирается работа X и определяется, достаточно ли имеется ресурсов для начала ее выполнения в день D_i ? Если ДА, то переходим к **Шагу 4**. Если НЕТ, то переходим к **Шагу 9**.

Шаг 4. Начало выполнения работы X окончательно назначается на день D_i , а наличное количество ресурсов уменьшается на сумму ресурсов, требуемых для выполнения работы X . Переход к **Шагу 5**.

Шаг 5. Проверяется условие, все ли работы из списка тех, что могут начинаться в день D_i , рассмотрены? Если НЕТ, то переход к **Шагу 6**. Если ДА, то переход к **Шагу 7**.

Шаг 6. Рассмотренная и закрепленная только что за днем D_i работа X исключается из списка, переходим к **Шагу 3**.

Шаг 7. Проверяется условие, имеются ли еще работы в проекте, для которых не произведено окончательное закрепление сроков начала выполнения? Если ДА, то переход к **Шагу 8**. Если НЕТ, то переход к **Шагу 13**.

Шаг 8. Выбирается следующий день ($D_i = D_i + 1$) и переходим к **Шагу 1**.

Шаг 9. Проверяется условие: является ли работа X критической? Если ДА, то переход к **Шагу 11**. Если НЕТ, то переход к **Шагу 10**.

Шаг 10. Возможный срок начала работы откладывается на 1 день. Переход к **Шагу 5**.

Шаг 11. Проверяется условие, можно ли передать данной работе ресурсы с не критических работ, выполнение которых уже запланировано на этот день? Если НЕТ, то переход к **Шагу 10**. Если ДА, то переход к **Шагу 12**.

Шаг 12. Начало выполнения критической работы X окончательно назначается на день D_i , приводится в соответствие количество ресурсов на связанных работах, а наличное количество ресурсов уменьшается на сумму ресурсов, требуемых для выполнения работы X (за минусом того количество ресурсов, которое было перенесено с другой работы). Переход к **Шагу 5**.

Шаг 13. Алгоритм считается завершенным.

1.6. ОБЗОР ПРОГРАММНЫХ СРЕДСТВ ДЛЯ УПРАВЛЕНИЯ ПРОЕКТАМИ

В данном пункте представлен обзор некоторых программных решений для автоматизации деятельности по управлению проектами.

Перечислим основные задачи, для решения которых используются системы управления проектами (СУП):

- разработка календарного плана проекта без учета ограниченности ресурсов или с учетом такой ограниченности;

- определение критического пути и резервов времени исполнения операций проекта;
- определение потребности проекта в финансировании, материалах и оборудовании;
- определение распределения во времени загрузки возобновляемых ресурсов;
- анализ рисков и планирование с учетом рисков;
- учет исполнения проекта;
- анализ отклонений хода работ от запланированного и прогнозирование основных параметров проекта.

Как правило, СУП делятся на системы начального уровня, к которым, учитывая их функционал, наиболее применим термин «Системы календарного планирования и контроля» (СКПК) и профессиональные системы управления проектами. Хотя в последние годы отмечается устойчивая тенденция «подростания» систем начального уровня к профессиональным пакетам и еще более активное расширение функциональности последних, цены на системы из разных групп могут заметно различаться. Если СКПК попадают в диапазон \$200-800, то профессиональные СУП могут стоить заметно больше \$5000.

В настоящее время существует несколько сотен систем, так или иначе, реализующих функции СКПК. Реально, на рынке СНГ стабильно присутствует не более 10 систем. Среди них есть и российские разработки.

1.6.1. Системы начального уровня.

Принципиальных функциональных отличий между СКПК начального уровня на самом деле не так много. Практически все они имеют сходный набор функций. Перечислим основной, де-факто, стандартный их набор:

- Поддержка расписания из неограниченного количества задач с учетом приоритетов задач, расчет критического пути, вычисление резервов времени (длительность в часах, днях, неделях или комбинированная);
- Умение работать с пользовательскими календарями для задач и ресурсов;
- Поддержка всех видов связей и типов работ, различных типов ресурсов (материальные, трудовые);
- Способность работать с иерархической структурой работ (WBS – Work Breakdown Structure);
- Возможность выполнения выборки, сортировки, группировки, суммирования, по кодам WBS и ID работ;
- Поддержка основных видов визуального представления (диаграмма Ганта, PERT-диаграмма, таблица работ/ресурсов, таблица связей, гистограммы ресурсов).

1. MS Project 2002 (разработчик – Microsoft).

Фактически MS Project 2002 – собирательное название для четырех относительно самостоятельных продуктов:

Microsoft Project Standard – инструмент для работы менеджера проекта, являющийся дальнейшим развитием MS Project 2000;

Microsoft Project Professional – более продвинутая редакция предыдущего пакета, обладающая целым рядом функций, необходимых для крупных корпоративных клиентов;

Microsoft Project Server – серверный продукт, который создан на базе Microsoft Project Central 2000. Он предназначен для поддержки коллективного сотрудничества в рамках проекта. Фактически представляет собой системную платформу для организации коммуникаций в рабочей группе;

Microsoft Project Web Access – средство удаленного доступа к серверу с Microsoft Project Server через Internet или корпоративную сеть intranet. Для работы с ним требуется наличие клиентской лицензии на доступ к Microsoft Project Server. С его помощью составляется отчетность, и осуществляются постановка задач исполнителям, а также поддержка управления корпоративными ресурсами и портфелями проектов.

2. SureTrak Project Manager (разработчик – Primavera inc.).

Являясь младшим (и самым дешевым – стоимость в России за 5 лет осталась неизменной: \$700) продуктом в семействе Primavera, SureTrak Project Manager позиционируется как продукт начального уровня для управления несложными проектами в небольших компаниях. Интерфейс – вполне стандартный. Очень хорошо реализован принцип WYSIWYG и масштабирование временной оси при отображении диаграммы Ганта. Продукт поддерживает MAPI, т.е. может работать совместно с системами электронной почты (умеет с их помощью отправлять данные проектов).

Для активного продвижения на рынке СНГ этот пакет был полностью локализован. В российском варианте поставки русскоязычный интерфейс, система помощи и руководство пользователя. Из особенностей можно отметить удобную функцию «луч» (Progress Spotlight). При выделении на временной оси (диаграмме Ганта) временного промежутка, в таблице работ выделяются цветом операции, выполнение которых запланировано в этот временной интервал. SureTrak имеет как собственный формат данных, так и без каких либо дополнительных настроек «понимает» формат программы P3. В настоящее время продукт представлен версией 3.0.

1.6.2. Профессиональные системы

В отличие от СКПК, профессиональные системы управления проектами в своей функциональности уже заметно отличаются друг от друга. Как правило, это уже не отдельные программы, а комплексы, в состав которых входят различные модули, предназначенные для решения специфических задач.

1. Primavera Project Planner (разработчик – Primavera inc.).

Для построения интегрированной системы управления проектами компания **Primavera inc.** предлагает несколько продуктов. Для использования на нижних уровнях управления – уже упоминавшийся SureTrak Project Manager,

профессиональный пакет управления проектами Primavera Project Planner (P3) для работы со сложными многоуровневыми иерархическими проектами и систему масштаба предприятия, работающую по технологии клиент/сервер **Primavera Project Planner for the Enterprise (P3e)**.

В качестве системы управления контактами, предлагается полностью локализованный продукт **Expedition**; обеспечивать доступ к проектной информации, используя Интернет, призван **Webster for Primavera**. Такое разнообразие может сбить с толку, поэтому мы рассмотрим Primavera Project Planner (P3) как продукт, наиболее близкий к теме данного обзора.

Интерфейс системы – стандартный, оконный. В поставке – несколько десятков шаблонов представления проекта (в документации – *макетов* (layout)), пользователю предоставляется возможность создавать и сохранять собственные макеты. Поставляемый в составе пакета генератор отчетов Report Smith позволяет создавать любые табличные и графические отчетные формы. Возможна иерархическая организация проекта по произвольной комбинации кодов. Радует отличная реализация принципа WYSIWYG при выводе отчетов на печать.

Для моделирования проекта доступен обширный набор инструментов, включающий в себя до 20 уровней WBS и 16 пользовательских полей данных. Реализованы 9 типов работ (задача, веха, «гамак», встреча и др.), все типы зависимостей между работами; 10 типов ограничений. Текущее расписание проекта может сравниваться с неограниченным числом базовых планов.

Развита функция *глобальной замены* для внесения изменений в данные проекта с применением логических, арифметических и строковых выражений.

Для управления ресурсами и стоимостями доступны все, стандартные для такого класса продуктов, инструменты. Стоимости ресурсов во времени, а также их пределы потребления могут быть различными. Особенно интересной является возможность создавать собственные профили использования ресурсов в дополнение к 10 существующим.

Структура статей затрат может поддерживать неограниченное количество счетов с 12 разрядным кодом.

В пакете реализован анализ отклонений хода работ от запланированного *Методом освоенного объема* (Cost/Schedule Control System Criteria – C/SCSC) и прогнозирование основных параметров проекта. В качестве средства анализа рисков предлагается продукт **Monte Carlo**. Он позволяет оценить вероятность выполнения проекта в заданные сроки в пределах бюджета.

В Primavera Project Planner имеется экспорт данных в форматы dBase и Lotus. Для двустороннего обмена данными с удаленными пользователями предназначена функция Primavera Post Office.

2. Open Plan (разработчик – Welcom Software Technology).

Этот продукт позиционируется как профессиональная система управления проектами масштаба предприятия. Выпускается в трех версиях: Enterprise, Professional и Desktop.

Интерфейс продукта весьма оригинален. Рабочее пространство представлено в виде нескольких рабочих столов, на которых помещаются ярлыки к стандартным объектам (файлы проектов, календарей, ресурсов, кодов, шаблонов), так и к любому файлу. При открытии проекта открывается «записная книжка проекта» – набор рабочих столов с ярлыками к файлам, непосредственно относящимся к проекту. В поставку входит несколько десятков наиболее распространенных шаблонов представления проекта. Применение шаблона к проекту осуществляется простым перетаскиванием нужного ярлыка на записную книжку проекта. Отдельного упоминания заслуживает функция «Директор Управления Проектами» (ДУП). ДУП – это инструментарий автоматизации повторяющихся процессов при управлении проектами. Объектами ДУП могут быть не только стандартные формы, представления и процедуры Open Plan, но и объекты из других приложений, например, текстового редактора, электронных таблиц, САД. В поставке – 35 шаблонов ДУП, разбитых, согласно рекомендациям PMI (www.pmi.org) на 8 категорий. Естественно, есть функция создания и сохранения пользовательских шаблонов представления и шаблонов ДУП.

В продукте весьма развита система ресурсного планирования. Реализовано два базовых метода расчета расписания:

- Ресурсное планирование при ограниченном времени – приоритетной является необходимость придерживаться общей даты завершения проекта при попытке минимизировать степень перегрузки ресурсов. В результате ресурсы могут быть перегружены.

- Ресурсное планирование при ограниченных ресурсах – приоритет отдается предотвращению перегрузки ресурсов, даже если это приведет к выходу проекта за рамки расписания. При этом строки сдвигаются на столько, на сколько это необходимо для полного избегания переагрузки ресурсов.

Реализован тип материальных ресурсов с ограниченным сроком хранения. При назначении исполнителей на операции можно указывать требуемую квалификацию или альтернативный ресурс и тогда, при ресурсном планировании, система предложит оптимальный, с точки зрения загрузки, ресурс. Благодаря иерархической организации ресурсов, можно создавать любые структуры статей затрат.

Следует особо отметить, что функция анализа рисков встроена в систему, тогда как в некоторых продуктах она поставляется как отдельный модуль. Для длительности избранных или всех работ проекта вводятся оптимистическая и пессимистическая оценки. Далее по методу Монте-Карло определяется вклад вероятностей в даты проекта.

Возможности сортировки, фильтрации, создания пользовательских полей и глобальной замены традиционно сильны для продуктов такого класса. Можно пользоваться стандартным набором или создать собственные.

В состав продукта входит модуль Web Publisher, с помощью которого осуществляется публикация данных проекта на веб-сервере. Этот модуль хотя и делает свое дело, но его реализация далеко не идеальна.

В качестве системы управления бюджетом проектов Welcom Software Technology предлагает продукт Cobra. Совместное использование Cobra с Open Plan или с другой СУП позволяет построить интегрированную систему управления календарным графиком и затратами проекта.

Стоимость Open Plan Professional около \$ 6000, версии Desktop ~ \$1000 (могут меняться в зависимости от комплекта поставки).

3. Spider Project (разработчик/представитель в России – компания «Технологии управления “Спайдер”», www.spiderproject.ru).

Без преувеличения можно сказать, что Spider Project лучшая российская система управления проектами. Версия под DOS появилась еще в 1992 году. От версии к версии заметно улучшается не только интерфейс системы, но и ее функциональность. Текущей является версия 9.07 (вышла 9 сентября 2005).

У этого продукта много отличий от западных собратьев, но основным из них является подход к определению длительности операций. В большинстве известных пакетов операции характеризуются длительностью их исполнения. В Spider Project наряду с длительностями можно задавать физические объемы работ на операциях. Длительность определяется пакетом в процессе составления расписания работ в зависимости от производительности назначенных ресурсов. В связи с этим, имеется отличие и в определении задержек на связях операций. Наряду с положительными и отрицательными временными задержками, реализованные во всех пакетах, можно использовать и объемные задержки. Дело в том, что с временными задержками может возникнуть ситуация, когда работа началась, но исполняется медленнее, чем было запланировано и временная задержка может исчерпаться раньше, чем будет выполнен запланированный объем работ.

Пакет позволяет использовать неограниченное количество составляющих стоимости, причем в разных валютах. Так же можно создать неограниченное количество различных иерархических структур работ и ресурсов.

Для анализа исполнения проекта, а также для анализа «что если» очень важно иметь возможность сохранять прежние версии проекта и иметь возможности для сравнения и анализа отклонений текущей версии проекта от предыдущих версий. В Spider Project есть возможность хранить неограниченное количество версий проекта и анализировать ход исполнения работ не только по сравнению с какой-то базовой версией, но и с любой другой.

Расчет расписания проекта методом критического пути производится без учета ограничения по ресурсам и имеет точное математическое решение. Если же при расчетах учитывается ограниченность ресурсов, то понятие резервов, в том числе и полного резерва (total float) теряет смысл. В Spider Project вычисляется ресурсный критический путь и резервы сроков исполнения операций с учетом ограниченности ресурсов.

Алгоритм анализа рисков так же отличается от реализованного в других системах. При моделировании рисков в качестве исходной информации ис-

пользуются не оценки длительности (оптимистические, пессимистические), а оценки производительности ресурсов.

Непривычно, но достаточно продумано, реализована поддержка групповой работы над проектом. В Spider Project нет одновременного доступа на изменение данных. Ответственный за свою часть проекта (фазу) представляет менеджеру проекта свои файлы. И решение принять или отвергнуть изменения остается за менеджером проекта. Именно такое решение, по мнению разработчиков, позволяет избежать неразберихи при изменении проектных данных. С этих позиций разработана и система групповой работы через Интернет.

Так же следует отметить хорошую справочную систему продукта, в которую, помимо руководства пользователя включен переработанный русский перевод книги Project Management Body of Knowledge.

1.7. БАЗОВЫЕ АСПЕКТЫ РАБОТЫ И НАСТРОЙКИ MS PROJECT.

В этом и нескольких последующих параграфах представлено описание функций и возможностей программы MS Project.

Интерфейс программы Microsoft Project является стандартным для офисных программ от данного производителя.

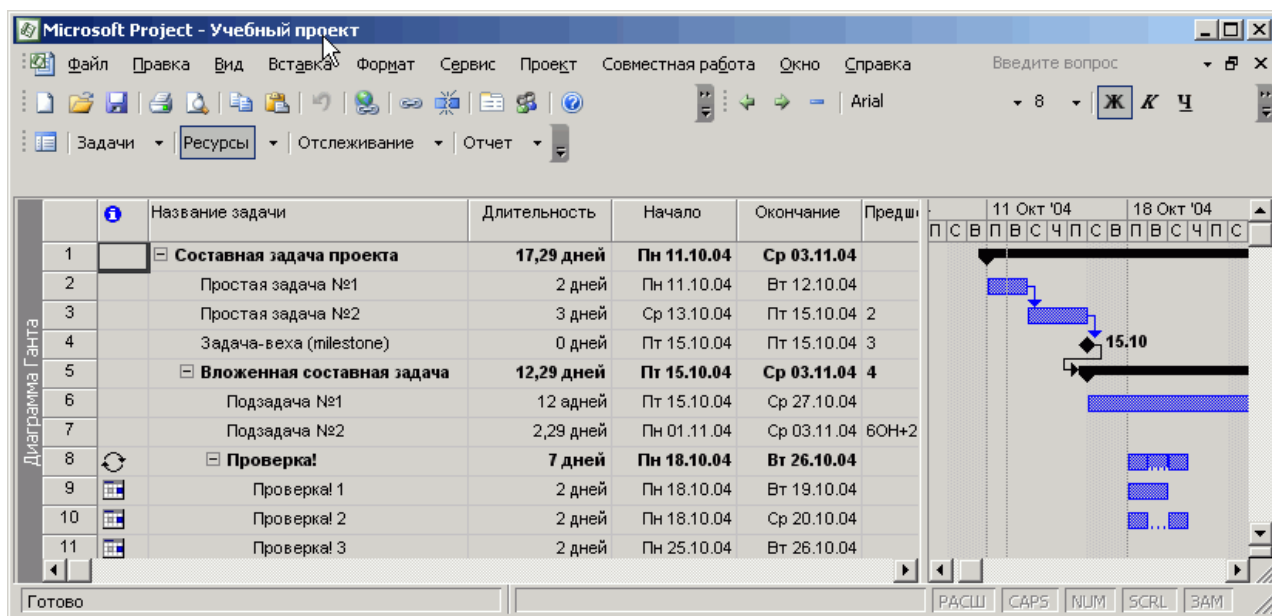


Рис. 13. Интерфейс программы Microsoft Project

Основным средством для отображения и работы с информацией в MS Project является *вид*, иначе называемый *представлением* (view). Вид отображает в удобной для пользователя форме данные из внутренних таблиц программы, описывающих проект. Компонентами вида могут являться *таблица*, *диаграмма* и *форма*. Программа MS Project содержит несколько predefined видов. Приведем названия и краткую характеристику некоторых из них:

1. *Диаграмма Ганта*. Данный вид представляет собой комбинацию таблицы с информацией о задачах проекта и диаграммой Ганта для проекта.

2. *Использование задач.* Вид состоит из таблицы с задачами проекта и информацией о трудозатратах для каждой задачи.
3. *Календарь.* Представляет план проекта как календарь органайзера, где отмечены выполняемые в определенный день задачи.
4. *Сетевой график.* Данный вид является сетевой диаграммой типа *activities on nodes* и позволяет наглядно представить состав и связь между работами.
5. *График ресурсов.* Показывает в виде диаграммы загрузку ресурсов по датам.
6. *Лист ресурсов.* Представляет собой таблицу для ввода информации о ресурсах, содержащихся в проекте.

Пользователь выбирает текущий отображаемый вид при помощи пункта главного меню Вид. Пользователь имеет возможность создать собственный вид, указав соответствующую таблицу, диаграмму, форму. Любой вид и его компоненты допускают разнообразную настройку. Так, например, можно изменить состав колонок, входящих в таблицу вида, настроить параметры графического отображения данных в диаграмме и т. п.

Для изменения настроек программной среды следует воспользоваться командами меню Сервис | Параметры...

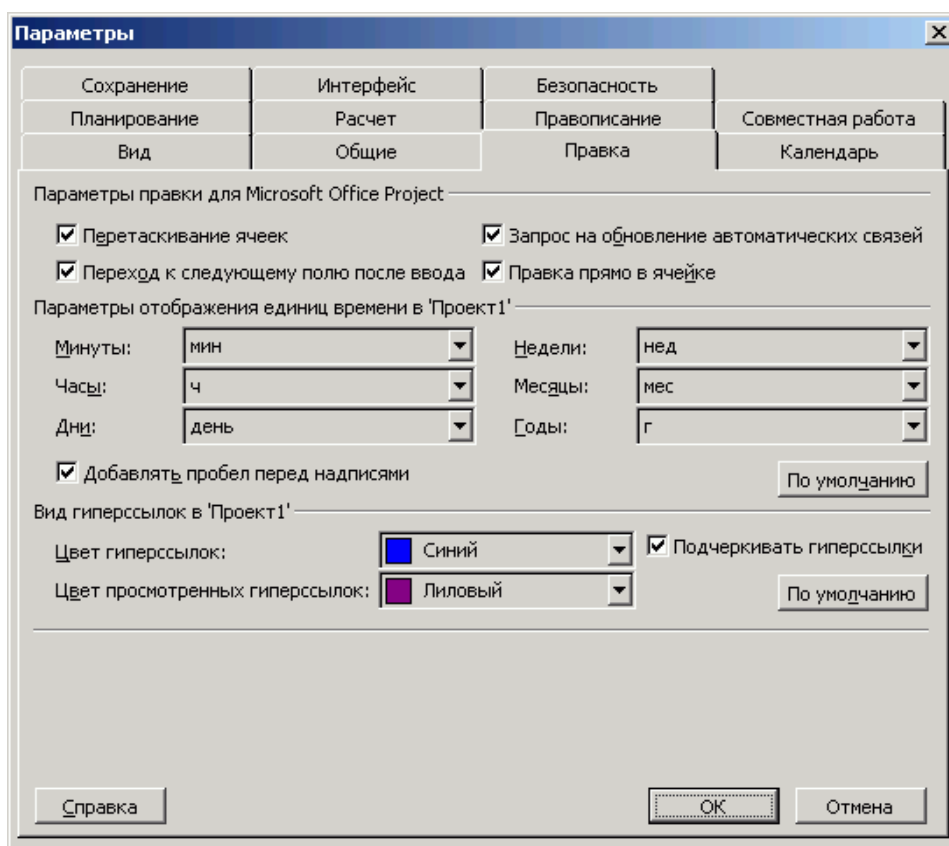


Рис. 14. Параметры программной среды

Настройка календарей проекта производится при помощи команд Сервис | Изменить рабочее время.... Существует возможность как отредактировать

один из трех базовых календарей (Стандартный, 24 часа, Ночная смена), так и создать новый календарь.

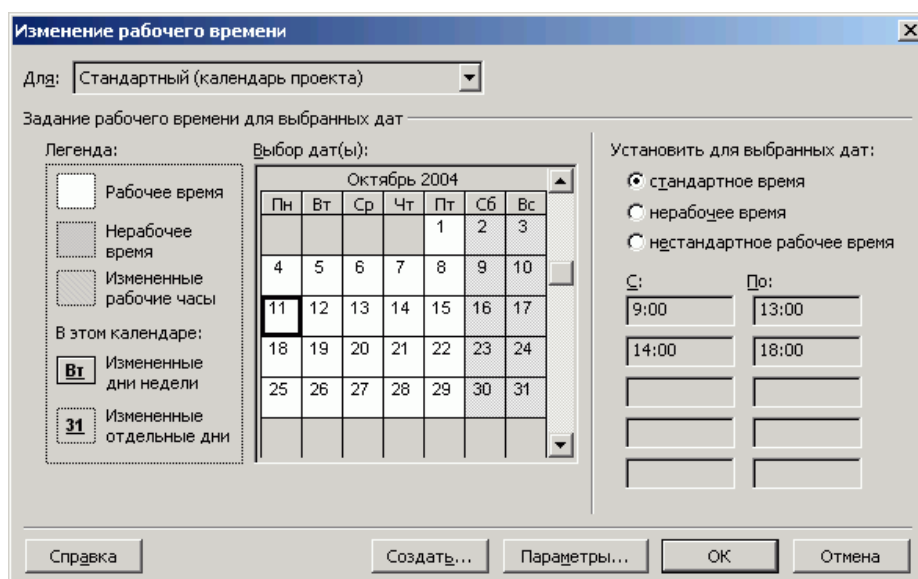


Рис. 15. Изменение календаря проекта

Рекомендуется проводить разработку плана проекта по следующей схеме:

1. Выполнить первоначальную настройку программной среды, включая настройку календарей.
2. Начать проект. Выполнить настройку свойств проекта.
3. Определить состав задач проекта и их взаимосвязей. Выполнить настройку параметров отдельных задач.
4. Произвести ресурсное планирование: определить состав ресурсов и их назначений задачам, устранить перегрузку ресурсов.
5. Произвести стоимостный анализ проекта.

Данная схема позволяет создать *базовый план* проекта. В дальнейшем на основании этого базового плана производится *отслеживание хода выполнения* проекта. В схеме не выделены такие задачи, как составление отчетов по проекту, экспорт плана проекта в другие программные среды, публикация плана проекта в Интернет.

Когда начинается новый проект, желательно произвести его настройку при помощи диалогового окна «Сведения о проекте» (Проект | Сведения о проекте...).

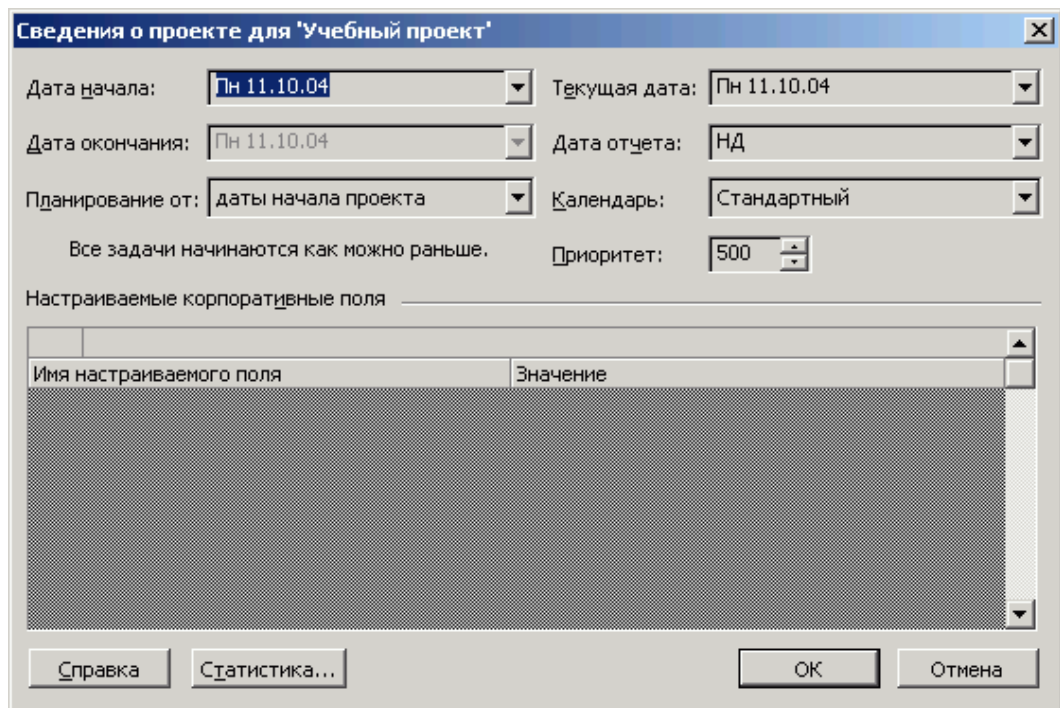


Рис. 16. Сведения о проекте

Основными параметрами проекта являются следующие:

1. Методика планирования проекта (Планирование от:). При **планировании от начала** известна начальная дата проекта, а завершающая дата проекта вычисляется. Все задачи проекта начинаются по умолчанию как можно раньше. При **планировании от конца** известна дата окончания проекта. Все задачи проекта начинаются по умолчанию как можно позже.

2. Календарь проекта – выбирается из списка. Это может быть как один из трех базовых календарей, так и собственный календарь пользователя, созданный ранее. Календарь проекта служит основой для календарей задач и ресурсов проекта.

3. Приоритет проекта (число от 1 до 1000) имеет смысл, только если одновременно ведется работа с несколькими проектами. Приоритет позволяет производить фильтрацию проектов в видах (более важные – менее важные).

1.8. ВИДЫ ЗАДАЧ. ХАРАКТЕРИСТИКИ ОТДЕЛЬНОЙ ЗАДАЧИ

В MS Project существует несколько основных видов задач. *Простая задача* – вид задачи по умолчанию. *Составная задача* – это задача, объединяющая в себе несколько простых или составных задач. Рекомендуется создать специальную составную задачу, объединяющую все задачи проекта. Составная задача отображается особым образом на диаграмме Ганта (рис. 17).

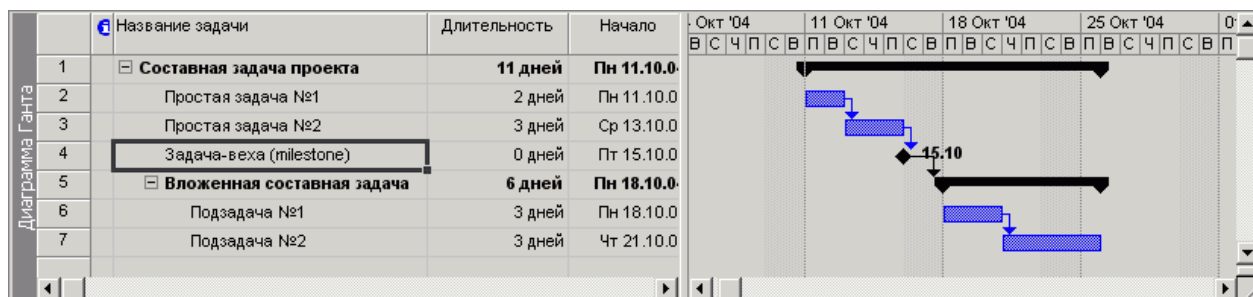



Рис. 17. Составная задача проекта

Для получения составной задачи из нескольких простых достаточно переместить простые задачи на больший уровень вложенности при помощи кнопок на панели инструментов  или использовать сочетания клавиш Alt+Shift+Left, Alt+Shift+Right.

Задача-веха (milestone) служит для выделения окончания произвольной фазы проекта. Для того чтобы пометить задачу как веху, при вводе задачи достаточно указать ее длительность равной нулю. Если требуется сделать вехой задачу с ненулевой длительностью, используется специальный переключатель в диалоговом окне свойств задачи (см. ниже).

MS Project позволяет вводить в проект *повторяющиеся задачи* (при помощи команд Вставка | Повторяющаяся задача...). При этом появляется диалоговое окно, которое позволяет настроить параметры повторения (рис. 18).

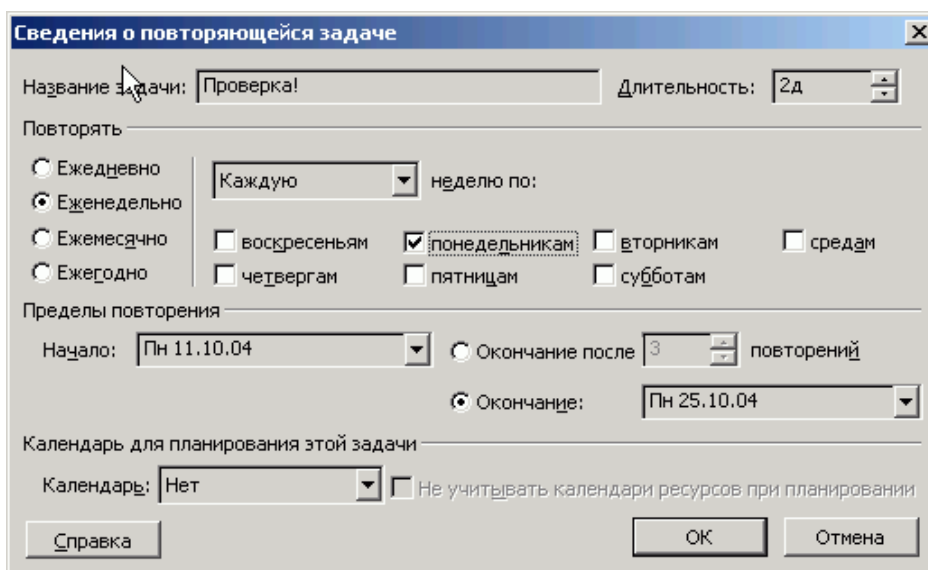


Рис. 18. Сведения о повторяющейся задаче

Повторяющаяся задача отображается как составная задача, состоящая из отдельных подзадач – своих повторений.

Простую задачу можно при необходимости отметить как *прерванную (отложенную) задачу*. Это соответствует ситуации, когда выполнение задачи отложили на некоторый период времени. При этом указанная длина задачи не меняется. Для задания прерванной задачи следует воспользоваться контекстным

меню задачи на диаграмме Ганта. Помечается точка разрыва, и величина разрыва задается простым растягиванием (рис. 19).

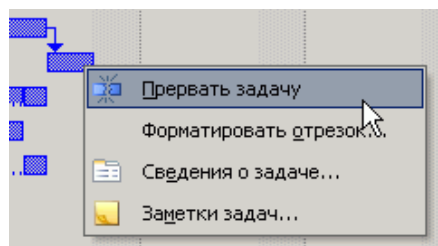


Рис. 19. Прерывание задачи

Рассмотрим отдельные параметры задачи. Для настройки параметров служит диалоговое окно Сведения о задаче (Проект | Сведения о задаче или Shift+F2).

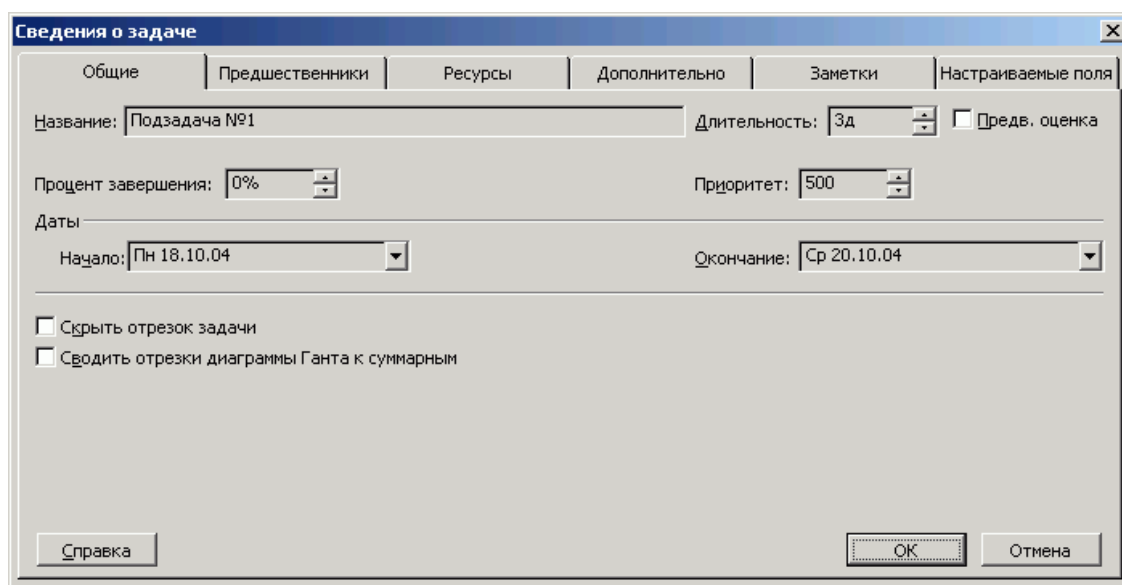


Рис. 20. Общие сведения о задаче

Любая задача имеет название и длительность. Длительность задачи может измеряться в минутах, часах, днях (стандартное значение), неделях, месяцах и годах. Для некоторых задач следует использовать *астрономическую длительность*, то есть считать длительность задачи независимо от выходных дней в календаре. Такие задачи помечаются символом «а» при вводе единицы длительности. Если оценка продолжительности задачи *предварительная*, может быть отмечен соответствующий флажок в диалоговом окне или введен знак вопроса после длительности задачи. Например, ввод длительности задачи в виде 20ад? означает предварительную оценку длительности задачи в 20 астрономических дней.

Процент завершения задачи вводится при отслеживании хода выполнения проекта и отображается в наглядной форме на диаграмме Ганта. При начальном вводе задачи редактировать это значение не нужно.

Приоритет задачи (от 1 до 1000) позволяет выделить наиболее важные задачи. Приоритет учитывается при выравнивании загрузки ресурсов, также возможно фильтровать задачи в видах по приоритетам.

Обратите внимание на параметры даты начала и окончания задачи. Не следует вводить их вручную, так как это изменяет ограничения задачи (см. ниже). Эти даты обычно рассчитываются автоматически на основании других параметров задачи и проекта.

При помощи таблицы в разделе Предшественники задаются связи между задачами проекта. В таблице можно указать название задачи-предшественника, тип связи и величину запаздывания.

Ид.	Название задачи	Тип	Запаздывание
6	Подзадача №1	Окончание-начало (ОН)	20%

Рис. 21. Ввод предшественников задачи

Допускаются следующие типы связей между задачами:

1. **Окончание-начало (ОН)**. Задача не может быть начата до тех пор, пока не завершатся все предшествующие задачи. Данный тип связи устанавливается по умолчанию.
2. **Начало-начало (НН)**. Задача не может быть начата до тех пор, пока не начнутся все предшествующие задачи.
3. **Окончание-окончание (ОО)**. Задача не может быть завершена до тех пор, пока не завершатся все предшествующие задачи.
4. **Начало-окончание (НО)**. Задача не может быть завершена, пока не начнутся все предшествующие задачи.

Величина запаздывания служит для более тонкой настройки связи между задачами. Она может быть как положительной, так и отрицательной (в этом случае идет речь об опережении). Величина запаздывания может измеряться как в абсолютных единицах (днях, неделях), так и в процентах длительности задачи-предшественницы.

Отметим, что редактирование связи между работами удобно выполнять при помощи вида Сетевой график. На сетевом графике достаточно выделить две задачи и выполнить команду Правка | Связать задачи (Ctrl+F2). Щелчок

на связи между задачами вызывает диалоговое окно Зависимость задач, которое позволяет изменить тип связи и величину запаздывания.

Рассмотрим параметры задачи, связанные с ограничением ее времени выполнения (рис. 22).

Рис. 22. Параметры, связанные с ограничением времени выполнения

Параметр **Крайний срок** устанавливает соответствующую дату. Фактически, крайний срок не является ограничением времени выполнения задачи. Дата крайнего срока отображается на диаграммах, а если задача продолжается дольше, чем ее крайний срок, то она надлежащим образом маркируется (рис. 23).

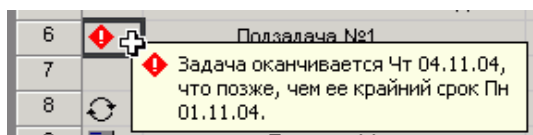


Рис. 23. Маркировка задачи

Тип ограничения задает одно из восьми возможных условий планирования задачи и представлен в табл. 7. При этом при необходимости вводятся соответствующая дата.

Таблица 7

Тип ограничения задачи

Тип ограничения	Жесткость	Пояснения
<i>Как можно позже</i>	Гибкое	Задача начинается как можно позже. Данное ограничение используется по умолчанию при планировании проекта от даты завершения.
<i>Как можно раньше</i>	Гибкое	Задача начинается как можно раньше. Данное ограничение используется по умолчанию при планировании проекта от даты начала.
<i>Начало не позднее</i>	Умеренно жесткое	Наиболее поздняя возможная дата начала задачи задается пользователем.

<i>Начало не ранее</i>	Умеренно жесткое	Наиболее ранняя возможная дата начала задачи задается пользователем.
<i>Окончание не позднее</i>	Умеренно жесткое	Наиболее поздняя возможная дата окончания задачи задается пользователем.
<i>Окончание не ранее</i>	Умеренно жесткое	Наиболее ранняя возможная дата окончания задачи задается пользователем.
<i>Фиксированное начало</i>	Жесткое	Указывается точная дата начала задачи.
<i>Фиксированное окончание</i>	Жесткое	Указывается точная дата окончания задачи.

Параметры Тип задачи и Фиксированный объем работ будут рассмотрены при описании ресурсного планирования.

Параметр Код СДР определяет код структурной декомпозиции работ, который требуется применить к задаче. В данном поле содержится алфавитно-цифровой код, который может быть использован для представления положения задачи в иерархической структуре проекта. Коды СДР обычно являются уникальными; иными словами, каждая задача имеет свой код СДР. Код СДР по умолчанию – номер задачи в структуре проекта. Для настраиваемого кода СДР можно ввести или определить формат при помощи команды Проект | СДР | Определить код...

При помощи флажка Пометить задачу как веху можно определить вехи с ненулевой длительностью.

Закладка Ресурсы позволяет назначить на выполнение задачи определенный состав ресурсов. Можно вводить названия ресурсов и величину их использования в строки таблицы (планирование ресурсов «от задач»), однако обычно вначале определяется общий состав ресурсов проекта, затем производится назначение.

На закладке Заметки можно ввести текстовые комментарии для задачи. Поле для комментариев допускает внедрение различных объектов (рисунки, файлы).

Закладка Настраиваемые поля позволяет редактировать величину настраиваемых полей для задачи. Под *настраиваемыми полями* понимаются специальные поля, изначально отсутствующие среди характеристик задачи (или ресурса), но определенные пользователем. Для создания настраиваемых полей проекта служит диалоговое окно Настройка полей (Сервис | Настройка | Поля).

1.9. РЕСУРСНОЕ ПЛАНИРОВАНИЕ В MS PROJECT

Следующим этапом планирования после ввода состава работ, настройки их параметров и установления связей между работами является ресурсное планирование. Для ресурсного планирования в MS Project возможно использование двух подходов: планирование «от ресурсов» и планирование «от задач».

Планирование «от ресурсов» используется в том случае, если заранее известна полная информация об объеме ресурса и его доступности (график рабочего времени ресурса). Для планирования от ресурсов обычно используется вид

Лист ресурсов. Данный вид представляет собой таблицу со следующими колонками:

1. Название ресурса.
2. Тип. Различают два типа ресурсов: *трудовые ресурсы* (work resources) и *материальные ресурсы* (material resources). Трудовые ресурсы являются возобновляемыми – после окончания одной задачи их можно использовать на другой задаче (рабочие и т. п.). Материальные ресурсы можно использовать только однократно (бумага, сырье).
3. Единицы измерения материалов – справочное поле, используемое для оформления информации. Его можно заполнить только для материальных ресурсов.
4. Краткое название – справочное поле, используемое для оформления информации.
5. Группа – название группы, которой принадлежит ресурс. Поле заполняется при необходимости отобразить, отсортировать, отфильтровать или отредактировать ресурсы по группам ресурсов.
6. Макс. единиц – данное поле имеет смысл только для трудовых ресурсов и позволяет определить максимальное количество ресурса, которое может быть назначено работам проекта. Величина в поле измеряется в процентах. Запись в данном поле значения 200% для ресурса *программист* означает, что в нашем распоряжении имеются два программиста.
7. Стандартная ставка. Данное поле определяет стоимость использования трудового ресурса в единицу времени (по умолчанию – в час) при обычных несверхурочных трудовых затратах. Для материальных ресурсов значение в этом поле показывает цену единицы.
8. Ставка сверхурочных. Поле задает стоимость использования трудового ресурса в единицу времени при сверхурочных трудовых затратах (при превышении максимальной доступности). Поле допускает ввод значений только для трудовых ресурсов.
9. Затраты на использование. Значение этого поля – стоимость, начисляемая каждый раз при использовании ресурса. Затраты добавляются при назначении задаче единицы трудового ресурса. Значение поля не меняется со временем использования ресурса. Для материального ресурса в данное поле вводятся затраты, начисляемые один раз независимо от числа единиц.
10. Начисление. Поле содержит способы и время оценки стандартных и сверхурочных затрат ресурса к затратам задачи. Имеются следующие варианты: В начале, По окончании, Пропорциональное (по умолчанию). При выборе для поля значения «В начале» затраты начисляются сразу после начала задачи, которое определяется датой в поле «Фактическое начало». Если установлено значение «По окончании», затраты не начисляются до тех пор, пока значение оставшихся трудовых затрат не равно нулю. При указании значения «Пропорциональное» затраты начисляются в процессе выполнения трудовых затрат по календарному плану и при получении результатов о фактических трудовых затратах, а также вычисляются умножением единиц затрат на трудовые затраты.

11. Базовый календарь – календарь, используемый при планировании для ресурса (имеет смысл только для трудовых ресурсов).

12. Код – любой код, аббревиатура или число, которое необходимо ввести как часть сведений о ресурсе.

Кроме возможности вводить параметры ресурса в таблице вида *Лист ресурсов*, существует дополнительный способ уточнения параметров ресурса в диалоговом окне Сведения о ресурсе.

Доступен с	Доступен по	Единицы
НД	НД	250%

Рис. 24. Сведения о ресурсе

Если значение параметра Тип резервирования – Предложенный, то добавление данного ресурса в проект считается под вопросом. Если значение данного параметра Выделенный, то добавление данного ресурса считается решенным. Этот тип резервирования устанавливается по умолчанию.

Таблица Доступность ресурса используется для установки даты начала и даты окончания работы ресурса в проекте. Можно устанавливать различные уровни максимальной доступности ресурсов в разное время работы проекта. Эти значения вместе с календарем ресурса определяют возможные трудозатраты ресурса без превышения доступности.

Закладка Рабочее время позволяет отредактировать календарь ресурса. Закладка Затраты служит для ввода сведений о стоимости ресурса и подробнее будет рассмотрена позднее. Использование закладок Заметки и Настраиваемые поля для ресурса совпадает с аналогичными закладками для задачи.

После заполнения информации о ресурсах необходимо назначить ресурсы для каждой задачи. Это можно выполнить несколькими способами. Первый способ заключается в использовании закладки Ресурсы диалогового окна на-

стройки параметров отдельной задачи. Вторым способом – использовать вид *Использование задач*. Данный вид содержит две таблицы. В первой перечислены все задачи проекта и для каждой указаны назначенные на нее ресурсы. Во второй приведено распределение времени использования по дням для каждого ресурса. Наконец, можно использовать вид *Использование ресурсов*. Он похож на вид *Использование задач*, но отображает список ресурсов, для каждого из которых перечислены те работы, на которые он назначается.

Любая задача, участвующая в проекте, может быть охарактеризована тремя параметрами: длительность D , объем ресурсов U , необходимых для выполнения задачи, трудозатраты W . Данные параметры находятся в следующей взаимосвязи:

$$W = D * U.$$

Установив определенный Тип задачи, мы можем зафиксировать один из этих параметров. Тип задачи выбирается в диалоговом окне сведений о задаче. Таблица 8 показывает, как изменение одного из свойств задачи влияет на другие свойства в зависимости от ее типа.

Таблица 8

Зависимости длительности, объема ресурсов и трудозатрат

Тип задачи	Изменение объема ресурсов	Изменение длительности	Изменение трудозатрат
	приводит к пересчету		
Фиксированный объем ресурсов	длительности	трудозатрат	длительности
Фиксированные трудозатраты	длительности	объема ресурсов	длительности
Фиксированная длительность	трудозатрат	трудозатрат	объема ресурсов

В дополнение к указанию типа задачи можно использовать признак *Фиксированного объема работ*. Этот признак можно указать, если задача не относится к типу *Фиксированные трудозатраты*. Данный признак проявляется себя только в том случае, если после первоначального назначения некоторого ресурса на задачу производится переназначение данного ресурса. В случае фиксированного объема работ изменяется не общий объем работ по задаче, а степень загруженности ресурса. Например, пусть некоторая задача длится 3 дня и ее выполняет один программист. В этом случае объем работ равен $3 * 8 = 24$ человеко-часа, а ресурс программист загружен на 100%. Если добавить на данную задачу второго программиста, то увеличится объем работ до 48 человеко-часов. В случае фиксированного объема работ при добавлении второго программиста объем работ останется прежним, 24 часа, но каждый из программистов будет загружен на 50%.

Одной из задач, возникающих при ресурсном планировании, является задача устранения *перегрузки ресурсов*. Существует несколько причин появления перегрузки ресурсов:

1. Назначение задаче ресурса в количестве, превышающем его максимальный объем;
2. Одновременное назначение ресурса на несколько задач, из-за чего суммарный объем назначений превышает максимальный допустимый объем ресурса.
3. Назначение ресурса на задачи, выполняемые в то время, когда ресурс недоступен.

Факт перегрузки ресурса регистрируется MS Project автоматически. В таблице видов Использование задач или Использование ресурсов перегруженные ресурсы помечаются специальным значком. Для получения более подробной информации можно выбрать один из перегруженных ресурсов и вызвать вид График ресурсов. На графике выделено нормальное использование ресурса и перегрузка (Allocated и Over allocated).

Автоматическое выравнивание ресурсов осуществляется функцией Выравнивание загрузки ресурсов (Сервис | Выравнивание загрузки ресурсов...). Рассмотрим параметры настройки данной функции (рис. 25):

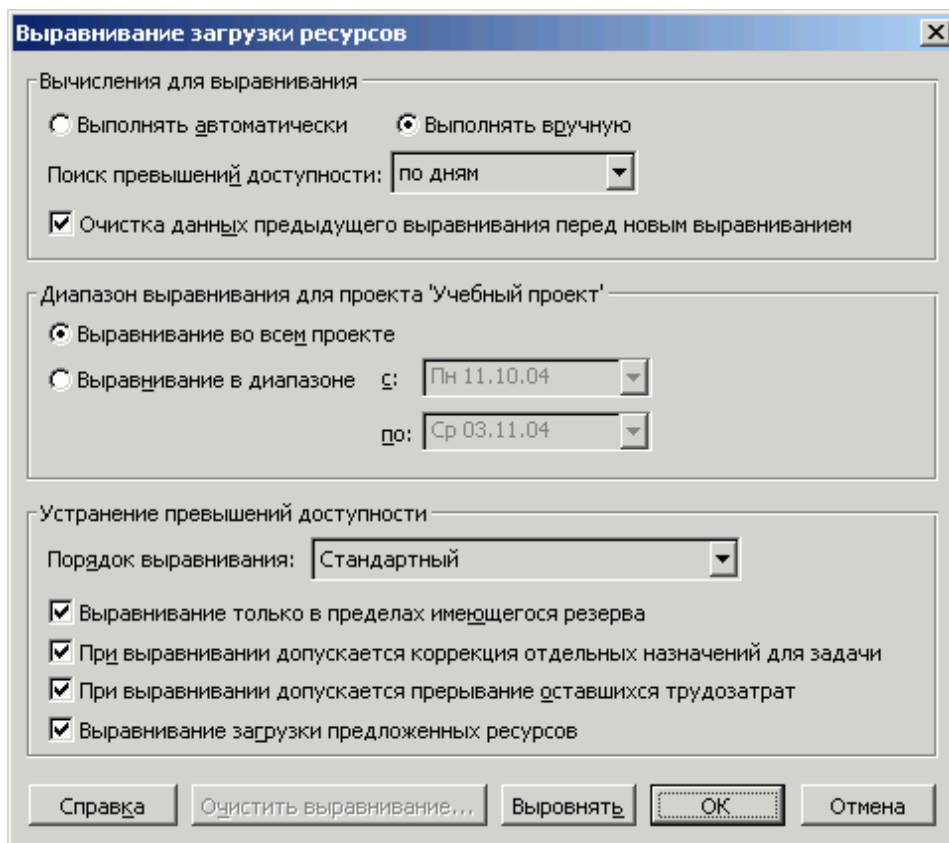


Рис. 25. Выравнивание загрузки ресурсов

Первая группа параметров позволяет задать автоматическое (при возникновении перегрузки любого ресурса) или ручное выполнение выравнивания ресурсов. В крупном проекте автоматическое выравнивание может замедлить ра-

боту с календарным планом. При выборе этого параметра необходимо снять флажок Очистка данных предыдущего выравнивания перед новым выравниванием. Этот флажок устанавливается по умолчанию, однако при автоматическом выравнивании очистка значений может существенно повлиять на календарный план.

Параметр Поиск превышений доступности определяет, при каком условии должно выполняться выравнивание загрузки. В зависимости от выбранного периода времени рассчитывается величина, при превышении которой для ресурса выполняется выравнивание. Например, если выбрано значение по дням, то перегруженными будут считаться ресурсы, превышение потребности в которых более 8 часов (при стандартном календаре ресурса).

В группе параметров Диапазон выравнивания задается, следует ли проводить выравнивание загрузки для всего проекта или только для задач, попадающих в заданный интервал времени.

Параметр Порядок выравнивания определяет, в каком порядке MS Project следует выполнять задержку или прерывание задач с превышением доступности. Возможны следующие значения параметра:

1. Только по идентификаторам – по мере необходимости применяется задержка задач с большими номерами идентификаторов, и лишь после этого анализируются другие условия.

2. Стандартный – при определении необходимости и способа выравнивания задач учитываются связи с задачами-предшественниками, временной резерв (задача, у которой больше общий временной резерв, будет отложена первой), даты (задача с более поздней датой начала будет отложена первой), приоритеты и ограничения. Этот параметр используется по умолчанию.

3. По приоритетам, стандартный – при определении необходимости и способа выравнивания задач в первую очередь учитываются приоритеты, а затем – связи с задачами-предшественниками, временной резерв, даты и ограничения.

При установлении флажка Выравнивание только в пределах имеющегося резерва предотвращаются задержки даты окончания проекта. Однако во многих проектах, где нет большого числа встроенных временных резервов, использование этого параметра может не привести к существенным изменениям в результате выравнивания.

Если установлен флажок При выравнивании допускается коррекция отдельных назначений для задачи, то разрешено при выравнивании изменять время, когда трудозатраты ресурса являются независимыми от трудозатрат других ресурсов на эту задачу. Это глобальный параметр для всех задач, и он установлен по умолчанию. Если необходимо разрешить выборочное выравнивание назначений для определенных задач, можно добавить в лист задач поле «Выравнивание назначений», а затем выбрать в нем значение «Да» или «Нет».

Если установлен флажок При выравнивании допускается прерывание оставшихся трудозатрат, то разрешено прерывание задач при выравнивании путем прерывания оставшихся трудозатрат по задачам или назначениям. Это

глобальный параметр для всех задач, и он установлен по умолчанию. Если необходимо разрешить выборочное прерывание оставшихся трудозатрат для определенных задач, можно добавить в лист задач поле «Допускается прерывание при выравнивании», а затем выбрать в нем значение «Да» или «Нет».

Включение флажка Выравнивание задач с предложенными ресурсами позволяет задействовать в процессе выравнивания задач, в которых наряду с подтвержденными ресурсами используются предложенные ресурсы.

При запуске операции выравнивания в MS Project последовательно проверяются все ресурсы. Если доступность ресурса превышена, выполняется поиск задач, вызывающих превышение доступности, а также выявление среди найденных задач таких, которые могут быть отложены. В MS Project не откладываются задачи, которые имеют:

- ограничение «Фиксированное начало» или «Фиксированное окончание»;
- ограничение «Как можно позже», если проект планируется от даты начала;
- ограничение «Как можно раньше», если проект планируется от даты окончания;
- приоритет 1000, означающий отказ от выравнивания;
- фактическую дату начала. Однако если установлен флажок При выравнивании допускается прерывание оставшихся трудозатрат, любые оставшиеся трудозатраты могут быть прерваны в результате выравнивания.

После определения задач, которые могут быть отложены, выбирается задача, которая будет отложена, на основе зависимостей задач, даты начала, приоритета и ограничений.

Чтобы увидеть, к каким изменениям задач привело выравнивание, в меню Вид выберите команду Другие представления. Просмотрите задачи в представлении Диаграмма Ганта с выравниванием, чтобы увидеть результаты выравнивания, а также задержки задач в результате выравнивания.

В MS Project существует механизм управления распределением ресурсов, что позволяет в некоторых случаях избавиться от перегрузки ресурсов. Этот механизм основан на понятии профиля использования ресурса. *Профиль использования ресурса (Work Contour)* – это график распределения ресурса внутри работы. Так как профиль обычно применяют к трудовым ресурсам, то можно считать, что это график распределения рабочего времени ресурса.

Существует восемь стандартных профилей, они представлены в табл. 9.

Таблица 9

Стандартные профили

Имя профиля	Пояснения
1. Плоский	Равномерное распределение единиц ресурса по всему сроку работы
2. Возрастающий	За первую половину длительности работы выполняется 25% объема
3. Убывающий	За первую половину длительности работы выполняется 75% объема

4. Двойной пик	За первую половину длительности работы выполняется 50% объема, но в 1-й и 2-й половине есть по пику нагрузки
5. Ранний пик	Первая половина – 70% объема, в первой половине один пик
6. Поздний пик	Вторая половина – 70% объема, во второй половине один пик
7. Пирамида	Плавное возрастание так, что к середине сделано 50% объема работы
8. Черепаха	Среднее между профилями 1 и 7

Для назначения профилей используется вид Использование задач или Использование ресурсов. Двойной щелчок в ячейке колонки Трудозатраты открывает диалоговое окно Сведения о назначении, в котором имеется параметр Профиль загрузки.

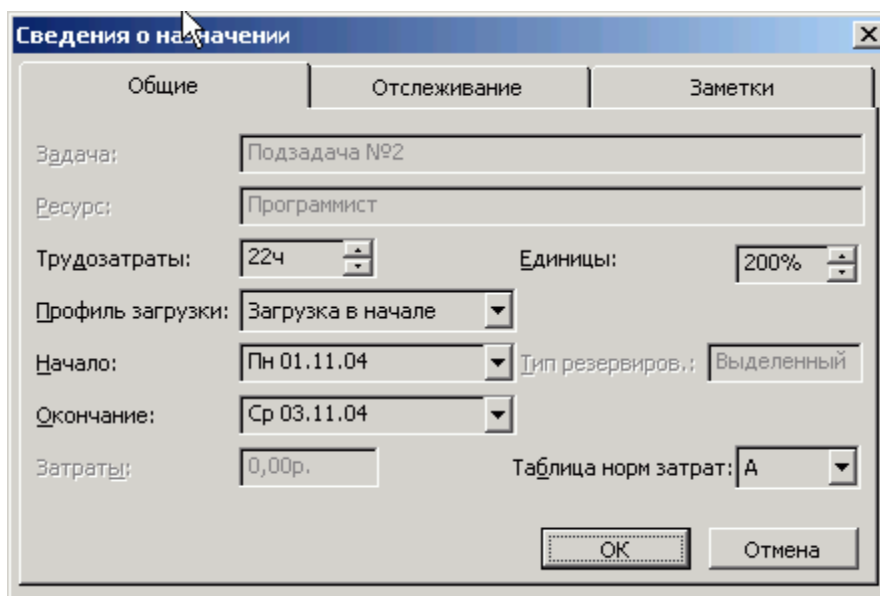


Рис. 26. Сведения о назначении – установка профиля

Существует возможность установить «пользовательский» профиль загрузки ресурсов. Для этого необходимо вручную изменить значения в правой таблице видов *Использование задач* или *Использование ресурсов*.

1.10. СТОИМОСТНЫЙ АНАЛИЗ ПРОЕКТА В MS PROJECT

Стоимость проекта рассчитывается на основании стоимости отдельных ресурсов. Для задания стоимости ресурса служит закладка Затраты диалогового окна Сведения о ресурсе.

Сведения о ресурсе

Общие | Рабочее время | Затраты | Заметки | Настраиваемые поля

Название ресурса: Программист

Таблицы норм затрат

Введите значение ставки или изменение в процентах относительно предыдущей ставки. Например, если затраты на использование ресурса сокращаются на 20%, введите -20%.

А (по умолчанию)	В	С	Д	Е
10,00р./ч				
Дата действия	Стандартная ставка	Ставка сверхурочных	Затраты на использование	
--	10,00р./ч	20,00р./ч	120,00р.	
Чт 14.10.04	0,00р./ч	0,00р./ч	0,00р.	

Начисление затрат: Пропорциональное

Справка | Подробности... | ОК | Отмена

Рис. 27. Ввод данных для расчета стоимости проекта

Используя таблицы норм затрат, можно назначать различные нормы выбранным ресурсам в разных таблицах. Каждая вкладка представляет собой таблицу, которую можно использовать для назначения различных норм затрат, соответствующих разным задачам. Таблицы норм затрат могут использоваться для увеличения или уменьшения норм по времени, например, в связи с ростом заработной платы или сокращением материальных ресурсов. По умолчанию ресурсы назначаются задачам с использованием норм затрат из таблицы А. Можно изменить назначенную таблицу в диалоговом окне Сведения о назначении.

Рассмотрим состав колонок таблицы норм затрат.

1. Дата действия. В эту дату начинается действие указанных в строке стандартной ставки, ставки сверхурочных работ и затрат на использование. Два тире (--), проставленные в поле Дата действия, означают, что соответствующая ставка действует в настоящий момент. Эта ставка применяется по умолчанию, когда нет другой даты действия, или используется для тех дат, которые не покрываются датами действия.

2. Стандартная ставка. Почасовая ставка, начисляемая для обычных трудозатрат или стандартного рабочего времени данного ресурса.

3. Ставка сверхурочных. Почасовая ставка, начисляемая для сверхурочных трудозатрат данного ресурса.

4. Затраты на использование. Начисляются, когда используется данный ресурс, не зависимо от величины трудозатрат.

Смысл параметра Начисление затрат был рассмотрен ранее (см. способ ввода ресурсов в *Лист ресурсов*).

После ввода данных о стоимости ресурсов MS Project автоматически выполняет необходимые вычисления. Данные вычислений отображаются в Таблице затрат (Вид | Таблица | Затраты).

1.11. АНАЛИЗ ПРОЕКТА ПО МЕТОДУ PERT

MS Project позволяет провести оценку длительности проекта по методу *PERT*. Напомним, что данный метод вычисляет продолжительность D отдельной задачи как взвешенное среднее между величинами O – *оптимистическая длительность* задачи, E – *ожидаемая длительность* задачи, P – *пессимистическая длительность* задачи по формуле

$$D = \frac{w_1 * O + w_2 * E + w_3 * P}{(w_1 + w_2 + w_3)}$$

Классический метод PERT предполагает, что $w_1=1$, $w_2=4$, $w_3=1$.

Для ввода данных, необходимых для PERT-анализа, используется специальная панель инструментов, вызываемая командами Вид | Панели инструментов | Анализ по методу PERT:



Рис. 28. Панель инструментов для PERT

Данная панель делает доступными следующие возможности

- Ввод данных о задачах с оптимистической, ожидаемой и пессимистической оценками времени.
- Непосредственное вычисление продолжительности отдельных задач и проекта в целом по методу PERT.
- Ввод данных о различных длительностях задачи в виде формы.
- Настройка числовых коэффициентов w_1 , w_2 , w_3 для метода PERT (сумма коэффициентов должна быть равна 6).
- Ввод данных о различных длительностях задач в виде таблицы.

После проведения расчетов по методу PERT длительности любой задачи, для которой были введены оптимистическая, ожидаемая и пессимистическая оценки, пересчитывается по формуле, указанной выше.

В MS Project отсутствуют стандартные средства для расчета и отображения дисперсии отдельной задачи. Данный недостаток устраним введением дополнительного настраиваемого поля в таблицы, отображающие длительность задач. Для создания поля воспользуемся командами Сервис | Настройка | Поля... (рис. 29).

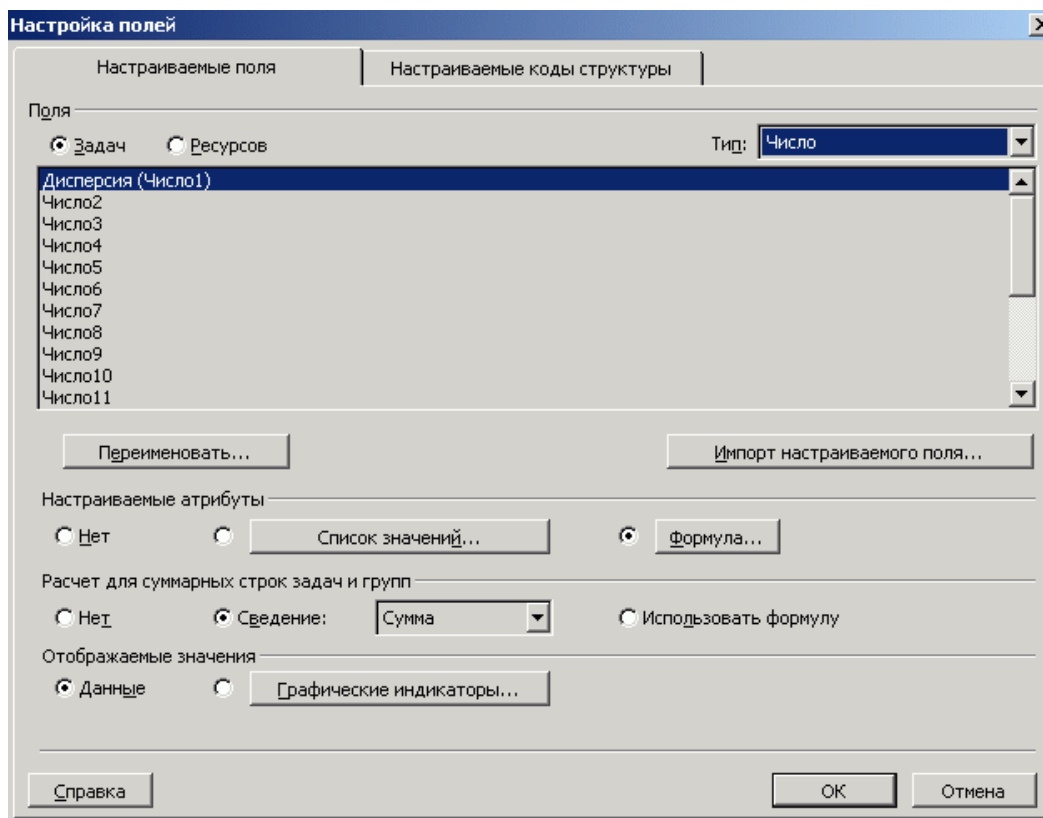


Рис. 29. Окно ввода и настройки пользовательских полей

В появившемся диалоговом окне необходимо выбрать поле для задач, числовой тип поля, переименовать одно из полей в списке (в нашем случае новое название – *Дисперсия*), установить флажок расчета поля по формуле и флажок расчета для суммарных строк задач и групп как «Сведение: Сумма». Формула для расчета дисперсии вводится в соответствующем диалоговом окне (рис. 30).

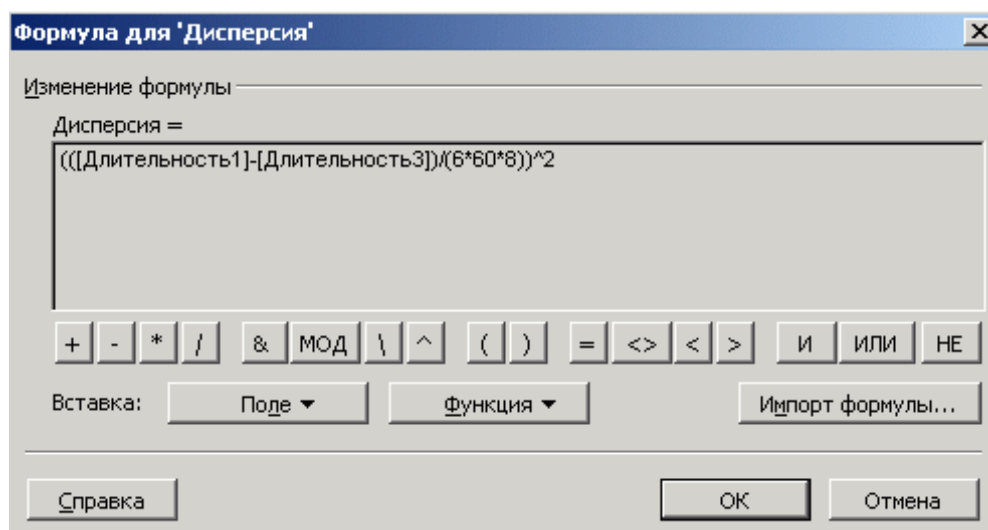


Рис. 30. Формула для расчета дисперсии

Внутренние поля [Длительность1] и [Длительность3] содержат оптимистическую и пессимистическую оценку продолжительности задачи. Обратите внимание на использование множителя 60*8. Так как длительность задач рас-

считывается в минутах, необходимо привести ее в соответствие с вводимой длительностью. В нашем случае длительность приводится к дням. Считается, что в нашем рабочем календаре восемь рабочих часов.

После определения настраиваемого поля для вставки его в таблицу необходимо выполнить команду контекстного меню (вызвать в заголовке таблицы) Вставить столбец.

1.12. АНАЛИЗ РИСКОВ

Риск – это вероятность события или стечения обстоятельств, которые могут оказать на проект отрицательное влияние. Процесс управления рисками состоит в определении, анализе и устранении рисков проекта с тем, чтобы они не могли превратиться в проблемы и вызвать ущерб или потери.

Анализ рисков состоит из нескольких этапов. Вначале определяются возможные риски. Затем для каждого риска определяется стратегия смягчения влияния риска на проект, то есть действия, предпринимаемые для предотвращения риска или в случае осуществления риска для того, чтобы проект был успешно завершен.

Определение риска в MS Project реализуется, в основном, как применение фильтра к одному или нескольким существующим (или пользовательским) полям таблицы с данными.

Выделяют следующие типы рисков:

I. Риски в расписании.

1. Задачи с предварительными длительностями
2. Слишком короткие задачи
3. Слишком длинные задачи
4. Задачи с большим числом ресурсов
5. Задачи с большим числом зависимостей

II. Ресурсные риски.

1. Использование неопытных сотрудников
2. Ресурсы с большим объемом работ
3. Ресурсы со сверхурочной работой
4. Сотрудники с уникальными навыками
5. Материалы с единственными поставщиками

III. Бюджетные риски.

Задачи с предварительными длительностями – это обычно задачи, в выполнении которых у сотрудников нет опыта. Обнаруживаются такие задачи в проекте с помощью стандартного фильтра Проект | Фильтр: Задачи с оценкой длительности. Для задач с предварительной длительностью желательно увеличить планируемую длительность до пессимистической. Кроме того, можно добавить в проект отдельную задачу по освоению нового оборудования или технологии, выполняемую раньше того, как это оборудование или технология будут использоваться.

Задачи со слишком короткой длительностью возникают в проекте либо из-за оптимистической оценки сотрудником времени выполнения задачи, либо

как результат ошибки менеджера, выделяющего на задачу время, не советуясь с исполнителями. Для выявления коротких задач требуется выделить все задачи с длительностью меньше дня (или другого срока) и задачи, у которых при PERT-анализе расчетная длительность совпала с оптимистической (с определенной поправкой). Данную операцию можно выполнить при помощи пользовательского фильтра (Проект | Фильтр | Другие фильтры | Создать...).

Появившееся диалоговое окно позволяет сконструировать фильтр, используя имена полей, логические связки и операции сравнения.

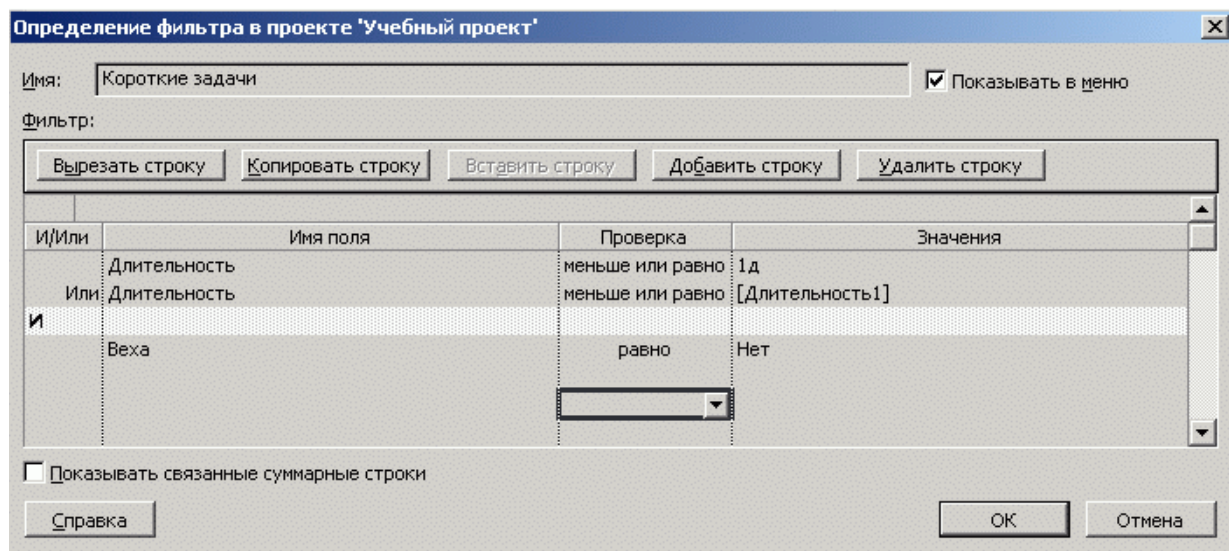


Рис. 31. Определение фильтра в проекте

Представленный на рис. 31 фильтр выделяет задачи с длительностью, меньшей или равной одному дню, или те задачи, у которых длительность меньше или равна оптимистической длительности в методе PERT (внутренне поле [Длительность1]). Не выделяются задачи-вехи.

Слишком длинные задачи делают сложным оценку трудозатрат и загрузку ресурсов. Такие задачи должны быть разбиты на более мелкие. Для выявления длинных задач следует воспользоваться фильтром или автофильтром (Проект | Фильтр | Автофильтр).

Для выявления *задач с большим числом ресурсов* можно использовать настраиваемое поле, значение которого вычисляется по формуле $Len([Названия\ ресурсов])$. Данная формула вычисляет длину строки внутреннего поля, которое хранит список всех ресурсов, назначенных на задачу. Естественно, это позволяет оценить количество назначенных ресурсов только приблизительно.

Для *задач с большим числом зависимостей* увеличивается риск срыва из-за задержки задачи-предшественницы. Стандартными средствами MS Project определить количество задач-предшественниц нельзя. Для этого в общем случае требуется использование специальных макросов, написанных с использованием Visual Basic for Applications (данная возможность в программе поддерживается). При помощи пользовательского фильтра можно проверить наличие символа «;» в поле Предшественники. Это поле содержит строку с именами задач-

предшественников, разделенных точкой с запятой. Следовательно, фильтр позволит отобразить задачи, у которых более чем один предшественник.

Перейдем к рассмотрению ресурсных рисков. Возможность задержки выполнения задачи при использовании *неопытных сотрудников* очевидна. Для того чтобы пометить трудовой ресурс как не имеющий опыта, можно использовать созданное настраиваемое поле. Рекомендации по устранению данного риска: избегайте от неопытных сотрудников в проекте или увеличьте срок задачи, которую выполняет данный сотрудник, до пессимистического.

Ресурсы с большим объемом работ и ресурсы со сверхурочной работой выявляются при помощи автофильтров, применяемых в видах Использование ресурсов или Использование задач. Для определения *сотрудников с уникальными навыками* следует применить настраиваемое поле. *Материалы с единственными поставщиками* могут быть отмечены при помощи текстового комментария.

После выявления рисков проекта необходимо определить меры, смягчающие влияние рисков на проект. Это можно сделать двумя путями: разработать план сдерживания рисков или план реакции на них.

План сдерживания рисков (mitigation plan) состоит из задач, которые включаются в основной план проекта и, будучи выполненными, существенно снижают вероятность осуществления риска. Желательно, чтобы задачи плана сдерживания рисков не влияли на общую длительность проекта.

План реакции на риски (contingency plan) определяется в проекте, но не оформляется в виде задач до осуществления риска. Если риск осуществляется, нужные задачи добавляются в план проекта. В основном плане проекта задачи плана реакции на риски оформляются в виде текстовых комментариев, связанных с определенными задачами или ресурсами.

Для смягчения влияния рисков рекомендуется заложить в проект финансовый и временной буфер. *Финансовый буфер* реализуется простым увеличением стоимости проекта на некий коэффициент (обычно 5-15%). Для реализации *временного буфера* в проект помещаются дополнительные задачи-буферы, составляющие элементы критического пути. Для некритических задач величина временного буфера рассчитывается по возможным временным задачам. Общая закономерность для любого проекта: чем меньше в проекте критических задач, тем более устойчив проект к рискам в расписании.

1.13. ОТСЛЕЖИВАНИЕ ХОДА ВЫПОЛНЕНИЯ ПРОЕКТА

После составления плана проекта и его утверждения актуальной является задача отслеживания проекта по мере его выполнения. Для этого необходимо иметь некий «эталон», с которым в процессе выполнения проекта будет выполняться сверка. Таким эталоном выступает *базовый план проекта*.

Для отслеживания проекта используется группа команд меню Сервис | Отслеживание (рис. 32).

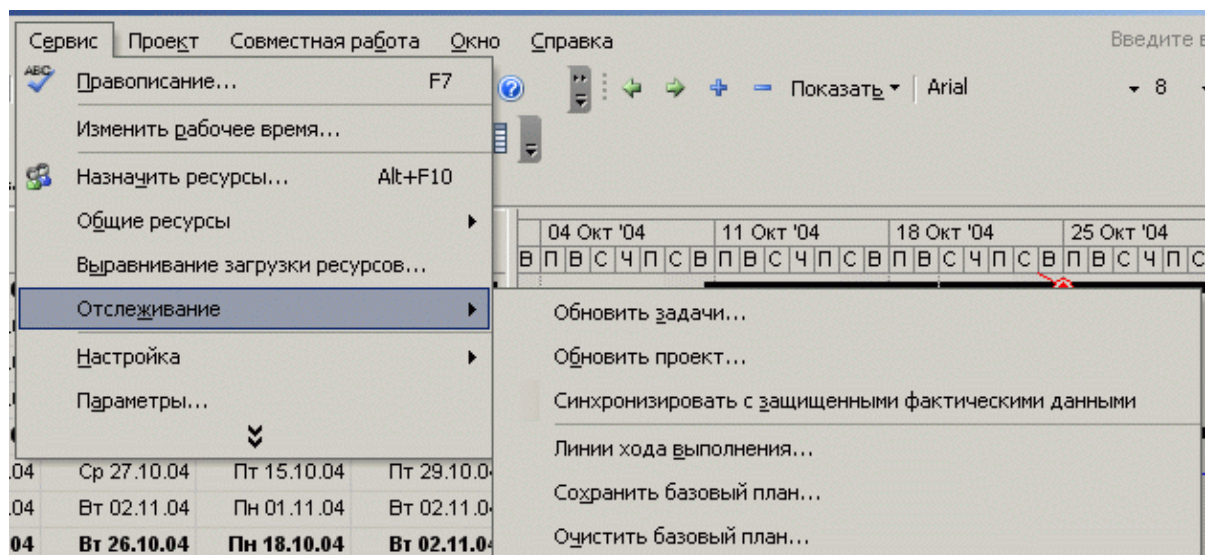


Рис. 32. Команды для отслеживания проекта

Базовый план сохраняется при помощи соответствующей команды данного меню. При сохранении базового плана можно настроить следующие параметры. Можно выбрать один из 11 «слотов» базовых планов, которые мы хотим сохранить. Можно сохранить базовый или *промежуточный план*. Промежуточный план (в отличие от базового) хранит только даты начала и окончания задач. Планы можно сохранить как для всего проекта, так и для выбранных задач.

Вид Диаграмма Ганта с отслеживанием отображает для каждой задачи такие параметры как фактическую длительность, длительность по базовому плану и процент выполнения. Процент выполнения отдельной задачи вводится в процессе выполнения проекта. Для этого можно использовать диалоговое окно свойств задачи или команду Обновить задачи. Для визуального контроля хода выполнения проекта можно вывести *Линию хода выполнения*. Данная линия размещается на графике Ганта и проводится через пункты, соответствующие проценту выполнения задач (рассчитанному автоматически или введенному вручную). Если для некоторой задачи данная линия отклонилась от вертикали вправо, то это означает, что задача выполняется с опережением. Если линия хода выполнения отклонилась влево, это означает задержку в ходе выполнения задачи.

2. ЭЛЕКТРОННОЕ ДЕЛОПРОИЗВОДСТВО

2.1. ДОКУМЕНТ, ДОКУМЕНТООБОРОТ, ДЕЛОПРОИЗВОДСТВО

Понятие *документа* является базовым в теории делопроизводства. Это понятие также активно используется в законодательстве и правовой сфере. В настоящее время используется несколько десятков различных определений термина «документ». Исторически одно из первых определений возникло и было закреплено в XIX веке, когда формировалось *архивоведение* – наука о работе с документами и их хранении. В архивоведении *документ* определяется как *материальный объект, содержащий зафиксированную на нем информацию*. Данное определение до сих пор используется в некоторых нормативных и законодательных актах. Исходя из определения, можно сформулировать типичные виды работ с документами: учет, размещение в хранилище, хорошее обеспечение условий хранения, передача во временное пользование и т. п. Ключевым моментом в определении является то, что документ воспринимается как материальный объект.

Со временем акценты работы с документами сместились. Архивисты и правоведы стали более интересоваться не материальной сущностью документа, а информационно-смысловой. Это потребовало соответствующим образом изменить определение документа.

Приведем несколько современных определений документа, принятых в законодательстве и архивоведении. В законе Российской Федерации «Об информации, информатизации и защите информации» (1995 г.) *документ* определяется как *зафиксированная на материальном носителе информация с реквизитами, позволяющими ее идентифицировать*. В этом определении явно прослеживается смещение акцента с материального носителя информации на саму информацию. Ключевым становится не материальная целостность документа, а его информационная целостность и безопасность. Данное определение документа используется в нескольких белорусских стандартах. В частности, в Госстандарте Республики Беларусь 2000 г. «Информационные технологии. Защита информации. Процедуры выработки и проверки электронной информационной подписи» и в Госстандарте РБ 2000 г. «Информационные технологии. Защита информации. Функция хеширования».

В англоязычной литературе используются следующие определения термина «документ». Международным советом по делам архивов в 1988 году принято определять *документ* как *комбинацию носителя и записанной на нем или в нем информации, которая может иметь доказательную или справочную ценность*. Данным определением закреплялся постулат о единстве и неразрывной связи носителя и информации. Группой зарубежных юристов, архивистов и специалистов в области информационных технологий было предложено определение *документа*, как *информации, созданной на некотором носителе некоторой техникой известной в настоящее время или могущей быть изобретенной в будущем*. Это определение является достаточно обобщенным, его цель –

избежать переписывания законов в будущем, при возникновении новых технологий, нового представления информации.

Рассмотрим понятия «документооборот» и «делопроизводство». Достаточно часто эти понятия воспринимаются как синонимы, хотя это и не соответствует действительности. Точное определение данных терминов можно найти в большинстве книг по современному делопроизводству.

Делопроизводство – это деятельность по созданию документов и организации работы с ними. Под «организацией работы» понимают создание условий, обеспечивающих движение, поиск и хранение документов. Из организации работы с документами следует особо выделить движение, так как движение документов между пунктами их обработки и есть непосредственно документооборот. Точнее, документооборот – это движение документов в организации с момента их получения или создания до завершения исполнения или отправки.

Делопроизводство, в зависимости от выполняемых в управлении функций, может быть организационно-распорядительным, бухгалтерским, нотариальным, кадровым, и т. п. Каждый из видов делопроизводства имеет свои отличительные особенности, однако общим для любой функции управления есть организационно-распорядительное (административное) делопроизводство. Поэтому, когда заходит речь об автоматизации делопроизводства, то имеется в виду автоматизация именно административного делопроизводства, как основы, базовой платформы для построения корпоративной информационной системы любого учреждения.

После определения понятий «документ», «документооборот», «делопроизводство» опишем стадии, которые проходит любой документ в процессе своего жизненного цикла. Эти стадии включают:

- создание;
- визирование, согласование;
- подписание, утверждение;
- регистрацию;
- рассмотрение;
- исполнение;
- списание в архив;
- хранение, уничтожение.

Движение документов в любой организации осуществляется в виде потоков документов, циркулирующих между пунктами обработки информации и пунктами технической обработки. По отношению к аппарату управления различают потоки входящих (поступающих), исходящих (отправляемых) и внутренних документов. Следовательно, когда мы говорим об автоматизации делопроизводства, то автоматизации подлежит движение документов и их обработка в пунктах обработки, а также все стадии жизненного цикла документов.

Отметим, что исторически делопроизводство в разных странах складывалось по-разному. Точнее, в зависимости от менталитета и культуры того или иного народа с годами создавалась и соответствующая система документоведения.

Советская система делопроизводства характеризовалась, прежде всего, строгой *вертикальной направленностью*. То есть, все документы, поступающие в организацию, после регистрации направлялись на доклад к руководителю. После рассмотрения документа руководитель накладывал резолюцию, в которой указывал ответственного исполнителя. Далее документ попадал к ответственному исполнителю, который либо исполнял документ, либо направлял его на исполнение своим подчиненным. Таким образом, документ, обрастая резолюциями, двигался вглубь к основанию иерархии. После того, как документ исполнялся, он совершал обратный путь снизу на самый верх, где и докладывалось об его исполнении. Западное же делопроизводство тяготеет к *горизонтальной схеме*, когда поступающие в учреждение документы сразу направляются исполнителям без доклада «наверх».

Еще одним принципиальным отличием нашей делопроизводственной практики от западной является наличие органа, контролирующего исполнение документа. Перед тем как отправить документ ответственному исполнителю, он ставится на контроль в делопроизводстве учреждения. Таким образом, третье лицо – делопроизводитель – всегда знает, у кого находится документ на исполнении и когда он должен быть исполнен.

Именно эти два принципиальных отличия и определяют разницу в подходах ведения делопроизводства у нас и на западе. Поэтому автоматизация делопроизводства в постсоветских странах при применении западных систем автоматизации должна вестись с учетом этих принципиальных отличий.

2.2. ЭЛЕКТРОННЫЙ ДОКУМЕНТ

Рассмотрим понятие «электронный документ». Пытаясь дать определение этому понятию, мы неизбежно будем отталкиваться от понятия документа. Выделить электронные документы (ЭД) позволяет их специфика: человек не может воспринять информацию электронного документа в той физической форме, в которой они хранятся на носителе информации. Электронная информация всегда закодирована как последовательность электронно-магнитных импульсов. Чтобы отобразить ее в понятном пользователю виде необходимо *декодирование* с помощью программных и аппаратных средств.

Приведем несколько формальных определений электронного документа и их источников.

Электронный документ – это документ, в котором информация представлена в электронно-цифровой форме, то есть в форме последовательности двоичных символов (Закон РФ «Об электронной цифровой подписи»).

Электронный документ – это информация, представленная как форма состояний элементов электронной вычислительной техники, иных электронных средств обработки, хранения и передачи информации, которая может быть преобразована в форму, пригодную для однозначного восприятия человеком, и имеющая атрибуты для идентификации документа (Постановление Пенсионного фонда РФ «О введении в системе Пенсионного фонда Российской Федера-

ции криптографической защиты информации и электронной цифровой подписи»).

Обязательными признаками электронного документа, отличающими его от бумажного, являются:

- машинный носитель информации;
- электронная форма представления информации (специально устанавливается требование о воспроизведении информации в виде, доступном для восприятия человеком);
- способность храниться, обрабатываться и передаваться с помощью электронных средств.

Дополнительно в качестве признака ЭД иногда указывается наличие в нем электронной цифровой подписи.

Теоретики делопроизводства выделяют в электронном документе несколько компонентов, описывающих его *структуру*. При этом используется несколько подходов. Кеннет Тибодо (Kenneth Thibodeau) рассматривает три компонента электронного документа:

1. *Содержание* – информация, содержащаяся в документе.
2. *Структура* – способ организации содержания.
3. *Контекст* – связь между ЭД и другим материалом.

Дэвид Бирмен дополняет три компонента Кеннета Тибодо четвертым:

4. *Метаданные* – данные для подтверждения доказательной силы ЭД и долговременного хранения.

Дэвид Бирмен выделяет в метаданных 6 «слоев»:

- a. *Регистрация* – делопроизводственные реквизиты;
- b. *Термины и условия* – условия доступа к документу;
- c. *Структура* – формат записи ЭД (для программного обеспечения);
- d. *Контекст* – информация, утверждающая доказательную силу ЭД;
- e. *Содержание* – описание содержания ЭД;
- f. *История использования* – регистрация факта использования ЭД (время и т. п.).

В дальнейшем будем придерживаться следующей схемы деления ЭД на компоненты:

- I. *Содержание* – информация, которая документируется.
- II. *Контекст* – деловые, правовые, архивные, технические, делопроизводственные реквизиты, которые фиксируют различные моменты создания документа.
- III. *Метаданные* – технологическая информация, необходимая программному обеспечению для управления данными и их представления в удобном для пользователя виде.
- IV. *Носитель информации*.

Рассмотрим метаданные подробнее. *Метаданные* – это единство типов данных, структур данных и форматов данных. *Типы данных* можно классифицировать таким же образом, как это делается в любом языке программирования (простые – символьные данные, числовые; составные – календарные даты, векторы, матрицы). Типы данных задают множество допустимых значений данных и операций над ними.

Структуры данных описывают множество допустимых типов данных документа и отношения между ними. Структура данных обычно имеет несколько уровней. На нижнем уровне находится *элемент данных*, далее следует *запись* как совокупность некоторых однотипных или разнотипных элементов, затем *файл* – совокупность записей, и на верхнем уровне – *совокупность файлов*.

Выделяют несколько видов структур данных: линейная, иерархическая, сетевая, реляционная.

1. Линейная структура данных – связываемые единицы данных последовательно следуют одна за другой (пример: слово (буквы), фразы (слова), простой текстовый файл). Линейная структура данных самая примитивная и уязвимая: удаление единицы данных может привести к искажению информации. Линейная структура данных используется обычно на первом уровне структур.

2. Иерархическая структура данных – связь между единицами данных, при которой у каждой единицы есть предыдущая (доминирующая) и произвольное число последующих (подчиненных) (пример: каталоги файлов). Иерархическая структура данных более устойчива: нарушение связей в одной ветви не приводит к дезинтеграции системы в целом.

3. Сетевая структура данных – единицы данных являются узлами, связанными с произвольным числом других узлов (пример: гипертекст).

4. Реляционная структура данных – единицы данных собраны в таблицы вида «идентификатор записи – поле записи». При этом обработка записи ведется целиком (пример: реляционная БД).

Формат данных – это способ описания единиц данных и структуры данных. Формат данных зависит от типа данных, структуры данных, а также стандарта описания. Формат данных определяет тип программного обеспечения для работы с данными: графические и текстовые редакторы, электронные таблицы, СУБД.

Формат данных – самая непостоянная часть электронного документа. При изменении программного обеспечения для работы с документом зачастую необходимо менять формат представления данных (так называемая «миграция данных»). С форматом данных связаны проблемы совместимости форматов, открытости форматов, соответствия миграционной копии оригиналу.

Разберем вопрос классификации электронных документов. Задачей классификации является выявление общих свойства и характерных отличий документов, образующихся в ходе деятельности организаций и частных лиц. Классификация применяется как база для дальнейшей работы по обеспечению сохранности, безопасности и учета документов.

Существует несколько подходов к классификации ЭД. Первый подход закреплен в «Примерной инструкции по работе с машиночитаемыми документами в организациях, на предприятиях и в ведомственных архивах Республики Беларусь» (официальный документ 1996 года). Согласно данному подходу выделяется 4 вида классификации:

а) по носителю информации (магнитные диски, магнитооптика, магнитные ленты и т. п.);

б) по информационной технологии (автоматизированные системы, БД, информационно-поисковые и информационно-справочные системы, программы);

в) в зависимости от комплектности (только файлы, файлы и документация на бумаге);

г) по видам деятельности организаций (проектная, производственная, коммерческая, управленческая).

Другой подход предполагает классификацию ЭД по комбинации метаданных:

а) Одноранговые ЭД: документ состоит из одного типа данных, объединяемых одной структурой записи в файл определенного формата (простые текстовые файлы, оцифрованные изображения, аудио- и видео документы);

б) Двухранговые ЭД: несколько типов данных, объединяемых одной структурой записи в один файл (однотабличные реляционные БД, документы, состоящие из графических примитивов (чертеж, схема));

в) Трехранговые ЭД: несколько типов данных, несколько структур данных, один файл (файл издательской системы: текст + графика + таблица);

г) Четырехранговые ЭД: несколько типов данных, несколько структур данных, несколько файлов одного формата (гипертекстовые системы, документы САПР);

д) Пятиранговые ЭД: несколько типов данных, несколько структур данных, несколько файлов различного формата (Internet-сайт, мультимедиа энциклопедии, документы корпоративных информационных систем).

Рассмотренная классификация позволяет выявить прямую зависимость между рангом документа и затратами на его построение и обеспечение сохранности – чем выше ранг электронного документа, тем эти затраты больше.

2.3. ЭЛЕКТРОННЫЙ ДОКУМЕНТООБОРОТ

На сегодняшний день автоматизация документооборота также необходима, как автоматизация бухгалтерского учета в середине девяностых годов. Причин этому много. Во-первых, информацию необходимо обрабатывать как можно быстрее и качественнее, подчас информационные потоки не менее важны, чем материальные. Во-вторых, потеря информации или ее попадание в чужие руки может обойтись весьма дорого. Можно выделить ряд проблем, общих для тех организаций, где работа с документами ведется традиционным способом:

1. документы теряются;
2. накапливается множество документов, назначение и источник которых не ясны;

3. документы и информация, содержащаяся в них, попадает в чужие руки;
4. тратится масса рабочего времени на поиск нужного документа и формирование тематической подборки документов;
5. создается несколько копий одного и того же документа – на бумагу и копирование документов тратиться немало средств;
6. на подготовку и согласование документов тратится много времени.

Внедрение системы электронного документооборота позволяет решить все эти проблемы, а также:

- обеспечит слаженную работу всех подразделений;
- упростит работу с документами, повысит ее эффективность;
- повысит производительность труда сотрудников за счет сокращения времени создания, обработки и поиска документов;
- повысит оперативность доступа к информации;
- позволит разграничить права доступа сотрудников к информации.

Приведем некоторые данные из книги Майкла Дж. Д. Саттона «Корпоративный документооборот». Автор утверждает, что средняя стоимость жизни одного бумажного бланка, включая стоимость его разработки, распечатки, заполнения, хранения, поиска и т. д., составляет около 90 долларов. Это означает, что организация, где работают 2000 сотрудников, каждый из которых в год заполняет по 200 бланков (примерно по одному в день), тратит на только на это 36 млн. долларов в год! «Если внедрение систем электронного управления документами, – пишет автор, – позволит сократить хотя бы треть этих расходов, экономия составит 12 млн. долл. в год».

Еще один пример Майкла Дж. Д. Саттона. Один из главных эффектов, достигаемых от внедрения систем электронного управления документами – существенное уменьшение времени на поиск информации. Оказывается, что в компании, где с документами работают 2000 сотрудников, внедрение систем электронного управления документами приводит к экономии времени в объеме 200-400 человеко-лет (попросту говоря, можно уволить или найти лучшее применение как минимум для 200-400 человек!).

Обосновав необходимость внедрения систем электронного управления документами, опишем применяемое деление таких систем на подсистемы.

Выделяют следующие подсистемы автоматизации документооборота.

- Системы автоматизации делопроизводства;
- Архивы документов;
- Системы ввода документов и системы обработки образов документов;
- Системы управления стоимостью хранения документов;
- Системы маршрутизации документов;
- Системы комплексной автоматизации бизнес-процессов.

Каждая подсистема обладает набором специфических для нее функций. При этом отдельные подсистемы тесно взаимодействуют между собой. Разделение системы документооборота на подсистемы, предпринимаемое нами, носит несколько «академический» характер. В реальной практике программные продукты достаточно условно можно отнести к той или иной группе в нашей

классификации. Как правило, системы реализуют лишь часть функций, описанных ниже, при этом продукт одного класса может включать в себя часть функций систем другого класса. Поэтому построение систем автоматизации документооборота из существующих на рынке продуктов требует не только хорошего понимания конечной задачи, но и отличного знания рынка программного обеспечения.

Кратко остановимся на функциях каждой из перечисленных подсистем системы автоматизации документооборота.

2.3.1. Системы автоматизации делопроизводства

Функции автоматизации делопроизводства в том или ином виде представлены в любой системе автоматизации документооборота. В функции систем автоматизации делопроизводства не входит хранение и перемещение документов в организации. В их функции входит фиксация документов в специальной базе данных, выражающаяся в заполнении специальной карточки документа. Содержимое карточки документа может варьироваться в зависимости от сложившейся в организации ситуации. Структура документов, зафиксированных в базе данных, опирается на так называемую номенклатуру дел, имеющуюся, как правило, в каждой организации, а технология учета и обработки документов опирается на сформулированное в данной организации «Положение о делопроизводстве». Документы хранятся в бумажном виде в специальном архиве, но в базе данных отображается их текущее местоположение и статус, включая атрибуты контроля исполнения. Обычно в системах делопроизводстве различают входящие и исходящие документы, нормативно-распорядительные документы, документы коллегиальных органов управления, справочные документы. Документы, находящиеся на контроле исполнения, подразделяются по исполнителям, статусу исполнения, срокам исполнения и прочее. Каждый документ в системе представляет собой запись в базе данных, характеризующуюся набором значений атрибутов карточки. Помимо учета и поиска документов в базе данных, система должна обеспечивать генерацию отчетов, позволяющих получить ведомости исполнения документов и прочую сводную информацию.

Для разработки приложений, выполняющих функции автоматизации делопроизводства больше всего подходят стандартные инструменты, используемые для разработки автоматизированных рабочих мест, от настольных баз данных до систем на базе различных SQL серверов. Однако в том случае, если автоматизация документооборота не закончится данным шагом, то можно подумать и о других инструментах, обеспечивающих более последовательное развитие системы. Так, например, при переходе к электронному хранилищу документов база данных системы делопроизводства должна содержать ссылки на соответствующие объекты электронного архива, при использовании электронных средств маршрутизации документов система должна обеспечивать возможность рассылки документов на рабочие места пользователей, определения текущего местоположения документа и так далее.

2.3.2. Архивы документов

Архив документов это то, что собственно хранит электронный документ. При этом может храниться либо образ документа, либо его содержание, либо и то и другое. Помимо собственно хранения документов, архив должен обеспечивать навигацию по иерархии документов и их поиск.

В отличие от поиска по атрибутам документов, который имелся и в системах предыдущего класса, архивы документы должны обеспечивать полнотекстовый поиск по содержимому текстовых фрагментов в документе. В предельном случае поисковый механизм должен обладать некоторым интеллектом, то есть обеспечивать поиск близких грамматических конструкций, а также поиск близких по смыслу слов.

В архивах хранятся сами документы, поэтому система должна обеспечивать разграничение прав доступа к документам. Пользователь может идентифицироваться либо посредством сетевого имени, либо с помощью специального имени и пароля, определенного в системе управления архивом. Помимо разделения прав доступа на уровне пользователей система должна обеспечивать выделение групп пользователей или ролей.

Следующей функцией архива документов является обеспечение возможности групповой работы с документами, находящимися в стадии создания – это функция блокировок документов. Если один из пользователей системы начинает редактировать документ, он блокируется для доступа других пользователей до тех пор, пока с ним не закончится работа.

Еще одной функцией архива является поддержка контроля версий. Версии документов могут фиксироваться либо автоматически, либо по инициативе пользователя. В случае необходимости пользователь может вернуться к одной из предыдущих версий документа.

К сервисным функциям архива документов относятся возможность создания резервных копий документов без прекращения работы системы, интеграция с системами обеспечения оптимальной стоимости хранения данных и прочее.

2.3.3. Системы ввода и системы обработки образов документов

Одной из самостоятельных функций систем документооборота является ввод документов в архив. Под этим понимается перевод документов из бумажного вида в электронный. В простейшем случае эта процедура сводится к простому сканированию. Однако, как правило, простого сохранения образа документа оказывается недостаточно.

Образ документа может потребовать так называемого аннотирования, наложения на образ документа различных дополнительных образов, выделений, текстовых пометок и прочее. Помимо этого, образ документа должен быть снабжен набором атрибутов, который позволит его идентифицировать в системе делопроизводства и в архиве. Эти операции производятся вручную.

Более сложной функцией является автоматическое распознавание содержимого образа документа и формирование документа, содержащего его текст.

Для этого предназначены программы, относящиеся к классу ПО распознавания текста. Еще более сложной функцией является распознавание содержимого форм. При этом программа определяет наличие записей, в том числе и рукописных, в определенных полях бланка документа, распознает его содержимое и автоматически заполняет значения атрибутов данного документа в системе. При необходимости значения определенных полей бланка может выбираться из определенного в системе справочника.

2.3.4. Системы управления стоимостью хранения документов

Совершенно очевидно, что при сохранении в архиве образов документов объемы хранения могут быстро расти и достигать значительных объемов. При этом интенсивность обращения к документам, находящимся в архиве, далеко не равномерна. Документы, находящиеся в работе, очевидно, требуются достаточно часто, в то время как доступ к документам, работа с которыми уже завершена, осуществляется очень редко. Соответственно, система может обеспечивать различную оперативность доступа к различным документам. Так как стоимость хранения документов в архиве, как правило, обратно пропорциональна скорости доступа, то можно воспользоваться отмеченной закономерностью для оптимизации стоимости содержания архива.

Системы управления стоимостью хранения как раз и решают данную задачу. Они обеспечивают возможность работы с различными периферийными устройствами – накопителями на жестких магнитных дисках, оптическими стойками, накопителями на магнитной ленте и CD-ROM устройствами. Система осуществляет автоматический перенос данных с быстрых, но «дорогих» устройств, на более «дешевые» устройства в случае, если доступ к данным осуществляется недостаточно часто.

2.3.5. Системы маршрутизации документов

Системы маршрутизации документов занимаются непосредственно пересылкой документов на рабочие места исполнителей, сбором информации о текущем статусе документов, осуществляют консолидацию документов по завершению работы с ними на отдельных этапах, а также обеспечивают средства доступа к информации о текущем состоянии работ с документами.

Системы маршрутизации, как правило, содержат средства описания типовых маршрутов прохождения документов в организации. На основании разработанных маршрутных схем могут порождаться экземпляры бизнес-процессов работы с документами. В данном случае можно говорить о жесткой маршрутизации.

Альтернативой является так называемая свободная маршрутизация, при которой маршрут формируется «стихийно». Каждый пользователь системы, обладающий соответствующими правами, может определить следующего или следующих пользователей документа.

Администратор системы и менеджер, курирующий конкретный бизнес-процесс, может контролировать текущее состояние маршрута и вносить различные корректирующие воздействия в случае необходимости.

При маршрутизации документов возможны две схемы, назовем их Off-Line и On-Line. В первом случае при пересылке документа на рабочее место пользователя происходит его физическое извлечение из архива документов и доставка (например, с помощью электронной почты) на рабочее место клиента. По завершению работы документ обратно погружается в архив. В этом случае система маршрутизации сама является клиентом архива документов и вносит соответствующую информацию в учетную базу данных.

Вторая схема не подразумевает физического перемещение документа. Система маршрутизации документов обеспечивает клиенту интерфейс для доступа к заданиям на обработку документов, находящихся в архиве.

2.3.6. Системы комплексной автоматизации бизнес-процессов

Развитием систем маршрутизации документов являются Workflow-системы, или системы комплексной автоматизации бизнес-процессов. В отличие от систем маршрутизации документов, объектом маршрутизации в них является совокупность данных используемых в некотором бизнес-процессе. Пользователь получает на рабочее место информацию о том, что он должен сделать и все необходимые для этого данные. Workflow-приложение определяет, какое приложение должно быть запущено для реализации функций на данном рабочем месте, и загружает в него необходимые данные. Парадигма Workflow-системы предполагает, что пользователь должен выполнять только необходимые функции, всю рутинную работу – определение последовательности действий, доставку необходимой информации, контроль своевременности исполнения работы и прочее – выполняет система Workflow.

Функции Workflow-приложений выходят за рамки функций систем документооборота, однако, технологии, используемые в данных приложениях очень близки технологиям, используемым в системах маршрутизации документов, к тому же маршрутизация документов может рассматриваться как частный случай задачи построения Workflow систем, поэтому мы уделили им некоторое внимание.

2.3.7. Уровни внедрения систем электронного документооборота

Отметим следующий аспект, связанный с «глубиной» внедрения систем электронного управления документами в делопроизводство. Принято выделять три уровня внедрения информационных технологий.

1 уровень: автоматизированные системы управления бумажным документооборотом. Эти АС обращаются не с самим документом, а с сопровождающей информацией. По сути, это электронный аналог регистрационных карточек, журналов «Входящие и исходящие документы». Реквизиты бумажного документа заносятся в БД, по ним и определяется, чей он, когда поступил и так да-

лее. В настоящее время данный уровень считается пройденным этапом в большинстве организаций.

2 уровень. Ключевой признак этого уровня: наличие в системе электронной версии каждого документа. Это позволяет вести поиск по содержимому документа, обмениваться документами по сети, переслать документы по электронной почте.

В системе второго уровня обычно присутствуют справочные базы данных для облегчения деятельности (например, «код товара – наименование товара»). Нередко как подсистемы систем второго уровня поставляются программы сканирования и распознавания текста.

Системам второго уровня присущи следующие ограничения: они оперируют не электронными документами, а *электронными копиями* бумажных документов. Бумажные документы все равно есть и именно они имеют юридическую силу (подпись/печать). Именно бумажные документы обязаны храниться в архивах, а их электронные копии сохраняются только для удобства.

3 уровень. Система электронного управления документами третьего уровня должна обладать следующими возможностями:

- процедура экспертизы целостности документа;
- поддержка сертифицированных средств электронной цифровой подписи;
- подсистема хранения ключей к цифровой подписи¹.

В настоящее время в Беларуси действует только одна полноценная система этого уровня: Автоматизированная система межбанковских расчетов, поддерживается Белорусским межбанковским расчетным центром (БМРЦ) (разработчик – ИВА). Система позволяет выполнять платежи между банками в электронной форме с поддержкой цифровой подписи без бумажных подтверждений.

Для активного внедрения систем третьего уровня должна быть решена задача обеспечения подлинности электронных документов на таком уровне, который бы позволил отказаться от бумажных двойников. Также необходимы изменения в законодательной базе для архивов, чтобы они могли принимать электронные документы на хранение.

В заключение рассмотрим специфику, присущую корпоративным системам электронного управления документами. Корпорация – это крупное, территориально распределенное предприятие с многоуровневой системой управления. Корпоративные системы обладают следующими свойствами. **Масштабируемость** системы гарантирует, что ее производительный потенциал соответствует потребностям корпорации (или опережает их) в любой момент времени. Масштабируемость системы управления документами достигается размещением ее на масштабируемой программно-аппаратной платформе. Как правило, это означает, что корпоративная система является **многоплатформенной**. Следст-

¹ Некоторые системы второго уровня позволяют ставить электронные подписи под документами. Но это программное обеспечение обычно не стандартизировано, следовательно, имеет силу только для данной системы.

вием территориальной распределенности корпорации является необходимость **обеспечения распределенной работы и удаленного доступа к данным.**

Ниже перечислен стандартный набор функций корпоративной электронной системы управления документами:

- Единая система нумерации и учета документов.
- Возможность обмена регистрационными данными и ЭД между подразделениями.
- Сквозная система контроля исполнения документов.
- Аккумуляция всех документов, материалов и информации о документах, относящихся к одному вопросу в одно «дело».
- Создание единого эталонного бланка нормативных актов корпорации.
- Организация рассылки нормативных актов в подразделение.
- Внедрение единой технологии делопроизводства во всех региональных подразделениях.
- Возможность обмена электронными документами с другими системами управления документами.

2.4. LOTUS NOTES/DOMINO – ОБЩАЯ ХАРАКТЕРИСТИКА

Lotus Domino и Notes дают в руки разработчиков информационных систем новую уникальную по своим возможностям мощную среду разработки прикладных корпоративных систем и Internet/intranet-приложений.

Продукт Lotus Notes (Lotus Development Corporation, an IBM subsidiary) – наиболее успешный пример программного обеспечения для рабочих групп. На базе коммуникационных технологий Lotus и Java-технологий создаются системы информационной поддержки деловых процессов, управления знаниями, систем электронного документооборота, автоматизации организационно-распорядительной и производственно-хозяйственной деятельности различных фирм, предприятий и организаций. Notes уже достаточно широко используется в различных фирмах для организации совместной работы с документами и для координации деятельности сотрудников.

Делопроизводство и документооборот включают в себя работу с большим количеством разнородных документов, информация в которых слабо структурирована. Эти документы живут во времени и необходимо обеспечить последовательность действий над документом в течение его жизненного цикла и организовать взаимодействие вовлеченных в этот процесс людей.

Очевидно, что создание такой системы не тривиальная задача и требует нового подхода для решения, отличающегося от принципов разработки стандартных информационно-справочных систем и традиционных баз данных реляционного типа.

Поэтому и появился новый продукт для построения деловых приложений – Lotus Notes. Notes объединил достаточно хорошо известные современные технологии:

- Клиент – серверная технология.

- Система тиражирования данных (реплицирования).
- Система управления документоориентированной базой данных.
- Средства разработки приложений.
- Электронная почта.
- Средства защиты информации.

Каждая из этих технологий, взятая в отдельности, достаточно хорошо известна. Но объединенные вместе в рамках единой системы они дали совершенно новое качество.

Notes использует архитектуру клиент-сервер. Локальная сеть состоит из сервера Notes и рабочих станций клиентов. Локальные сети могут быть объединены в глобальную сеть Notes. Для этого серверы Notes объединяются с помощью модемов и коммутируемых или выделенных телефонных каналов. Одним из самых сильных свойств Notes – возможность неограниченного увеличения численности групп и их территориального расположения. Notes предоставляет возможность обмениваться информацией почти любого вида с практически неограниченным числом сотрудников по всему миру. Информация оптимальным образом распределяется между серверами и рабочими станциями. Наиболее трудоемкие процессы выполняются на сервере.

Реплицирование (тиражирование) баз данных – мощный и гибкий механизм обмена информацией между центральной базой на сервере и копией базы пользователя. Вы можете работать с копией БД и периодически связываться с сервером Notes для синхронизации своей копии БД с центральной базой на сервере. На всех серверах, входящих в глобальную сеть Notes, автоматически поддерживается синхронизация баз данных. На серверах Notes могут находиться *реплики БД* – специальные копии БД, обеспечивающие их синхронизацию. Изменения в реплике базы данных на одном сервере Notes отражаются в реплике базы данных, на другом сервере, как только серверы обмениваются информацией.

Notes – открытая система для распределения информации и построения деловых приложений и может функционировать на различных платформах серверов и клиентов. Изоляция разработчика от платформы – единственный путь для создания действительно открытой системы. Прикладные разработчики (на уровне API, или в интерфейсе пользователя Notes) получают три ключевых слоя изоляции: от операционной системы, от сетевого протокола и от физического местоположения хранилища объектов – БД Notes (через удаленный вызов процедур – remote procedure call – RPC). Система удаленного вызова процедур (RPC) направляет запросы либо на локальный диск, либо на соответствующий сервер. Web Navigator предоставляет полный доступ к Интернету и позволяет включать информацию, находящуюся в Web, в базы данных Notes.

Электронная почта позволяет сотрудникам обмениваться сообщениями с другими пользователями Notes и с пользователями других e-mail продуктов, отправлять письма через Интернет, вести совместные проекты, подготавливать и контролировать выполнение решений совещаний, организовывать дискуссии, составлять расписания вашего времени, расписания необходимых встреч, ис-

пользовать факсовые и телексные сообщения, подписывать документы, получать новости и информационные рассылки. Notes поддерживает все основные стандарты Интернета. Предоставляет доступ к почте Интернета, телеконференциям, адресным книгам и Web-страницам. Можно использовать базы данных Notes и свою почту из Интернета.

Notes обеспечивает различные уровни защиты данных. Можно защитить информацию на уровне сервера, базы данных, документа или поля в документе, позволяет использовать электронные цифровые подписи.

Система управления документоориентированной базой данных позволяет хранить информацию произвольного формата, включая числа, тексты, табличные данные, графику, видеоизображения, изображения, полученные с помощью сканера, факсимильные сообщения, звуковая информация и т.д.

Средства разработки баз данных удобны и обладают большой мощностью. Используется объектно-ориентированная управляемая событиями модель программирования. Как реакция на события, возникающие при выполнении действий над объектами интерфейса, автоматически вызываются созданные разработчиком скрипты – программы на LotusScript. LotusScript – это BASIC-совместимый объектно-ориентированный язык программирования. Чтобы разработчик имел возможность спокойно оперировать из LotusScript необходимыми объектами, имеется более 30 встроенных классов, которые содержат в совокупности порядка четырехсот методов и свойств. Имеются классы, позволяющих получать из LotusScript доступ к информации из реляционных баз через стандартный ODBC-интерфейс. Этих возможностей окажется достаточно для решения задач, связанных с интеграцией реляционных баз и баз Notes.

Все это делает Notes главным и каждодневным инструментом, который пользователи организаций любых размеров применяют для совместного использования информации, в которой они нуждаются – будь то сообщение электронной почты, документы реляционных баз данных или массивы данных, накопленных на серверах или документ, созданный любым приложением или находящийся на сервере World Wide Web.

2.5. УПРАВЛЕНИЕ ДОСТУПОМ В LOTUS NOTES/DOMINO

Любая база документов содержит специальный элемент, описывающий возможности пользователей при работе с базой. Этот элемент называется *Access Control List*, ACL (список управления доступом). В ACL выделены стандартные уровни доступа. При редактировании ACL пользователю назначается один из уровней (при этом возможна более тонкая настройка прав доступа). Диалог для редактирования ACL текущей базы вызывается командами `File|Database|Access Control...`

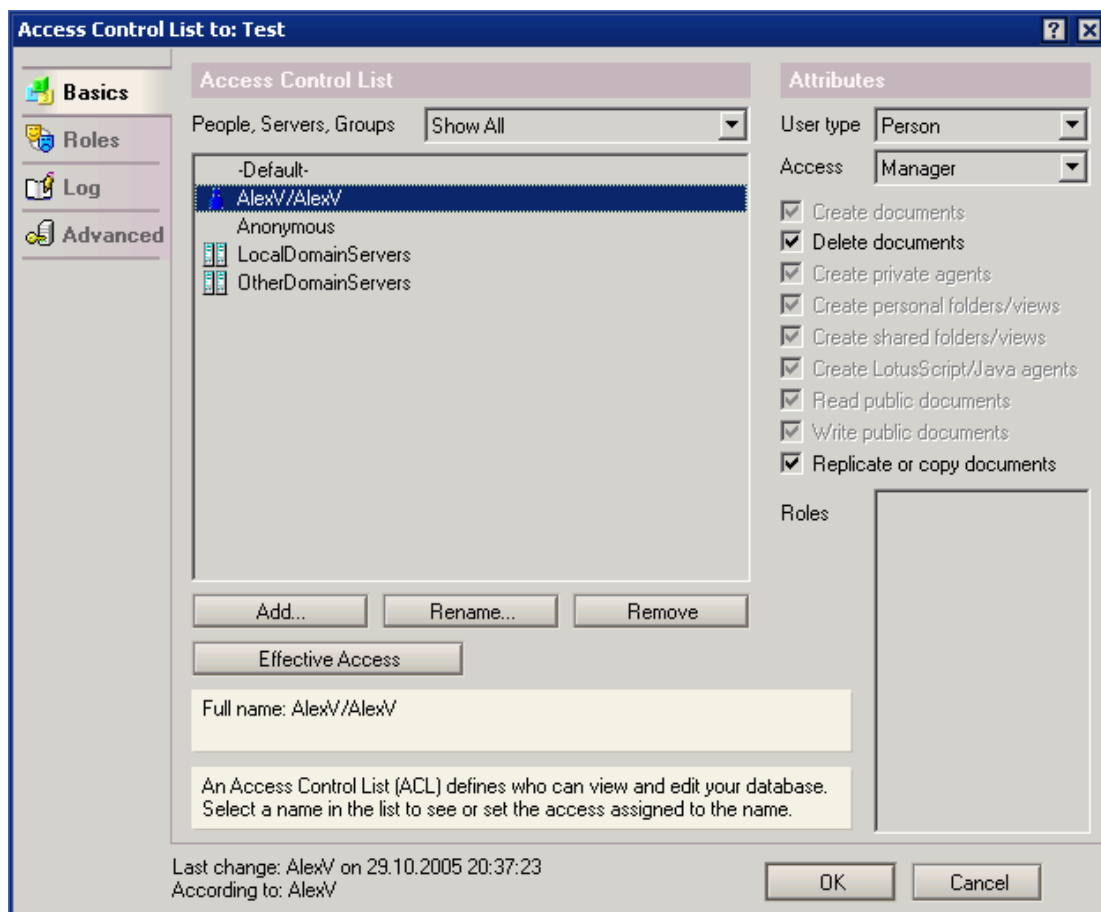


Рис. 33. Список управления доступом

Существует семь стандартных уровней доступа:

1. **Manager (управляющий)**. Может выполнять все действия в базе данных, включая чтение и редактирование документов; создание, модификацию и удаление элементов дизайна. Управляющий также может изменять ACL для остальных пользователей.
2. **Designer (разработчик)**. Может создавать, изменять и удалять документы и элементы дизайна, модифицировать репликационные формулы базы, создавать индекс полнотекстового поиска.
3. **Editor (редактор)**. Может читать, создавать и редактировать все документы в БД, в том числе и созданные другими пользователями, но не может изменить элементы дизайна, ACL и прочие атрибуты, присущие более высоким уровням доступа.
4. **Author (автор)**. Может читать все документы и создавать новые, но редактировать может только документы, созданные им.
5. **Reader (читатель)**. Может читать документы, но не может добавлять новые или редактировать существующие.
6. **Depositor (вкладчик)**. Может добавлять новые документы, но не может читать существующие, потому, что не видит их в представлениях базы.
7. **No Access (нет доступа)**. Полное отсутствие какого-либо доступа к базе (однако есть исключения, связанные с правами чтения и записи общих документов).

Возможности каждой группы пользователей демонстрирует табл. 10.

Таблица 10

Возможности уровней доступа

	No Access	Depositor	Reader	Author	Editor	Designer	Manager
Создание документов	-	+	-	-	+	+	+
Удаление документов	-	-	-	-	-	-	-
Создание агентов	-	-	-	-	-	+	+
Создание личных папок и представлений	-	-	-	-	-	+	+
Создание общих папок и представлений	-	-	-	-	-	+	+
Создание агентов на LotusScript/Java	-	-	-	-	-	-	+
Чтение общих документов	-	+	+	+	+	+	+
Запись и редактирование общих документов	-	+	+	+	+	+	+
Репликация или копирование документов	-	+	+	+	+	+	+

Если элемент в таблице выделен, то это означает, что его можно изменить в окне установки группы для отдельного пользователя. Удобным является рассмотрение *минимально возможных прав* (все изменяемые элементы установлены в -) и *максимально возможных прав* (все изменяемые элементы установлены в +). Так, например, пользователь группы Designer даже с минимальными правами может создавать документы в базе.

2.6. СОЗДАНИЕ БАЗЫ ДОКУМЕНТОВ В LOTUS NOTES/DOMINO

Создание новой базы документов можно осуществить тремя основными способами:

1. Путем копирования существующей базы.
2. Путем создания новой базы на основе шаблона.
3. Путем создания новой базы на основе пустого шаблона, т. е. «с нуля».

Создание новой базы путем копирования существующей выполняется командами File | Database | New Copy... При этом появляется диалоговое окно, в котором настраиваются следующие параметры:

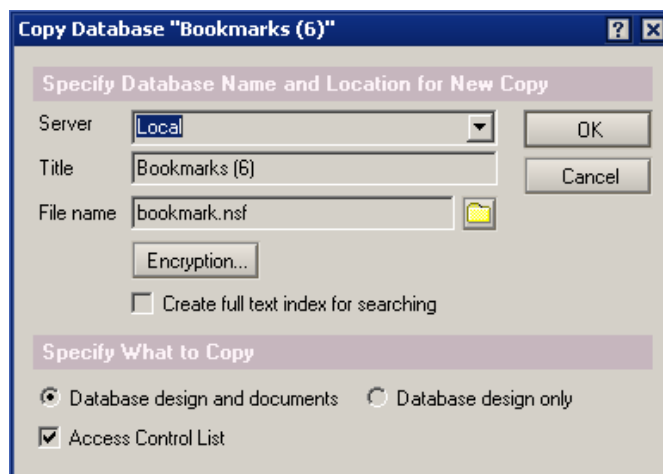


Рис. 34. Создание новой базы документов

1. *Server* – выбор сервера Domino, на котором будет создаваться база документов. Для локальных баз (которые не размещаются на сервере) следует установить это значение в *Local*.

2. *Title* – произвольное имя новой базы длиной до 96 символов. Данное имя будет отображаться на панели закладок (*Bookmark bar*).

3. *File name* – имя файла, в котором будет сохраняться база. Должно иметь расширение **.nsf*. Если местоположение файла не выбирается, то он размещается в специальной директории *Data*.

4. Кнопка *Encryption...* открывает диалоговое окно настройки параметров шифрования базы документов. Возможно использование базы без шифрования либо с тремя уровнями шифрования. Если база документов используется без шифрования, прочитать информацию из нее можно, получив физический доступ к файлу. При использовании шифрования доступ к базе возможен только при наличии соответствующего файла с идентификационными данными пользователя (*user ID*). Различают три уровня шифрования – *Simple*, *Medium*, *Strong*. Чем сильнее уровень шифрования, тем медленнее выполняется работа с документами базы.

5. Переключатель *Create full text index for search* управляет возможностью автоматического создания индекса для поиска в базе. Проиндексировать базу можно в любой момент времени.

6. Группа опций задает множество копируемой информации. Можно скопировать из существующей базы документов параметры дизайна и документы или только параметры дизайна. Также новая база может содержать копию ACL старой базы.

При создании базы документов на основе шаблона или «с нуля» следует выполнить команды *File | Database | New...* При этом появляется диалоговое окно для указания дополнительных настроек (рис. 35).

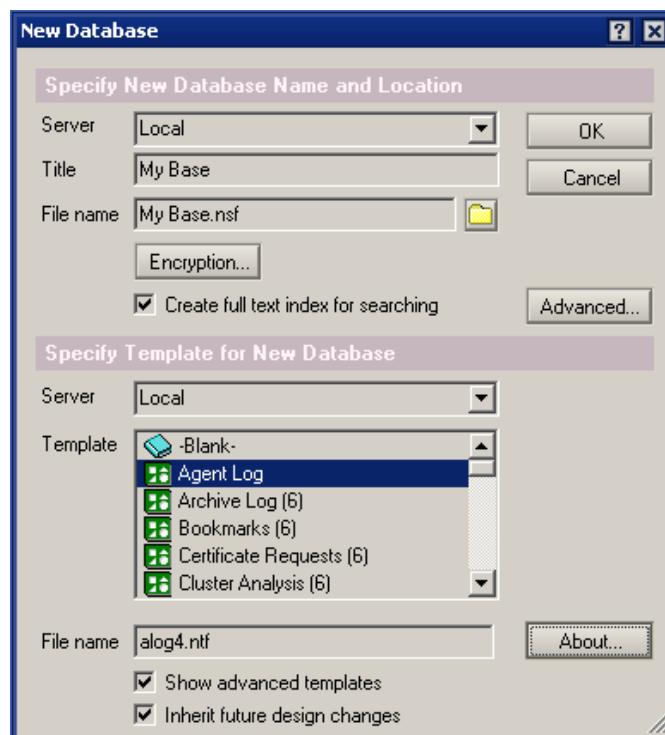


Рис. 35. Диалоговое окно для указания настроек

В случае использования шаблона он выбирается из списка Template (параметр Server – месторасположение шаблонов). Если установлен переключатель Inherit future design changes, то при изменении шаблона соответствующим образом изменится и база, использующая этот шаблон.

Рассмотрим настройки, специфичные для данного диалога. При нажатии на кнопку Advanced... возникает диалоговое окно с дополнительными параметрами базы документов (рис. 36).

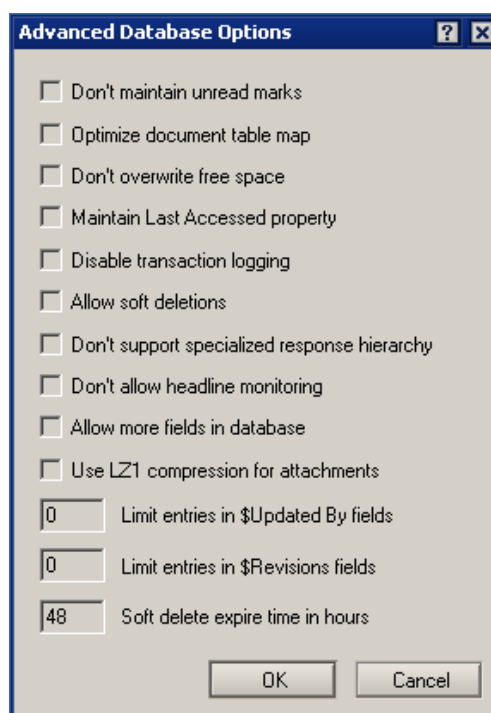


Рис. 36. Дополнительные параметры базы

Данные настройки оказывают существенное влияние на производительность при работе с базой, поэтому остановимся на них подробнее.

1. *Don't Maintain Unread Marks*. По умолчанию для каждого пользователя базы отслеживаются непрочитанные этим пользователем документы, которые помечаются особым образом. Включение опции повышает производительность. Обычно поддержка непрочитанных документов сохраняется для почтовых баз.

2. *Optimize Document Table Map*. Включение данной опции повышает производительность для баз документов, содержащих представления с условием отбора, конструируемым по полям формы.

3. *Don't Overwrite Free Space*. При удалении документов из базы образуется пустое пространство, которое затем заполняется специальными данными, чтобы исключить неавторизованный доступ к данным. Включение опции увеличивает производительность, но не рекомендуется для баз, особо критичных к безопасности.

4. *Maintain LastAccessed Property*. Если данная опция включена, специальное поле документа, содержащее время последнего доступа, обновляется автоматически при чтении или редактировании документа. Включение опции обосновано при использовании критерия архивирования документов, основанного на дате последнего доступа.

5. *Disable Transaction Logging*. Эта опция выключает поддержку транзакций для документов, что увеличивает производительность, но снижает надежность. Поддержка транзакций реально осуществляется только в случае хранения базы документов на сервере и при включенной поддержке транзакций на сервере.

6. *Allow Soft Deletions*. Включение данной опции позволяет восстанавливать случайно удаленные документы. Если опция включена, требуется задать период времени в часах, в течение которого удаленные документы можно восстановить (*Soft Delete Expire Time in Hours*). Включение данной опции подразумевает, что база будет содержать специальное представление, содержащее удаленные документы.

7. *Don't Support Specialized Response Hierarchy*. По умолчанию Notes поддерживает свойства «главный/подчиненный» для документов, что позволяет организовывать представление документов в виде иерархий. Для увеличения производительности данную поддержку можно отключить (изменения вступят в силу после сжатия базы).

8. *Don't Allow Headline Monitoring*. Lotus Notes/Domino предоставляет пользователю возможность подписаться на базу документов, что означает мониторинг и уведомление пользователя о происшедших в базе изменениях. Включение опции увеличивает производительность. Подписка на базу может быть отменена администратором на уровне сервера.

9. *Allow More Fields in Database*. Общая сумма символов имен полей базы не должна превышать 64 килобайта, что примерно эквивалентно 3000 полям. Включение опции увеличивает возможное количество полей до 23000.

10. Use LZ1 Compression for Attachments. Включение опции задеиствует для присоединенных файлов алгоритм компрессии LZ1 (по умолчанию применяется метод Хаффмана).

11. Limit Entries in \$UpdatedBy Fields. Поле \$UpdatedBy есть в любом документе и содержит список пользователей или серверов, редактировавших документ. При помощи опции можно ограничить количество значений в данном поле.

12. Limit Entries in \$Revisions Fields. Поле \$Revisions есть в любом документе и содержит список дат и времён редактирования документа. По умолчанию поле \$Revisions хранит 500 значений по 8 байт. Обычно достаточно хранения 10 последних дат.

2.7. СТРУКТУРА И СВОЙСТВА БАЗЫ ДОКУМЕНТОВ

Любая база документов в Lotus Notes\Domino представляет собой единственный файл, имеющий расширение *.NSF (Notes Storage Facility). Элементы базы документов называются *заметками (notes)*. Основные типы заметок перечислены в табл. 11.

Таблица 11

Типы заметок в базе документов

Тип заметки	Описание
Design	Набор всех форм, представлений, страниц и т. п., рассматриваемых как элементы дизайна базы
Info	Документ справки о базе About Database
Icon	Пиктограмма базы документов
Help	Документ справки Using Database
ACL	Список управления доступом для базы
Page	Средство для отображения информации, не связанной с полями документа (текст, графика, ссылка)
Form	Служит для ввода информации в поля документа и представления документа на экране
View	Список документов, программно выбираемых для отображения информации в табличном формате
Folder	Список документов, выбираемых пользователем для отображения информации в табличном формате
Outline	Метод предоставления приложению навигационной структуры
Document	Отдельный документ или запись
Item	Поле документа
Navigator	Карта изображений для навигационных целей
Frameset	Набор областей отображения, содержимое которых изменяется программно
Agent	Набор операторов или программа, которая выполняет в базе определенные действия в зависимости от событий
Shared images	Средство хранения совместно используемых файлов изображений
Shared files	Средство хранения совместно используемых полей
Shared applets	Средство хранения совместно используемых Java-апплетов
Subforms	Часть формы, внедряемая в другие различные формы базы

Shared fields	Общие поля базы или подчиненных форм
Script libraries	Средство хранения совместно используемых подпрограмм на языках LotusScript или Java
Shared actions	Программируемые кнопки, отображаемые в верхней части представлений и форм
Database script	Программируемые события, доступные на уровне базы документов

Для идентификации отдельной заметки в базе служит *универсальный идентификатор (UNID)*. Универсальный идентификатор отображается в свойствах заметки и может использоваться для программной ссылки на заметку. Заметки могут быть связаны друг с другом. Так, отдельный документ хранит ссылку на форму, которая использовалась для его создания, ссылки на свои поля и т. д.

Рассмотрим свойства базы документов. Окно свойств базы имеет семь закладок. Первая закладка служит для отображения общей информации базы (Basics) (рис. 37).

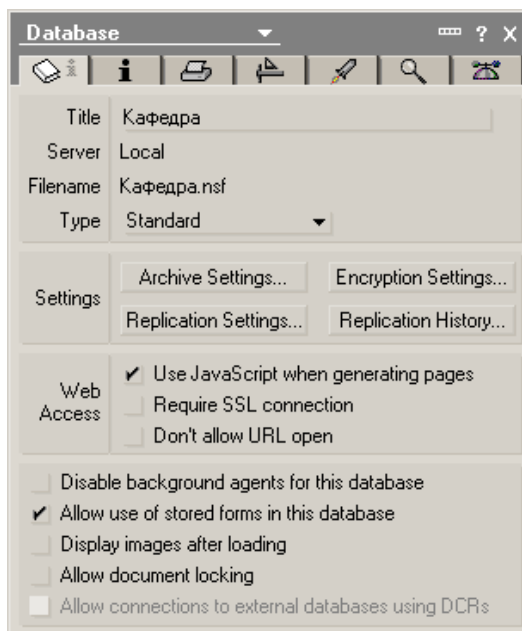


Рис. 37. Свойства базы – общая информация

Каждая база документов имеет заголовок (Title). Этот заголовок можно изменить при работе с базой, в отличие от имени файла (Filename), которое меняется на уровне ОС. Параметр Server указывает на сервер, на котором размещена база. Параметр Type определяет тип базы и связан с шаблоном, который использовался при создании базы.

Раздел Settings первой закладки свойств базы содержит кнопки для настройки параметров архивации базы (Archive Settings...), репликации (Replication Settings... и Replication History...) и шифрования (Encryption Settings...).

Раздел Web Access содержит переключатели, связанные с параметрами доступа к базе, используя Web-интерфейс. Настройка Use JavaScript позво-

ляет использовать для генерации страниц, отображающих документы, язык JavaScript. Включение настройки Require SSL connection требует использования при Web-доступе к базе защищенного соединения. Использование переключателя Don't Allow URL Open подавляет возможность открытия базы при переходе на документ базы по ссылке из другой базы или HTML-страницы.

Набор переключателей в нижней части закладки Basics позволяет контролировать следующие установки:

- Disable background agents for this database – выключение фоновых агентов для базы.
- Allow use of stored forms in this database – хранимая форма обеспечивает представление документа в том виде, в котором он редактировался посредством данной формы. Естественно, сохранение в произвольном документе кроме информации еще и формы увеличивает объем документа. Однако при пересылке такого документа на произвольный компьютер, можно быть уверенным в тождественности представления информации документа.
- Display images after loading – при включении опции все изображения, имеющиеся в документе загружаются в память и лишь затем отображаются. Таким образом, на экран выводится сначала текст документа, а затем изображения.
- Allow document locking – если данная опция включена, документ может быть заблокирован автором для избегания конфликтов репликации.
- Allow connections to external databases using DCRs – Data connection resources (DCRs) – это программные ресурсы, которые используются для подключения к внешним источникам данных.

Вторая закладка окна свойств базы документов служит для отображения дополнительной информации (Info) (рис. 38).

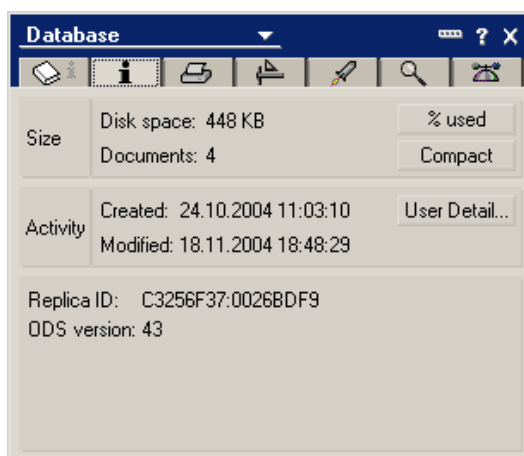


Рис. 38. Свойства базы – дополнительная информация

В разделе Size отображается занимаемое файлом базы место на носителе и количество документов в базе. Кнопка % used позволяет отобразить в процентах количество актуальных документов в базе (т.е. тех документов, которые не

помечены как удаленные). Кнопка Compact производит фактическое удаление документов, помеченных как удаленные, и сжатие файла базы.

Раздел Activity отображает даты создания и последней модификации документа. Кнопка User Detail... открывает специальное диалоговое окно, которое позволяет включить логгирование действий с базой и отображает имеющиеся записи лога.

В нижней части закладки отображается *идентификатор реплики базы* (уникален для базы, но совпадает у всех реплик одной базы) и версия *дисковой структуры* базы (On Disk Structure, ODS). ODS специфична для различных версий системы Lotus\Domino. Связь между ODS и версий Lotus\Domino представлена табл. 12.

Таблица 12

ODS для различных версий Lotus\Domino

Версия	Номер ODS
1.x	16
2.x	16
3.x	17
4.x	20
5.x	41
6.x	43

Третья закладка окна свойств базы управляет параметрами печати информации из базы (рис. 39).

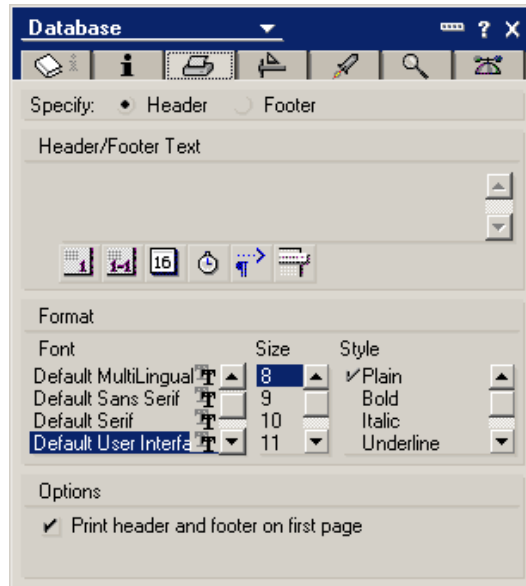


Рис. 39. Свойства базы – параметры печати

На данной закладке конструируются верхний и нижний колонтитулы, которые будут использоваться при распечатке любой информации (в частности, документа) из базы. Документ, в свою очередь, может иметь собственные колонтитулы, которые будут вложенными по отношению к колонтитулам базы.

Четвертая закладка свойств базы (Design) отображает параметры, связанные с дизайном базы документов (рис. 40).

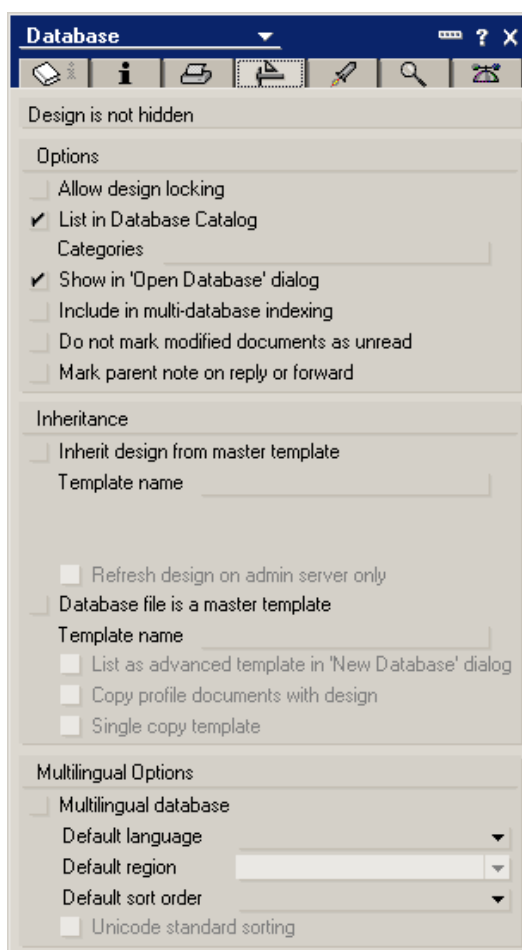


Рис. 40. Свойства базы – параметры дизайна

При помощи флага *Inherits design from master template* устанавливается отношение между шаблоном и базой, при котором все изменения, производимые в шаблоне, отображаются в дизайне базы (база *наследует* изменения в шаблоне). Флаг *Database file is a master template* превращает базу в шаблон дизайна, который может использоваться как основа для создания новых баз. Раздел *Multilingual Options* отвечает за поддержку в базе нескольких языков. Поддержка нескольких языков важна при проведении операций сортировки документов.

Закладка *Launch* определяют, что происходит при открытии базы документов (рис. 41).

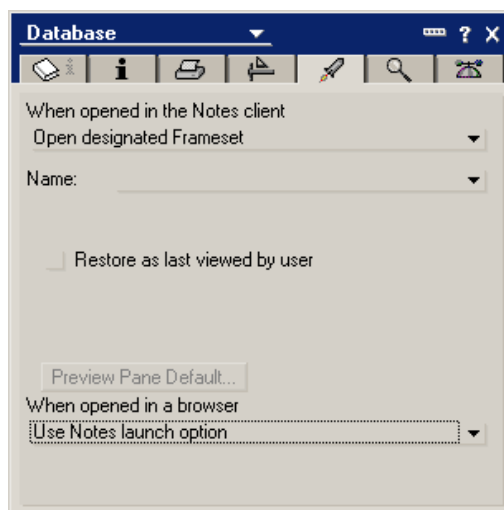


Рис. 41. Свойства базы – поведение при открытии

На закладке имеется два раздела, задающие действия, происходящие при открытии базы в Notes (When opened in the Notes client) и при открытии базы в Web-браузере (When opened in a browser). Характеристика возможных действий приведена в табл. 13.

Таблица 13

Характеристика возможных действий при открытии базы

When Opened in the Notes Client:	
Restore as last viewed by user	Установка данного переключателя позволяет открыть базу в том виде, в котором она последний раз использовалась перед закрытием
Open About database document	При открытии базы происходит показ специального документа About database с информацией о назначении базы
Open designated Frameset	Открытие указанного фреймсета
Open designated Navigator in its own window	Открытие указанного навигатора
Launch first attachment in About database	Если документ About database содержит присоединенный файл, то он запускается при открытии базы
Launch first doclink in About database	Если документ About database содержит ссылку на документ, при открытии базы происходит переход по этой ссылке (запуск ссылки)
When Opened in a Browser:	
Use Notes launch option	Установка по умолчанию, использование настройки из раздела When Opened in the Notes Client
Open About database Document	См. описание выше
Open designated Frameset	См. описание выше
Open designated Page	Открытие указанной страницы базы
Open designated Navigator in its own window	См. описание выше

Launch first doclink in About database	См. описание выше
Launch designated doclink	Запуск (переход по) указанной ссылке
Launch first document in view	Показ первого документа в указанном представлении

Следующая закладка – закладка Full-Text – содержит набор параметров и настроек, связанных с индексированием базы документов. *Индексирование* базы позволяет ускорить поиск информации в базе и предоставляет дополнительные опции поиска. Однако индекс базы занимает дисковое пространство (до 40% от общего объема базы).

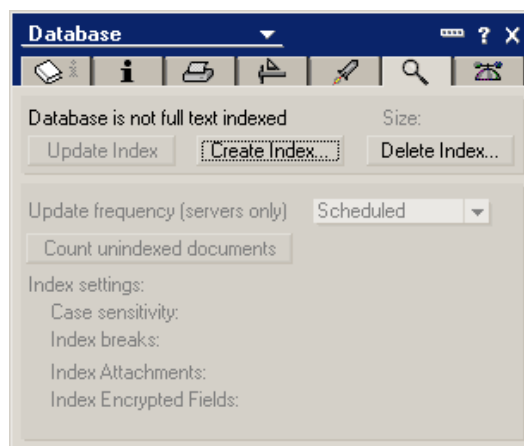


Рис. 42. Свойства базы – индексирование

Кнопки Update Index, Create Index и Delete Index служат для обновления существующего индекса, создания индекса и удаления индекса соответственно. При создании индекса отображается диалоговое окно с рядом дополнительных настроек (рис. 43).

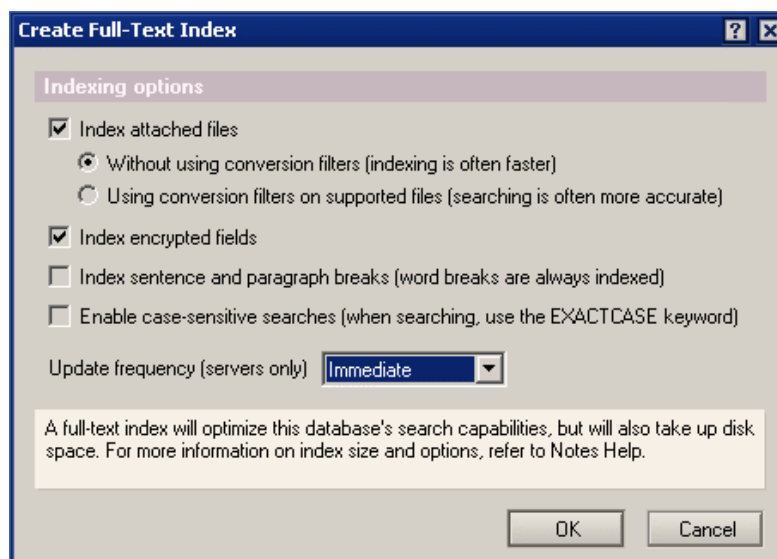


Рис. 43. Параметры индекса

Эти настройки позволяют, в частности, проиндексировать (по возможности) присоединенные к документам файлы, зашифрованные поля, создать индекс, учитывающий регистр слов.

Кнопка Count unindexed documents служит для подсчета неиндексированных документов в базе (это можно использовать для проверки целесообразности обновления индекса). Раздел Index settings отображает параметры текущего индекса.

Последняя закладка окна свойств базы – закладка Advanced (рис. 44).

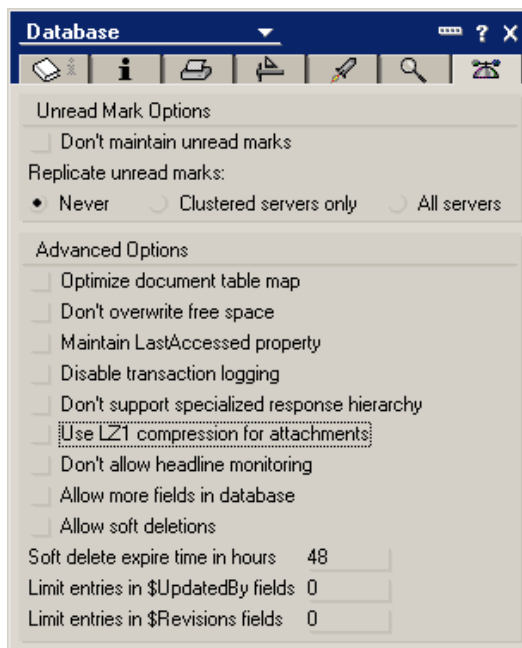


Рис. 44. Свойства базы – закладка Advanced

Параметры, настраиваемые при помощи данной закладки, практически идентичны параметрам диалогового окна Advanced, которое можно вызвать при создании базы документов.

2.8. СОЗДАНИЕ ФОРМ. СВОЙСТВА ФОРМЫ

Разберем вопросы, связанные с конструированием форм. Форма является своеобразным «окном», представляющим информацию из документа. Основной элемент формы – это поле, при помощи которого информация вводится в документ и отображается. Возможные другие элементы формы описаны в табл. 14.

Таблица 14

Элементы формы

Элемент	Описание и использование
Static text	<i>Статический текст</i> может использоваться как подпись к полю, заголовок или описание
Graphics	<i>Графика</i> используется для оформления формы, а также может отражать различные состояния документа
Table	<i>Таблицы</i> используются для выравнивания текста, графики и полей

Hotspot	<i>Активная область</i> может служить ссылкой на документ Notes или веб-страницу, организовывать всплывающие подсказки или выполнять формулы или сценарий
Button	<i>Кнопка</i> – это специальный вариант активной области, размещаемой в указанном месте формы
Action	<i>Действия</i> могут размещаться в меню и на Панели действий в виде кнопок. Использование действий доступно при работе с формами или представлениями. Действия инкапсулируют команды или код
Subform	<i>Подчиненные формы</i> – это общий ресурс базы, который отображается на форме при вызове функции, или размещается на форме постоянно. Подчиненные формы подобны обычным формам с точки зрения дизайна
Section	<i>Разделы</i> служат средством организации и представления информации
Layout region	Область компоновки – это специальные области форм, комбинирующие текст и графику
Computed text	Для установки значения <i>вычисляемого текста</i> используется произвольная формула
Image resources	<i>Ресурсы изображений</i> размещаются в базе и совместно используются ее элементами. Графика может быть представлена в форматах BMP, GIF или JPG
HTML	<i>Код HTML</i> может вводиться непосредственно в форму для отображения в браузере
Java applet	В форму могут быть внедрены <i>Java-апплеты</i>
JavaScript	<i>Сценарии JavaScript</i> могут быть внедрены непосредственно в форму, как и код HTML
Field	<i>Поля</i> – это основной элемент для ввода и отображения данных
Attachment	Форма может содержать <i>присоединенные файлы</i>
OLE object	Элементами формы могут являться <i>объекты OLE</i> (object linking and embedding)

Конструирование формы в Domino Designer выполняется как редактирование документа. Как и при обычном редактировании текста, для вставки отдельных элементов и их настройки применяются команды меню, окна свойств и кнопки на Панели инструментов. Для позиционирования отдельного элемента применяется курсор (т. е. требуемый элемент размещается в том месте формы, где находится курсор). После требуемой настройки форма сохраняется.

Рассмотрим возможности настройки отдельных свойств формы. Первая закладка окна свойств формы содержит общую информацию и параметры формы (рис. 45). Любая форма должна иметь имя (Name). Кроме имени, форма может иметь псевдоним, записываемый через вертикальную черту. Если использовать символ \ в имени формы, то в меню имя формы будет отображаться как подменю. Например, имя формы Поставщики\Информация\VInfo, означает, что форма имеет псевдоним VInfo, в меню ссылка на форму будет отображаться в подменю пункта Поставщики. Общая длина имени формы не должна превышать 255 символов.

Поле Comment содержит произвольный текстовый комментарий к форме. Поле Type задает тип формы и может принимать следующие значения: Document, Response, Response-to-Response.

Секция Display содержит набор переключателей, которые позволяют использовать имя формы как пункт указанного меню (Include in menu...), включить использование данной формы при конструировании поисковых запросов (Include in Search Builder). Опция Include in print позволяет включить использование формы при построении *компактных печатаемых страниц*, т. е. страниц, отображающих на печати содержимое нескольких форм.

Секция Versions позволяет включить отслеживание версий документов, созданных с использованием формы. Если используется контроль версий, то сохраняется каждая версия документа. Доступные в данной секции настройки позволяют указать, как формируются версии документа, и в каком виде они будут затем отображаться.

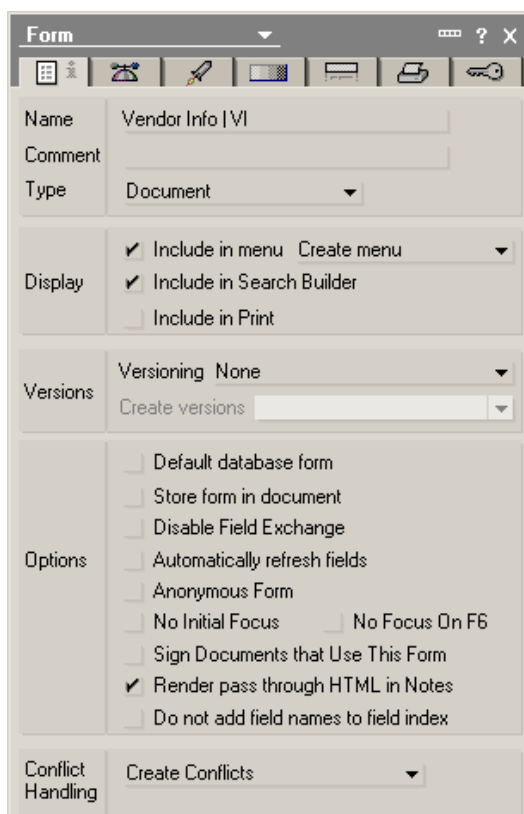


Рис. 45. Основные свойства формы

Секция Options содержит набор опций формы:

- Default Database Form – форма будет использоваться для отображения тех документов, у которых ссылка на форму, применявшаяся при создании, пуста.
- Store Form in Document – форма сохраняется вместе с документом. Это позволяет корректно отображать содержимое документа, например, при пересылке по электронной почте.
- Disable Field Exchange – включение опции запрещает обмен значений полей с данными из других приложений, используя технологии Notes/FX или OLE.
- Automatically Refresh Fields – включает обновление всех полей формы при изменении хотя бы одного поля.

- Anonymous Form – при сохранении документа не сохраняется информация об авторе. По умолчанию информация сохраняется в специальном внутреннем поле документа.
- No Initial Focus – при открытии документа в отдельном элементе фреймсета этот элемент не получает фокус.
- No Focus on F6 – запрещает получение фокуса формой при нажатии F6 или Shift+F6. Данные клавиши используются для перемещения фокуса в фреймсетах.
- Sign Documents That Use This Form – при сохранении документа используется цифровая подпись.
- Render Pass-Thru HTML in Notes – если включена данная опция, документ отображается в клиенте Notes так, как он отображается при использовании браузера.
- Do Not Add Field Names to Field Index – по умолчанию имена полей формы сохраняются в специальной таблице (Field Index), что позволяет отображать их в различных списках выбора. Включение опции запрещает сохранение имен полей.

Секция Conflict Handling управляет поведением документов при возникновении конфликтов. *Конфликт* – это ситуация, при которой два пользователя одновременно изменяют содержимое одного документа. Все поля и формы имеют специальный встроенный счетчик версий, который позволяет выявить конфликт. Если конфликт обнаружен, возможны следующие действия:

- Create Conflicts – установка по умолчанию. При возникновении конфликта создается *конфликтный документ*, представляющий отдельную версию главного документа и отображаемый специальным образом в представлениях.
- Merge Conflicts – при возникновении конфликта документы объединяются (за исключением случая, когда конфликт выявлен в одном и том же поле; тогда создается конфликтный документ).
- Merge/No Conflicts – аналогично предыдущей опции, но конфликтный документ не создается.
- Do Not Create Conflicts – конфликт игнорируется.

Следующая закладка окна свойств формы содержит настройки, связанные с событиями формы On Create, On Open, On Close, On Web Access.

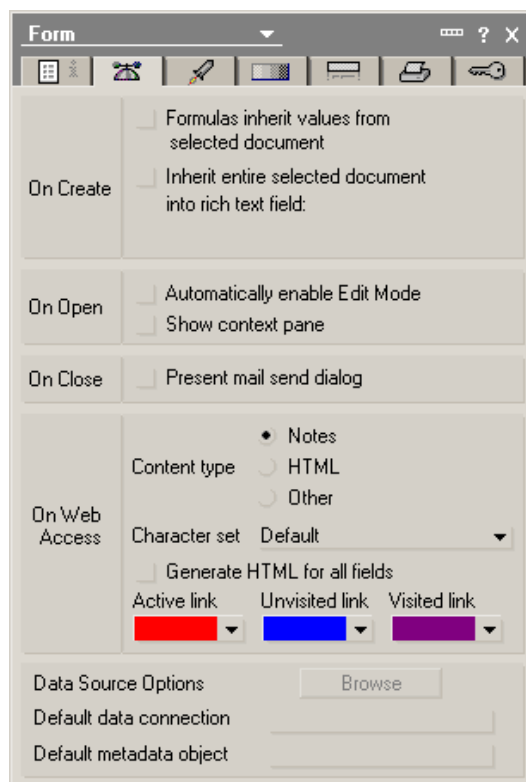


Рис. 46. Настройки событий формы

Секция On Create управляет параметрами наследования документов. *Наследование* происходит в единственном случае – когда создается новый документ. При этом базовый документ должен быть открыт или выделен в представлении. Существуют два вида наследования. При *наследовании на уровне полей* (Formulas inheris values from ...) происходит запись значений полей базового документа в поля документа-наследника. При этом предполагается совпадение имен полей или же значение поля по умолчанию должно быть задано как имя поля из базового документа. Второй вид наследования (Inherit entire selected document ...) – это вставка базового документа в одно из полей документа-наследника. Предполагается, что это поле документа-наследника будет иметь формат RTF. Возможна вставка содержимого базового документа или вставка ссылки на базовый документ.

Секция On Open имеет две опции. Опция Automatically Enable Edit Mode позволяет по умолчанию открывать документы в режиме редактирования (а не чтения). Опция Show Context Pane управляет отображением контекстной панели, на которой может быть выведен базовый документ или документ, на который указывает ссылка.

Опция секции On Close позволяет автоматически выводить диалоговое окно отправки почты при закрытии документа.

Секция On Web Access позволяет настроить отображение информации при доступе к документу через браузер. Параметр Character Set может быть использован при разработке многоязыковых приложений. Флаг Generate HTML for All Fields позволяет сгенерировать HTML-кода для скрытых полей формы. Также в данной секции можно выбрать цвета для ссылок.

Третья закладка опций формы позволяет осуществить автоматический запуск указных объектов формы. Условия запуска настраиваются. Также при помощи данной закладки можно выбрать фреймсет для размещения формы.

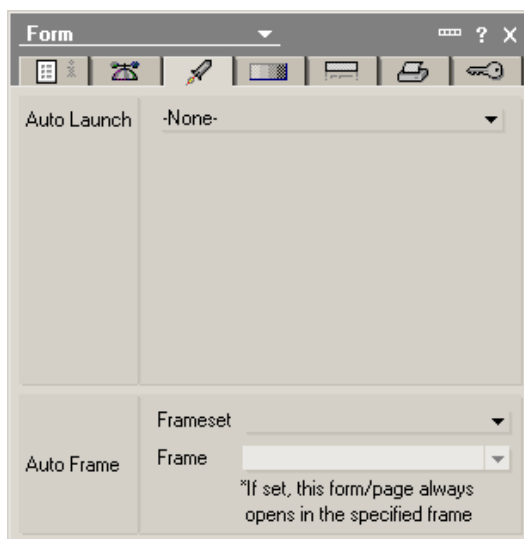


Рис. 47. Автоматический запуск объектов формы и выбор фреймсета

На четвертой закладке настраивается фоновый цвет или рисунок формы. Можно разрешить пользователям изменять данные параметры для документа во время работы в клиенте Notes (Allow users to change these properties).

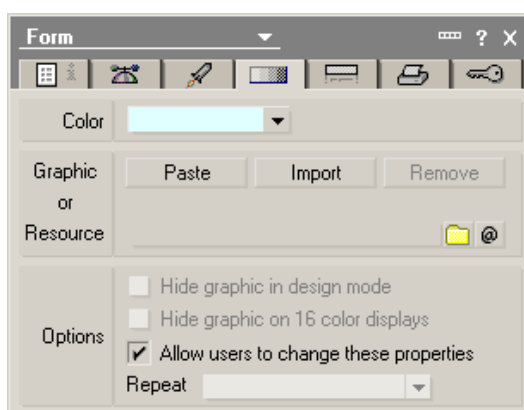


Рис. 48. Настройка цвета и рисунка формы

Пятая закладка позволяет настроить заголовок формы. *Заголовок формы* – это область в верхней части формы, которая не подвергается скроллингу при просмотре документа. Опции позволяют добавить заголовок к форме (Add header to form), задать размер заголовка (секция Size), настроить параметры границы заголовка (Border).

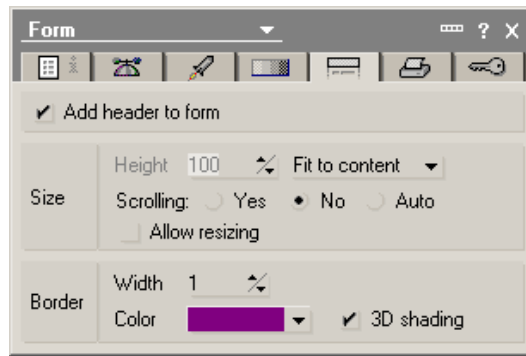


Рис. 49. Настройка заголовка формы

Шестая закладка управляет параметрами печати колонтитулов документа.

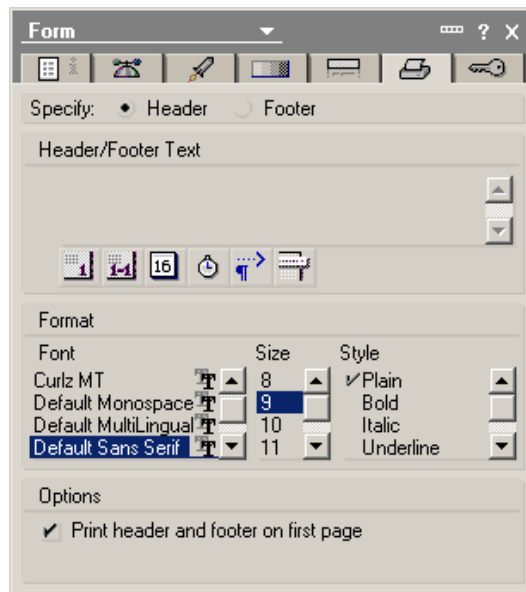


Рис. 50. Управление параметрами колонтитулов

Закладка Security содержит опции, связанные с обеспечением безопасного доступа к форме и полям.

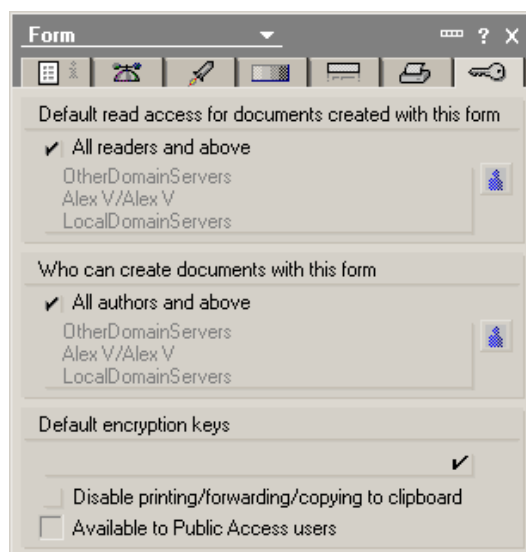


Рис. 51. Опции доступа к форме и полям

Первые два списка определяют пользователей, которые имеют право чтения (Default read access...) и создания (Who can create...) документов по данной форме. Поле Default Encryption Keys позволяет назначить *ключ шифрования* для формы. Если на форме имеются зашифрованные поля, то при сохранении документа ключ шифрования применяется к этим полям. Два переключателя в нижней части закладки позволяют запретить распечатку, пересылку и копирование информации из полей формы (Disable Printing/Forwarding/Copying to the Clipboard) и разрешить просмотр документов, созданных по форме, пользователям с правами «No access» или «Depositor» (Вкладчик) (Available to Public Access Users).

2.9. ПОЛЯ ФОРМЫ

Поля формы – это средство для ввода и отображения информации документа. В отличие от реляционных баз данных, большинство типов полей баз документов Domino позволяет хранить информацию, размер которой не зафиксирован жестко. Обычно поле хранит текст произвольной длины или список неких значений. Документы баз Domino являются неструктурированными по причине произвольного размера большинства полей, а также из-за того, что документы, даже созданные с использованием одной формы, могут иметь разный набор полей.

Настройка поля, помещенного на форму, выполняется при помощи окна свойств (рис. 52).

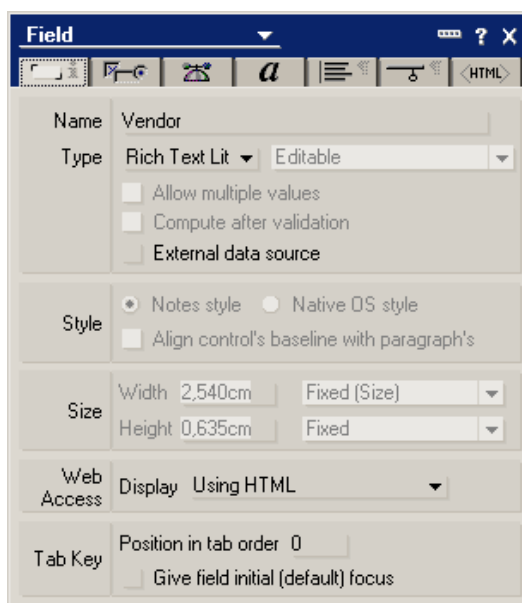


Рис. 52. Основные свойства поля

Поле должно иметь имя (Name). Имя должно быть записано без пробелов, ограничение на длину имени – 32 символа. Кроме имени, каждое поле имеет тип (Type), определяемый двумя параметрами. Один из параметров определяет, как информация заносится в поле (табл. 15).

Типы поля по способу получения информации

Название типа	Значение
Editable	<i>Редактируемое поле:</i> информация редактируется пользователем и сохраняется в документе. Поле может иметь значение по умолчанию.
Computed	<i>Вычисляемое поле:</i> значение поля определяется формулой. Поле сохраняется в документе.
Computed for display	<i>Вычисляемое поле для отображения:</i> значение поля определяется формулой. Поле не сохраняется в документе.
Computed when composed	<i>Вычисляемое при создании поле:</i> значение поля определяется формулой, когда документ создается. Поле сохраняется в документе.

Примечание: поля форматированного текста могут быть только редактируемыми или вычисляемыми.

Второй параметр типа определяет собственно тип данных поля. Он может принимать одно из следующих значений:

1. Text. Текстовое поле хранит неформатированный текст объемом до 32 КВ. Формат текста в таком поле зафиксирован разработчиком формы.

2. Date/Time. Поле данного типа предназначено для хранения даты и/или времени. При выборе этого типа для поля секция Style на закладке отображает значения Notes Style и Calendar/Time Control. При выборе стиля для отображения Notes Style значение показывается как число в поле. Если выбран стиль Calendar/Time Control, для выбора значений можно использовать специальный элемент управления. Для дополнительной настройки параметров поля используется вторая закладка окна свойств (рис. 53):

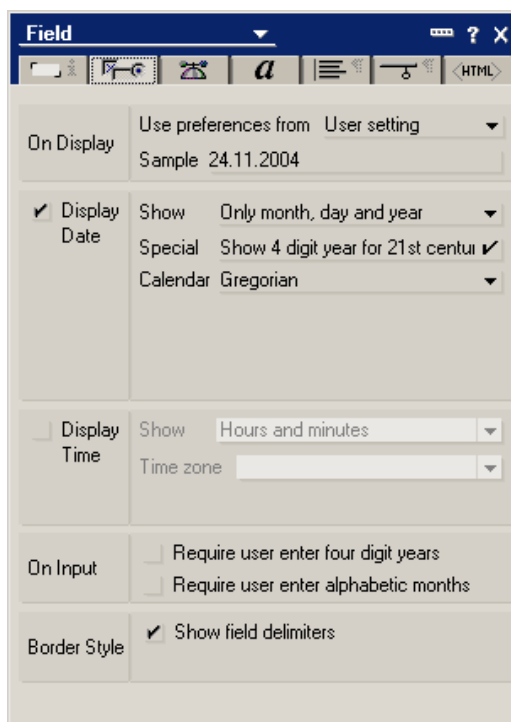


Рис. 53. Дополнительная настройка поля

3. Number . Поле для хранения чисел. Уточнение типа хранимого числа (целый, вещественный и т.п.), а также настройка параметров ввода и отображения чисел производится при помощи второй закладки окна свойств поля (рис. 54).

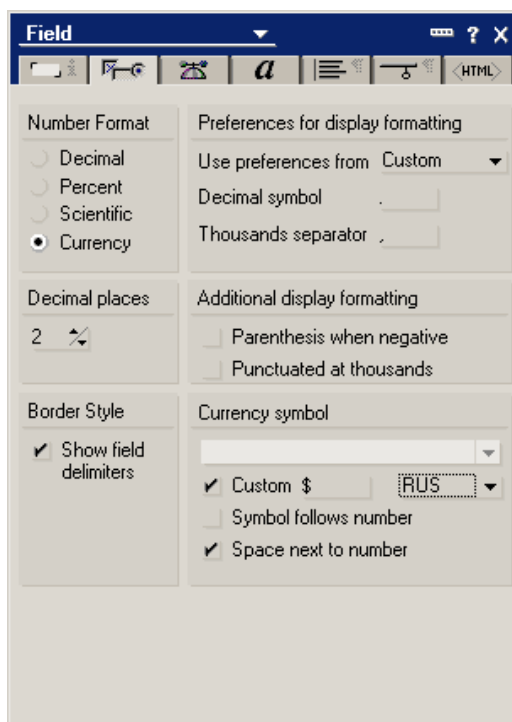


Рис. 54. Настройка числового поля

4. Dialog List, Check Box, Radio Button, List Box, Combo Box. Данные типы предназначены для работы с множеством ключевых слов. Они различаются способом выбора отдельного элемента множества. Настройка параметров этих полей происходит при помощи второй закладки окна свойств (рис. 55).

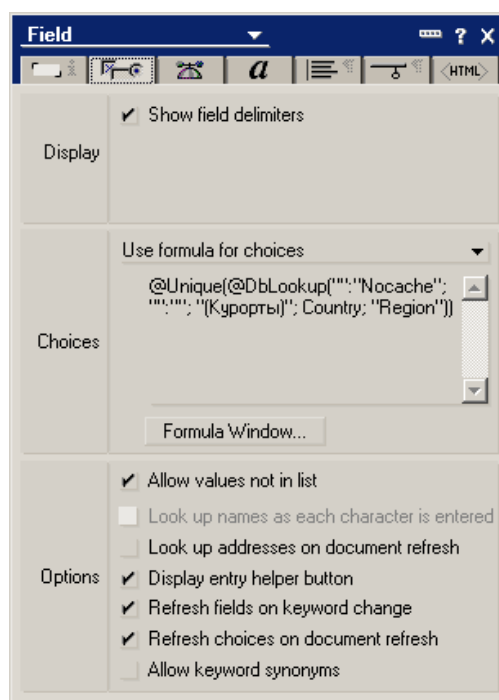


Рис. 55. Настройка поля для множества ключевых слов

Обратите внимание, что множество ключевых слов может быть получено различными способами (секция Choices) – путем явного перечисления, как результат работы формулы, как результат выборки данных из некоего представления. Опции позволяют разрешить ввод пользовательских элементов в список ключевых слов (Allow values not in list), управляют обновлением значения поля при изменении ключа (Refresh fields on keyword change) и т. п.

5. Rich Text. Поля данного типа содержат *форматированный текст* (текст в формате rtf). Форматированный текст допускает использование различных шрифтов, размеров, параметров выравнивания, а также может хранить в качестве своих элементов вложенные файлы, изображения, таблицы, вложенные документы, Java-апплеты, вычисляемый текст и другие элементы. При использовании поля данного типа следует учитывать, что эффективная работа с содержимым поля посредством встроенных формул обычно затруднена.

6. Rich Text Lite. Этот тип определяет поле форматированного типа с возможностью ограничения набора вводимых элементов. На второй закладке окна свойств поля отображается список типов элементов, которые можно поместить в поле, а также опции, задающие порядок отображения вложенных элементов.

7. Authors, Names, Readers. Поля данных типов предназначены для работы со списками имен. Поля типа Readers и Authors контролируют доступ к документу в дополнении к ACL. Поле Readers ограничивает доступ на чтение документа. Если у документа есть поле данного типа, и пользователь не прописан в списке значений поля, то он не сможет прочесть документ базы (он даже не будет знать о существовании документа, так как такой документ для него нигде не отобразится). Поля типа Authors ограничивают право на редактирование документа (сходным образом). Поля типа Names позволяют организовать выбор значений имен пользователей из списков адресной книги или ACL.

8. Password. Поле данного типа предназначено для ввода паролей – при вводе информации отображаются «звездочки». Следует иметь в виду, что подобная «защита» является чисто визуальной – значение поля не шифруется и может быть прочитано при работе или при просмотре окна свойств документа.

9. Time Zone. Поле данного типа содержит список для выбора временной зоны. Значения списка представлены как смещения относительно Гринвичского времени (GMT).

10 Color. Данное поле представляет элемент управления для выбора цвета. Цвет хранится как шестнадцатеричное целое, формат цвета – RGB.

Опция Allow multiple values разрешает запись в поле нескольких значений. В этом случае поле можно рассматривать как массив или список значений. Данная опция не доступна для следующих типов полей: Radio button, Combo box, Rich text, Password, Rich Text Lite, Time zone, Color.

Опцию Compute after validation можно установить только для вычисляемых полей. Включение опции означает, что поле будет вычислено после проверки корректности значений других полей (это полезно, если значение поля зависит от значений других полей).

Опция External Data Source указывает, что значение поля связано с внешним источником данных. При установке опции появляются дополнительные параметры для настройки этого внешнего источника данных.

Секция Style позволяет выбрать способ отображения поля – принятый в Notes или специфичный для операционной системы.

Секция Size позволяет задать размер поля и настроить параметры подгонки размера.

Секция Web access доступна для полей типа Rich Text и Rich Text Lite. Для таких полей определяется, как будет организован способ редактирования поля при доступе к форме в Web-клиенте.

Секция Tab key содержит параметры, управляющие перемещением фокуса по полям формы при нажатии клавиши Tab.

Вторая закладка окна свойств поля управляет параметрами отображения и форматом поля. Ее содержимое зависит от типа поля. Несколько возможных вариантов приведены на рисунках выше по тексту.

Третья закладка свойств поля – закладка Advanced. Она содержит несколько секций (рис. 56).

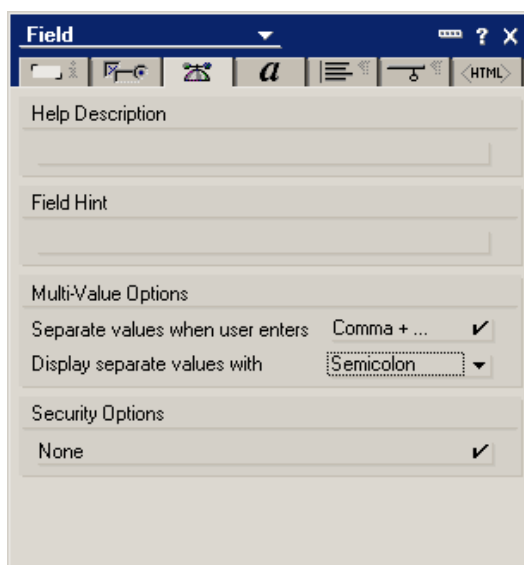


Рис. 56. Закладка Advanced

Параметр Help Description задает строку с подсказкой, которая отображается внизу документа при получении полем фокуса ввода. Параметр Field Hint – это текст, отображаемый в поле до получения им фокуса (это не значение по умолчанию, а некая подсказка).

Секция Multivalue Options содержит набор опций для полей, содержащих несколько значений. Секция Security Options позволяет установить некоторые настройки безопасности для поля (None, Sign If Mailed or Saved in Section, Enable Encryption for This Field, Must Have at Least Editor Access to Use).

Четвертая закладка свойств поля позволяет настроить параметры шрифта, которым отображаются значения поля. Эти настройки достаточно стандартны.

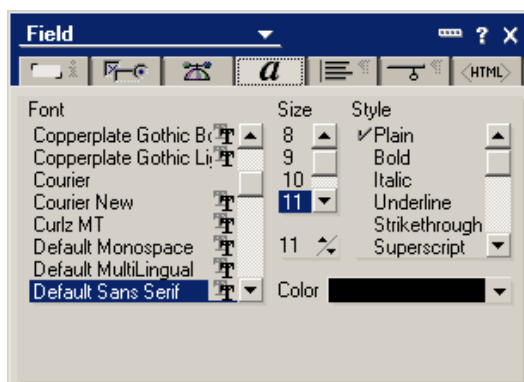


Рис. 57. Настройка параметров шрифта

Пятая закладка управляет параметрами выравнивания и отступов для параграфа, содержащего поле (рис. 58).

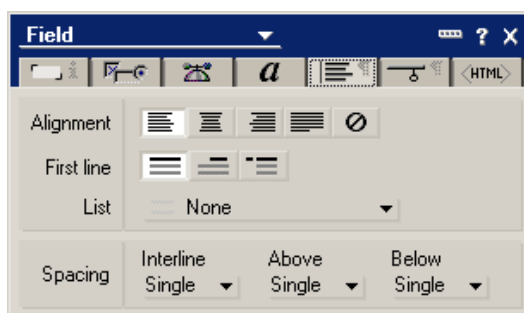


Рис. 58. Параметры выравнивания и отступов

Шестая закладка содержит набор опций, позволяющих скрыть поле при определенных условиях (рис. 59).

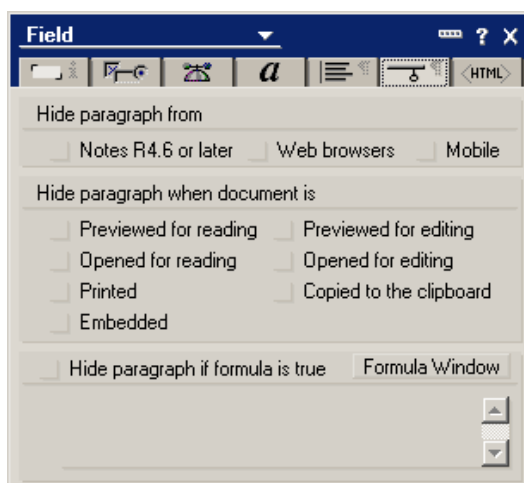


Рис. 59. Набор опций, позволяющих скрыть поле

В качестве таких условий может фигурировать формула, состояние документа (чтение, редактирование, печать), открытие документа в Web-браузере или старых версиях Notes.

Последняя закладка окна свойств управляет параметрами отображения поля в браузере (рис. 60).

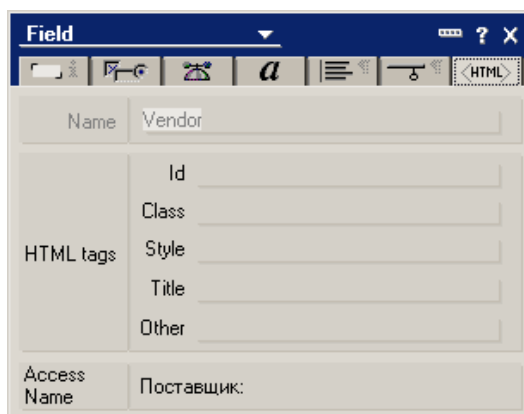


Рис. 60. Параметры отображения поля в браузере

Секция HTML Tags позволяет определить для поля несколько HTML атрибутов. Параметр ID используется для идентификации поля в JavaScript или Cascading Style Sheet (CSS). Параметры Class и Style используются в CSS. Текст, заданный параметром Title, появляется как подсказка при наведении курсора мыши на поле (при просмотре в браузере). При необходимости другие HTML-теги можно вводить в поле Other.

2.10. СОЗДАНИЕ ПРЕДСТАВЛЕНИЙ И ПАПОК

Представления, иначе называемые *видами* (View) – один из основных элементов любой базы документов. Представление – это множество документов, отобранных и упорядоченных согласно определенным условиям. В отличие от выборок из реляционных баз, которые для больших наборов данных могут занимать существенное время, связь документа с представлением обычно устанавливается в момент создания документа. Таким образом, даже для больших баз, содержащих тысячи документов, работа с представлением (обновление, сортировка) не занимает много времени.

Опишем основные этапы создания представления. Прежде все заметим, что создать представление может обычный пользователь готовой базы в произвольный момент времени. Для этого пользователь должен, конечно же, иметь определенный уровень доступа к базе. Полноценную настройку свойств представления можно произвести только в дизайнерах баз.

Для создания представления необходимо выполнить команду Create | View... или Create | Design | View... (при работе в дизайнерах). При этом появится диалоговое окно для основных параметров представления (рис. 61).

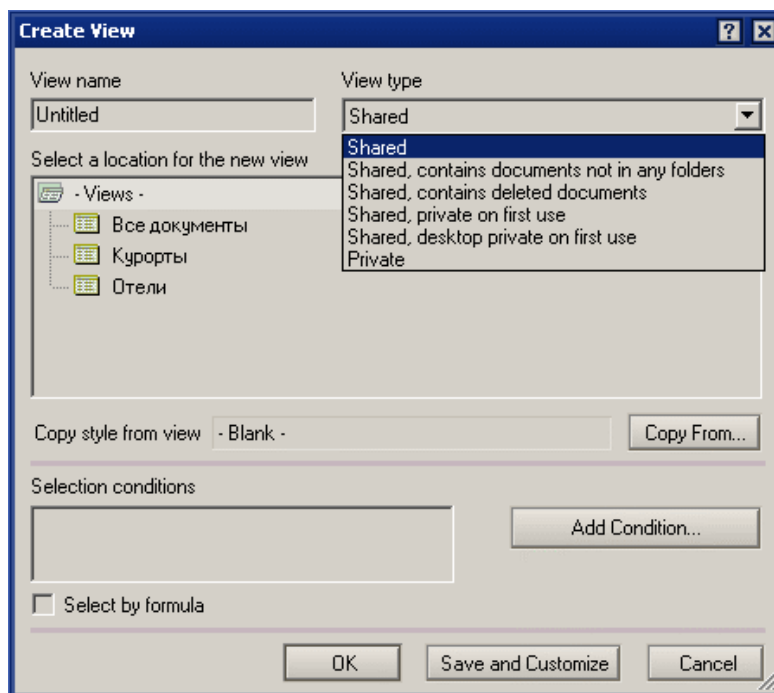


Рис. 61. Настройка основных параметров представления

Каждое представление должно иметь имя (View name). На имя представления не налагается специальных ограничений, однако, если использовать имя, заключенное в круглые скобки, получится *скрытое представление*. Скрытые представления не видны в списках представлений, но могут использоваться в формулах выбора значений.

Для представления также требуется указать тип (View type). Возможны следующие типы:

1. Shared – общее представление. Разумеется, в настройках доступа можно отрегулировать группы и роли, которые имеют или не имеют доступ к представлению. Кроме того, в зависимости от прав доступа, разные люди будут видеть в представлении разное количество и список документов. Однако если в таком представлении используются колонки для подсчета, например суммарных значений по различным категориям, то независимо от прав доступа человек будет видеть актуальные суммы **всех** документов по категории.

2. Shared, contains documents not in any folder – представление, которое на множестве документов, заданных формулой выбора, определяет подмножество тех документов, которые не лежат ни в одной из папок.

3. Shared, contains deleted documents – в представление попадают документы, которые были удалены. Эти документы из представления можно восстановить (при условии, что в базе разрешено Soft deletion).

4. Shared, private on first use – это общее представление, которое становится «приватным» (то есть видимым и принадлежащим только конкретному пользователю), после того как пользователь первый раз откроет его. При первом открытии создается копия представления, принадлежащая пользователю. Такие представления удобно использовать, например, для показа документов, созданных пользователем (формула выбора будет содержать условие Au-

thor=@UserName). Место хранения приватного варианта представления определяется самим пользователем при создании.

5. Shared, desktop private on first use – аналог предыдущего типа, только представление будет храниться на десктопе пользователя (в файле desktop.dsk). Для удаления представления (операция, необходимая для получения новой версии дизайна представления) необходимо удалить с рабочей области иконку базы, подтвердив удаление приватных представлений, и открыть базу документов заново.

6. Private – это представление, доступное только самому пользователю. Он сам его создает, если ему не хватает стандартных представлений базы. При наличии прав в ACL на создание приватных представлений такое представление будет храниться в базе на сервере, если прав нет – то в файле desktop.dsk.

Поле диалога Select a location for the new view позволяет выбрать расположение представления в иерархии представлений.

Если это необходимо, то новое представление можно создать на основе имеющегося (Copy style from view, кнопка Copy From).

Поле Select Condition и кнопка Add Condition... позволяют определить условие отбора документов в представление. При нажатии на кнопку появляется диалоговое окно, изображенное на рис. 62.

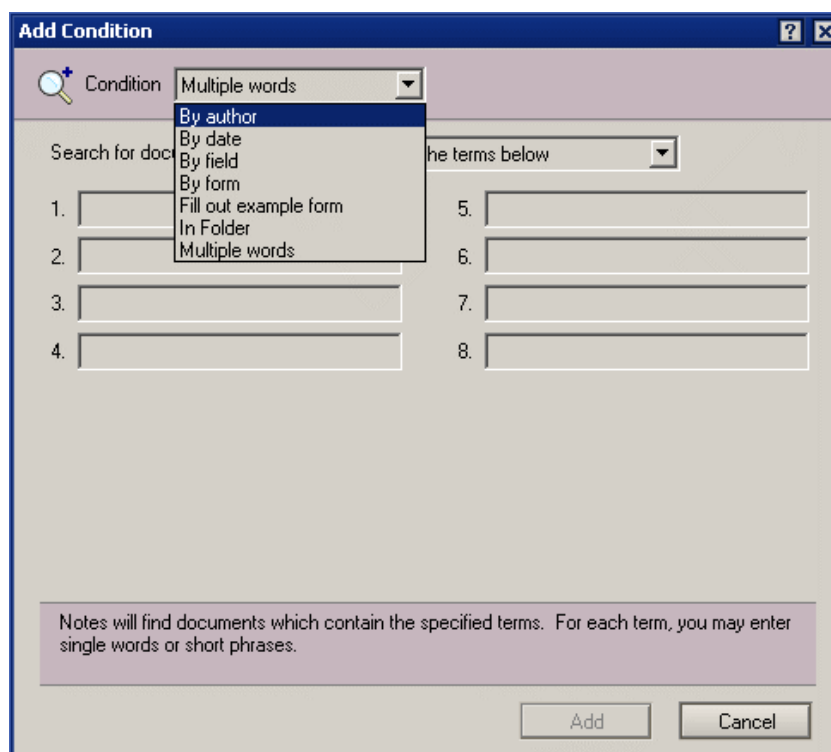


Рис. 62. Определение условий отбора документов

Можно задать одно из следующих условий отбора:

- By author – по автору документа (является автором/не является),
- By date – по дате создания или модификации документа,
- By field – по произвольному полю любой формы (содержит значение/не содержит),

- By form – по форме, использовавшейся для создания документа,
- Fill out example form – по форме, выбрав форму и заполнив некоторые ее поля,
- In folder – документы, которые содержатся в определенных папках или видах,
- Multiple words – документы, содержащие определенные слова.

Если установить флаг *Select by formula*, то для выбора документов в представлении требуется ввести @-функцию с ключевым словом *SELECT*. Например, следующее условие

```
SELECT @All
```

выбирает в представлении все документы.

После задания параметров представления следующим этапом является определения и настройка *колонок представления*. Окно для настройки колонок представления показано на рис. 63.

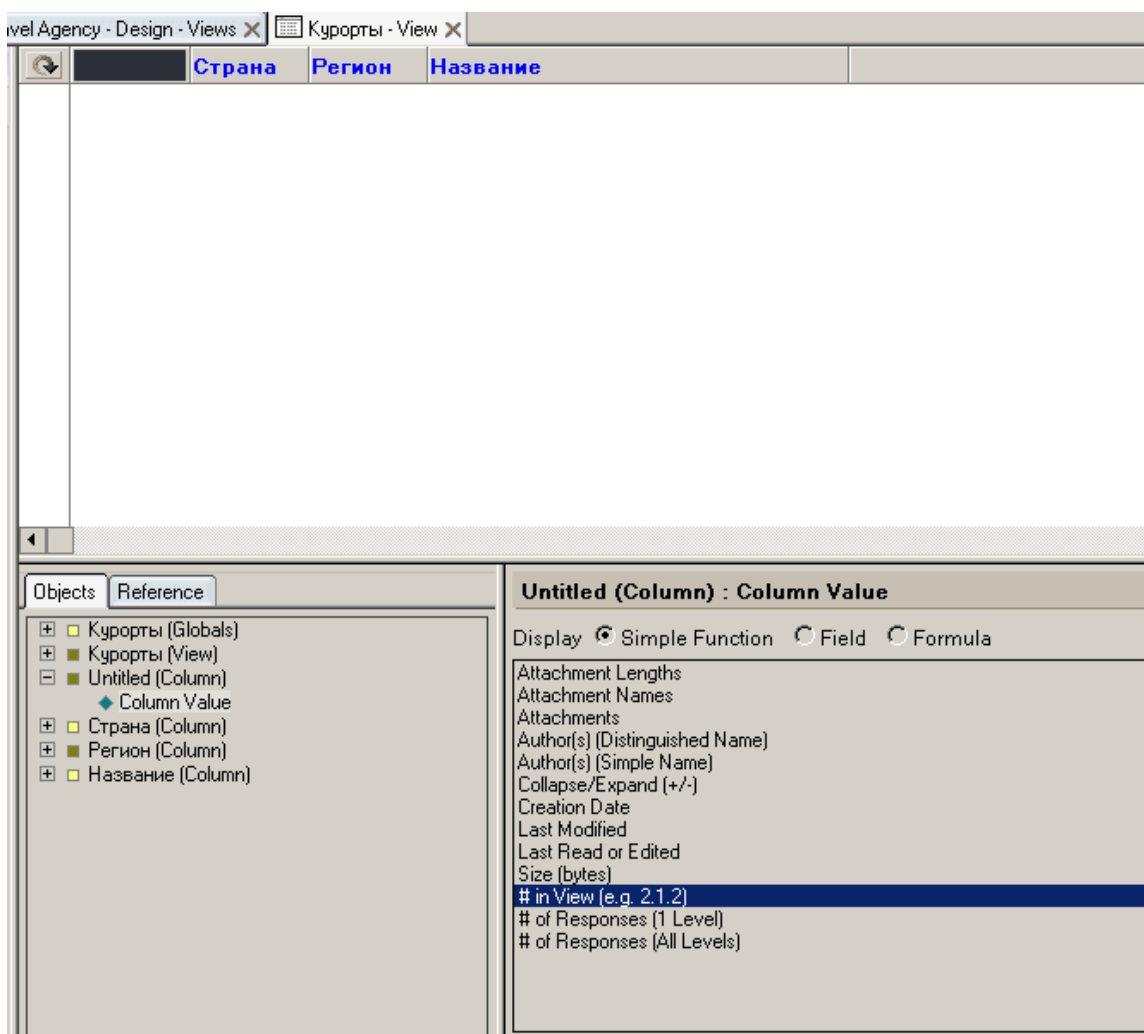


Рис. 63. Окно для настройки колонок

В верхней части окна находится список имеющихся в представлении колонок. Колонки можно перемещать, используя мышь (для изменения порядка

следования). Для создания новой колонки следует использовать контекстное меню в верхней части окна.

В нижней части окна определяется то, что будет отображаться в колонке. Возможно задание содержимого колонки как значения определенного поля (переключатель *Field*), как результата вычисления @-функции (переключатель *Formula*), либо как результат работы определенной простейшей функции (переключатель *Simple Function*).

После определения состава колонок можно настроить их свойства и свойства представления. Первая закладка окна свойств колонки – закладка *Info* общей информации.

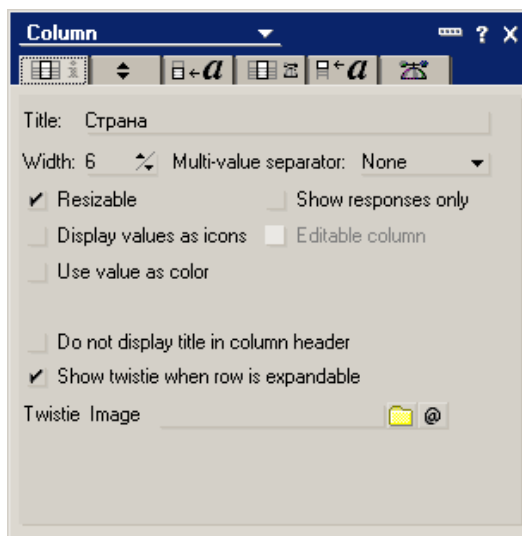


Рис. 64. Свойства колонки – общая информация

Описание значений установок данной закладки приведено в табл. 16.

Таблица 16

Параметры и описание установок

Параметр	Описание
Title	Заголовок колонки, по умолчанию – #.
Width	Ширина колонки (в символах).
Multivalue Separator	Разделитель, если в колонке помещено поле с множеством значений (None, Space, Comma, Semicolon, New Line).
Resizable	Позволяет изменять размер колонки пользователям.
Show Responses Only	В колонке отображаются только документы типа <i>response</i> (ответ) и <i>response-to-response</i> (ответ на ответ).
Display Values as Icon	Отображает в колонке значки вместо значений.
Editable Column	Если включен флаг представления <i>Allow Customization</i> , то данный параметр позволяет редактировать поле, соответствующее колонке, прямо в представлении.
Use Value as Color	Позволяет программно контролировать текст шрифта и фона колонки.
Do not display title in column header	Запрещает отображение заголовка колонки

Show Twistie When Row Is Expandable	Отображает значки в виде маленьких треугольников (Twistie), если строки представления собраны в группы (категории), которые можно развернуть.
Twistie Image	Позволяет выбрать изображение для Twistie.

Дадим некоторые комментарии к приведенным параметрам. Для разделения значений поля можно использовать переход на новую строку (New Line). Следует иметь в виду, что количество строк в одном ряду представления ограничено девятью. Для того чтобы использовать в колонке значки вместо значений, содержимое колонки должно быть задано как результат вычисления @-функции, которая возвращает номер отображаемого значка. Номер значка и изображение значка содержатся в табл. 17.

Таблица 17

Номер и изображение значка

	0	10	20	30	40	50	60	70	80	90	100	110	120	130	140	150	160	170	180
0																			
1																			
2																			
3																			
4																			
5																			
6																			
7																			
8																			
9																			

Вторая закладка окна свойств колонки – закладка Sorting, управляющая сортировкой и группировкой значений в колонке (рис. 65).

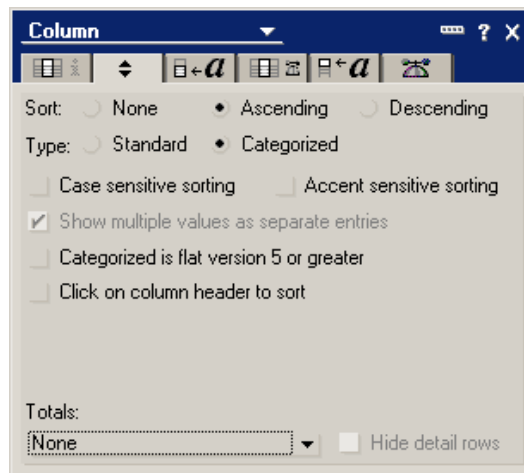


Рис. 65. Свойства колонки – сортировка и группировка

Первая группа параметров определяет сортировку значений в колонке (Sort). Тип сортировки Categorized задает группировку документов в колонке. Это позволит компактно отобразить подмножество документов с одинаковым значением некоторого поля. Остальные параметры управляют сортировкой: будет ли она регистрозависимой (Case sensitive sorting), разрешено ли пользователю производить сортировку щелчком по заголовку колонки (Click on column header to sort) и т. п. Параметр Totals может принимать одно из следующих значений: None, Total, Average Per Document, Average

Per Subcategory, Percent of Parent Category, Percent of All Documents. Данный параметр позволяет вычислить и отобразить суммарные значения по колонке или группе документов (в отдельной строке).

Третья закладка окна свойств колонки позволяет настроить параметры шрифта и выравнивания текста в колонке (рис. 66).

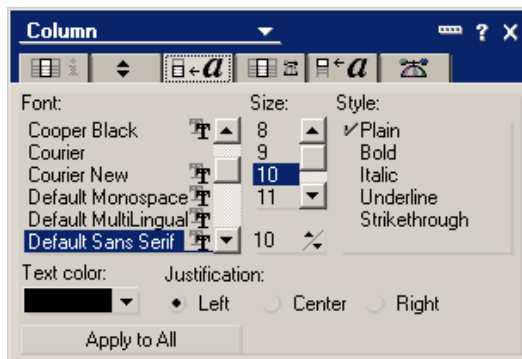


Рис. 66. Свойства колонки – настройка параметров текста

Четвертая закладка позволяет настроить параметры отображения для даты и времени, а также числовых значений, если значения именно таких типов отображаются в колонке (рис. 67).

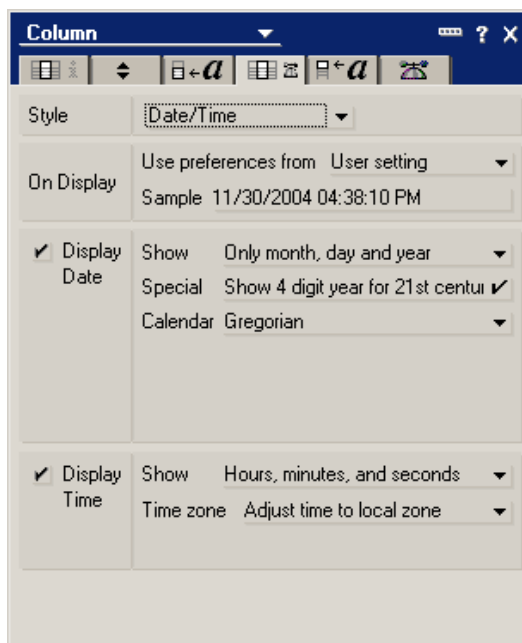


Рис. 67. Параметры отображения даты и времени в колонке

Пятая закладка управляет параметрами отображения заголовка колонки (рис. 68).

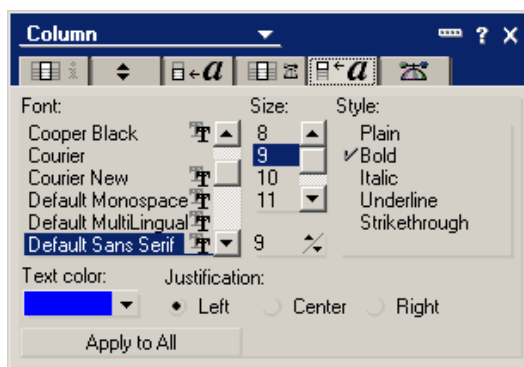


Рис. 68. Параметры отображения заголовка колонки

Последняя закладка – закладка *Advanced* – служит для настройки параметров колонки, связанной с использованием колонки в различных программных модулях и @-функциях (рис. 69).

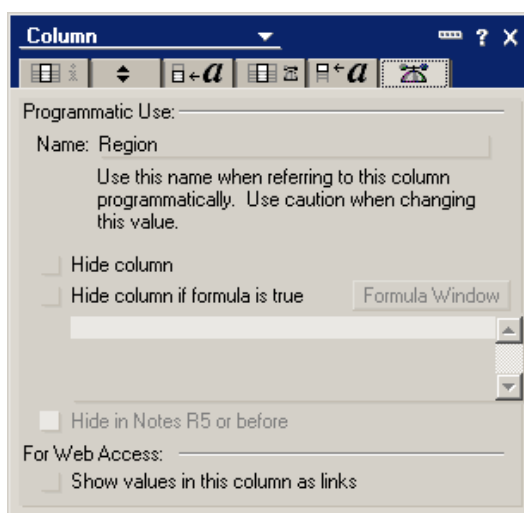


Рис. 69. Свойства колонки – закладка *Advanced*

Закладка позволяет определить имя для колонки (*Name*), позволяет скрыть колонку (безусловно или в зависимости от значения формулы), устанавливает параметры отображения колонки при *Web*-доступе.

Далее рассмотрим свойства представления, доступные для настройки. Первая закладка – это общие параметры представления (рис. 70).

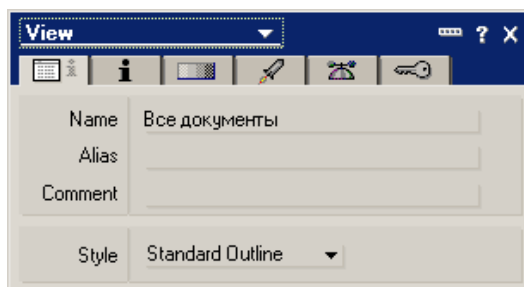


Рис. 70. Общие свойства представления

Параметр *Name* задает имя представления (максимум 64 символа). Можно также задать псевдоним (параметр *Alias*) и комментарий к представлению (параметр *Comment*). Если задать имя представления в круглых скобках, то пред-

ставление будет скрытым (исключение – представление с именем (\$All)). Параметр Style может принимать два значения: Standard Outline и Calendar. Большинство представлений создаются со стилем Standard Outline. Стилль Calendar отображает представление в виде календаря.

Вторая закладка окна свойств представления содержит набор опций представления (рис. 71).

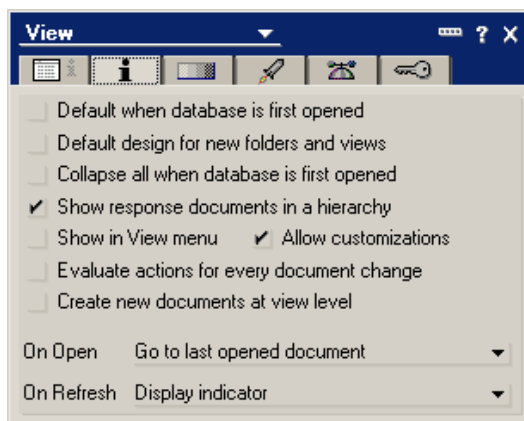


Рис. 71. Закладка опций представления

Назначение большинства опций очевидно. Опция Create New Documents at View Level позволяет редактировать и создавать документы, используя представления. Для колонки, значение которой можно редактировать, должна быть установлена опция Editable Column. В представлении для добавления документов будет отображаться специальная строка Ctrl+Click Here to Add a New Document.

Параметр On Open позволяет задать действия при открытии представления и может принимать три значения: Go to Last Opened Document (переход к последнему документу, который открывался), Go to Top Row (переход на верхнюю строку представления) и Go to Bottom Row (переход на нижнюю строку).

Многочисленное количество установок на закладке Style окна свойств представления (рис. 72) распадается на следующие категории:

- Body. Установка цвета строк и фонового рисунка для представления.
- Grid. Выбор стиля отображения и цвета сетки документов.
- Header. Настройка параметров заголовка представления (стиль, цвет, высота).
- Rows. Установка высоты строки и расстояния между строками, а также цвета для строк с неп прочитанными документами и общими значениями.
- Other. Параметры для отображения колонки выбора (левая колонка представления) а также параметр для установки расширения последней колонки представления на ширину окна.
- Margin. Установки параметров отступов для текста внутри строки представления.

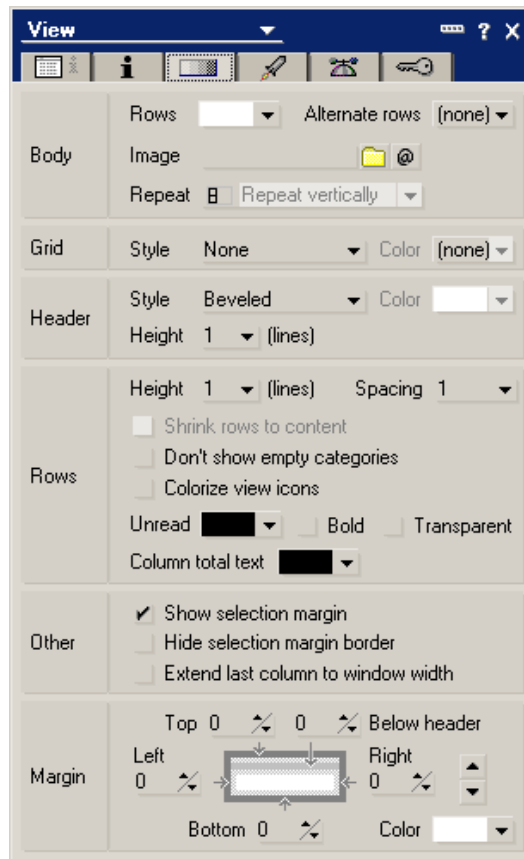


Рис. 72. Установки на закладке Style

Четвертая закладка окна свойств позволяет выбрать фреймсет и указать, в какой части фреймсета будет отображаться представление. Данная настройка имеет смысл только при Web-доступе к базе документов.

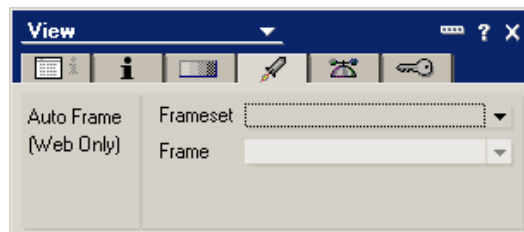


Рис. 73. Отображение представления в фреймсете

Пятая закладка предоставляет доступ к «продвинутым» опциям настройки представления (рис. 74).

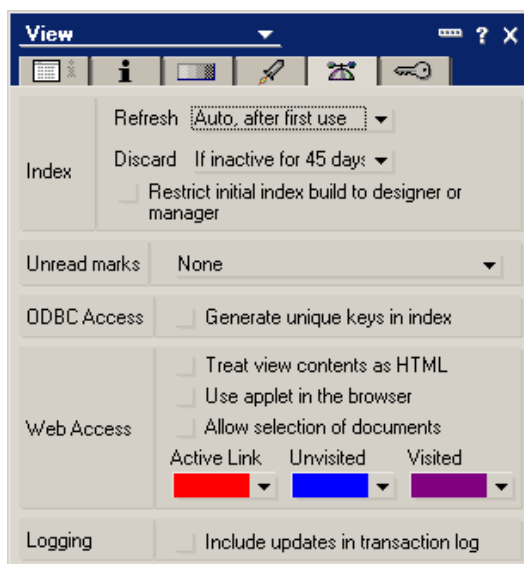


Рис. 74. Свойства представления – закладка Advanced

Закладка разделена на пять секций:

- Index. Данная секция управляет параметрами создания и обновления *индекса представления*. Индекс представления – это внутренняя структура, определяющая порядок документов. Индекс может быть обновлен вручную в любой момент времени при нажатии на клавишу F9.
- Unread Marks. Секция управляет параметрами расстановки пометок для неп прочитанных документов.
- ODBC Access.
- Web Access. Данная секция определяет поведение представления при Web-доступе.
- Logging.

Последняя закладка – закладка Security – управляет параметрами доступа к представлению (рис. 75).

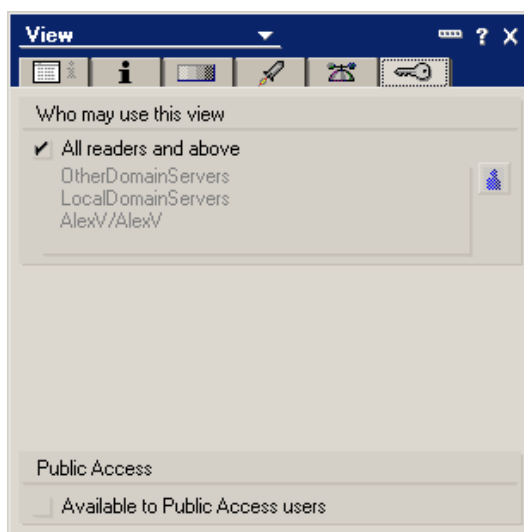


Рис. 75. Свойства представления – закладка Security

Обычно установлен флаг `All readers and above`, но можно снять флаг и выбрать пользователей, имеющих право доступа к представлению, из списка пользователей и групп.

Рассмотрим особенности, связанные с созданием папок (`folders`). *Папки* – это по сути те же представления, отличаются тем, что не имеют формулы выбора множества документов. Документы, которые будут лежать в той или иной папке, задаются самим пользователем или в бизнес-логике базы (привязывая, к какой папке будет относиться тот или иной документ при возникновении какого-либо события). Один документ может лежать в произвольном количестве папок одновременно. Физически это один и тот же документ, просто в документе в специальном атрибуте `FolderReferences` прописывается, к каким папкам он относится. Этот атрибут доступен на чтение и редактирование, например, в языке `Lotus Script`. Набор свойств папки и колонок папки аналогичен свойствам представления и колонок представления.

2.11. СОЗДАНИЕ ДЕЙСТВИЙ

Действие – это специальный элемент формы или представления, который может размещаться в меню и на *Панели действий* в виде кнопок. Действия инкапсулируют простейшие команды или код. Действие может быть запрограммировано с использованием @-функций, языков `LotusScript` или `JavaScript`.

В режиме проектирования можно просматривать действия в *Панели действий*. Для того чтобы отобразить панель, выполните команду `View | Action Pane`. Для добавления нового действия выполняется команда меню `Create | Action` (эта команда откроет *Панель действие*, если та не отображалась).

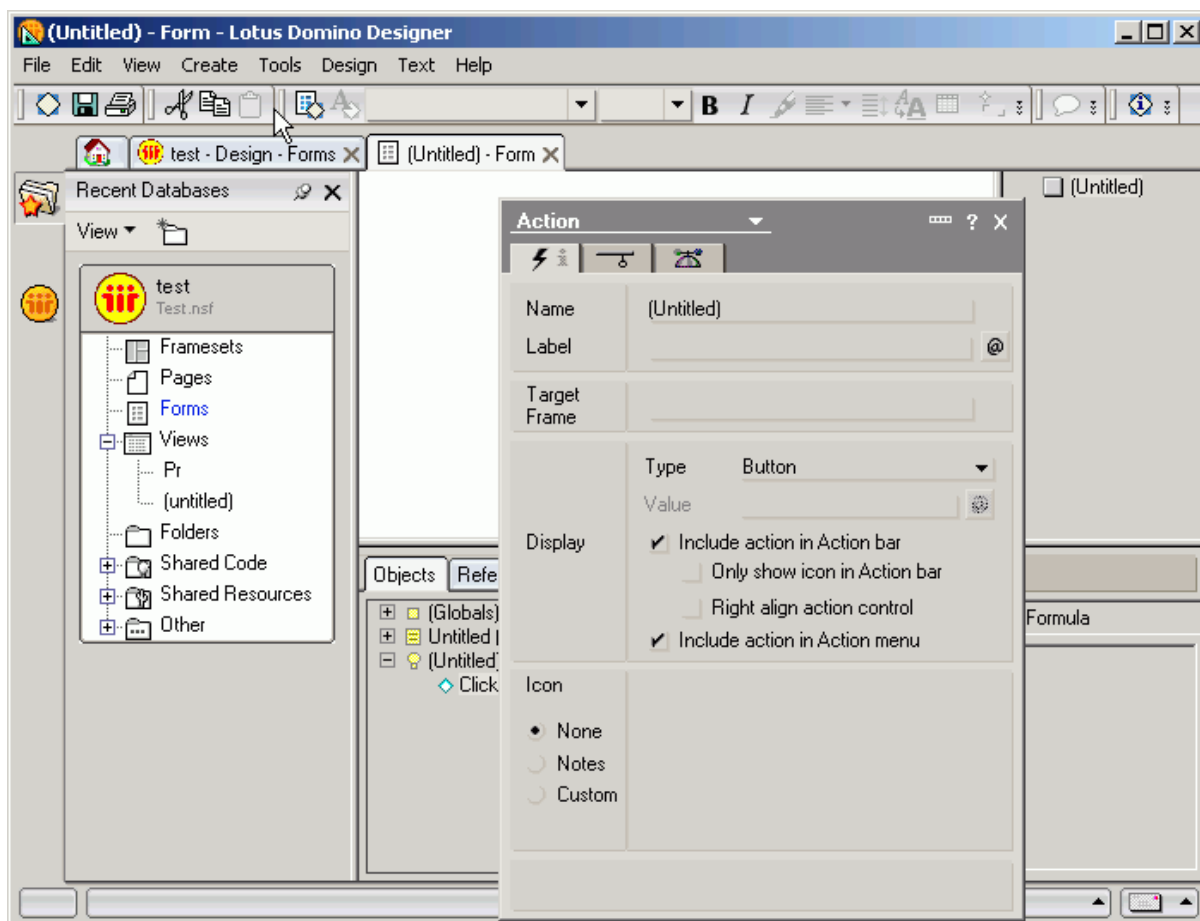


Рис. 76. Панель действий на форме

Каждая форма имеет шесть *стандартных* (Default) действий. В ранних версиях Lotus эти действия нельзя было удалить, они всегда отображались в меню Action при открытии формы в базе документов. В Lotus Notes 6 стандартные действия явно добавляются на форму командой Create | Actions | Insert System Actions. После добавления они могут быть удалены (достаточно выбрать действие на Панели действий и нажать Del). Стандартные действия перечислены в табл. 18.

Таблица 18

Стандартные действия и их применение

Действие	Применение
Categorize	Если форма имеет поле с именем Categories, это дает возможность пользователю изменять данное поле в одном или нескольких документах. Разработчики баз могут сортировать и распределять по категориям документы в представлениях
Edit Document	Открывает документ в режиме редактирования
Send Document	Отправляет документ пользователю или пересылает по почте базу документов. Для этого должно быть заполнено поле документа SendTo
Forward	Пересылает документ или базу в почтовом сообщении
Move to Folder	Перемещает документы в папку
Remove from Folder	Удаляет документы из папки (не удаляет документ)

Особый вид действий – *простые действия*² (Simple Actions). Эти действия не программируются, а выбираются из списка готовых. Ниже приведен список простых действий:

- Copy to Database (Копировать в базу документов)
- Copy to Folder (Копировать в папку)
- Delete from Database (Удалить из базы)
- Mark Document Read (Пометить документ как прочитанный)
- Mark Document Unread (Пометить документ как непрочитанный)
- Modify Field (Изменить поле)
- Modify Fields by Form (Изменить поля в форме)
- Move to Folder (Поместить в папку)
- Remove from Folder (Удалить из папки)
- Reply to Sender (Ответить отправителю)
- Run Agent (Запустить агент)
- Send Document (Отправить документ)
- Send Mail Message (Отправить почтовое сообщение)
- Send Newsletter Summary (Отправить сводку новостей)
- @Function Formula (Функция языка формул)

Возможно связывание нескольких простых действий в одном действии. Например, можно отправить документ по почте, а затем удалить его из базы.

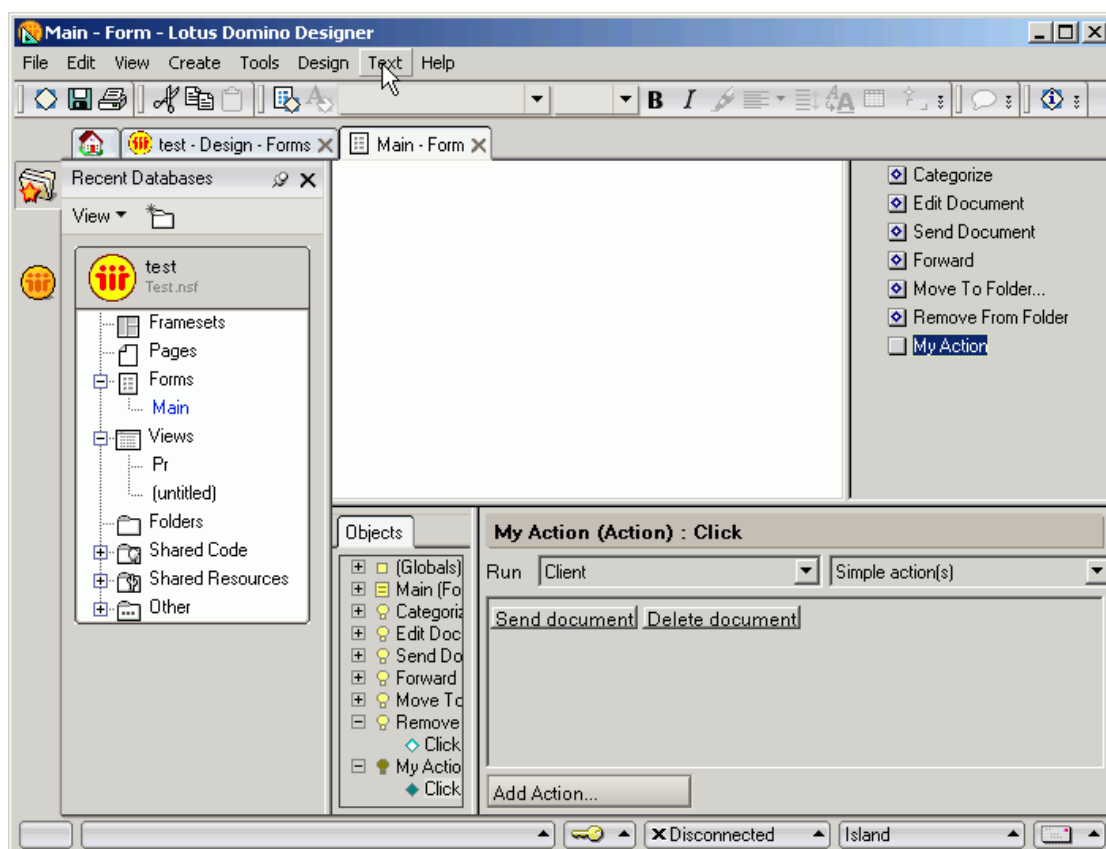


Рис. 77. Связывание нескольких простых действий в одном действии

² Простые действия и стандартные действия не работают в Web.

Как уже упоминалось в начале параграфа, если действия нельзя свести к стандартным или Simple Action, то его можно запрограммировать. Программирование сводится к написанию обработчиков событий действия, главное из которых (и единственно доступное при работе с @-функциями) – onClick.

Для действий доступно окно свойств. Первая закладка окна свойств – Action Info (рис. 78).

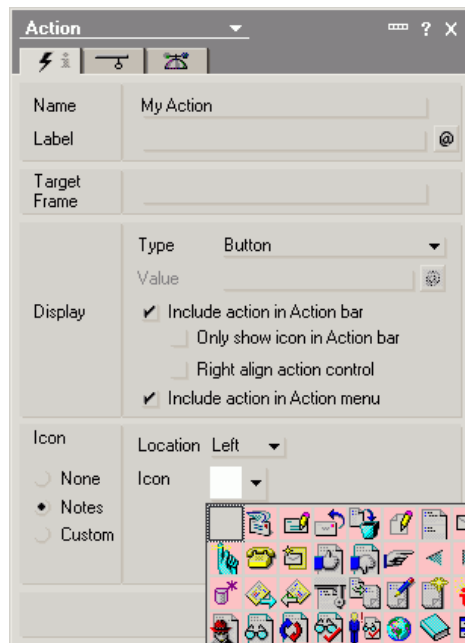


Рис. 78. Окно свойств для действий

На закладке указывается имя действия (Name) и фрейм (Target Frame), в котором показывается действие. Раздел Display управляет видом отображения действия. Можно указать, что действие должно отображаться на Панели действий, в меню или в обоих местах. Если установлено отображение действия в Панели, то можно включить показ действия в виде пиктограммы (Only show icon in Action bar) и выравнивание кнопки действия вправо (Right align action control). В разделе Icon выбирается пиктограмма для действия (фрагмент набора стандартных пиктограмм есть на рис. 78).

Вторая закладка окна свойств действия – закладка Hide-When (рис. 79).

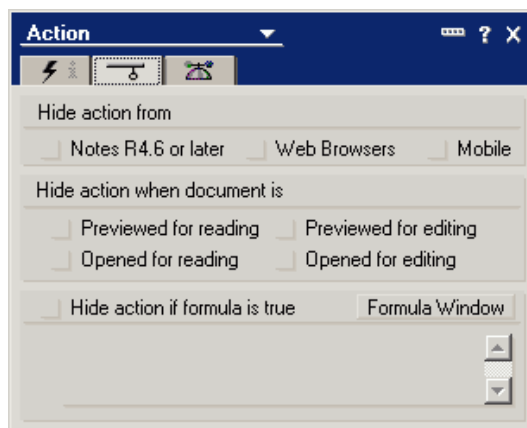


Рис. 79. Закладка Hide-When

При помощи закладки Hide-When настраиваются условия отображения действия. Например, не имеет смысла отображать действие Edit, когда документ уже находится в режиме редактирования. Можно скрывать кнопки от пользователей в зависимости от их роли в базе документов. Для этого используется последняя часть закладки (Hide action if formula is true). Например, чтобы скрыть действие Review (Просмотр) от всех пользователей, кроме входящих в группу Reviewer, можно применить такую формулу:

```
(@IsNewDoc | !@Contains(@UserRoles; "Reviewer"))
```

Последняя закладка свойств действия – закладка Advanced (рис. 80).

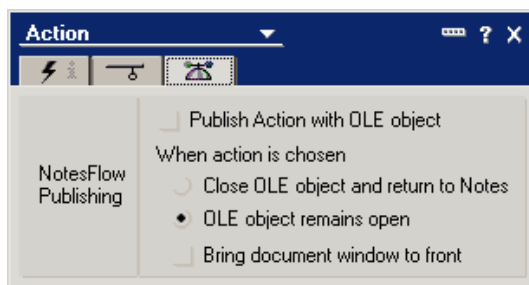


Рис. 80. Свойства действия – закладка Advanced

Параметры данной закладки необходимы при работе в Lotus с приложениями, поддерживающими технологию OLE.

2.12. ЯЗЫК ФОРМУЛ

Для разработчика баз документов возможно использование нескольких средств программирования. Это *формулы*, написанные с использованием @-команд и @-функций, *языки LotusScript, JavaScript и Java*. Формулы являются простейшим средством программирования. Допустимо использование формул в некоторых обработчиках событий формы, кнопки, документа, при проверке и вычислении значения полей.

Рассмотрим общие вопросы синтаксиса формул, построенных с помощью @-функций и @-команд. Прежде всего, отметим следующую разницу между @-функциями и @-командами. @-функции всегда возвращают некий результат. Например, вызов

```
@Name([CN]; @UserName)
```

вернет как результат строку, представляющую одну из частей имени пользователя. @-команда – это средство для программной работы с интерфейсом пользователя. Обычно @-команде может быть сопоставлена некая команда из меню Lotus Notes. Например, @Command([EditDocument]) выполняет открытие документа в режиме редактирования.

Формула состоит из *логических строк*. Каждая строка может быть: комментарием, вызовом @-функции или @-команды, оператором присваивания значения переменной либо полю. Логическая строка завершается точкой с запятой (;) (можно не ставить точку с запятой в конце последней строки). Для

удобства восприятия логическая строка может быть разделена на несколько физических строк. Регистр символов при наборе строки не имеет значения.

Для вызова @-функции применяется следующий синтаксис:

```
@Function(arguments)
```

Здесь @Function – имя функции, arguments – список аргументов, разделенных точкой с запятой. Если у функции нет аргументов, достаточно указать ее имя.

Синтаксис вызова @-команд похож на синтаксис вызова @-функции:

```
@Command([keyword]; arguments)
```

Здесь @Command – ключевое слово (не имя), которое указывает на то, что выполняется команда, keyword – имя команды, заключается в квадратные скобки, arguments – список аргументов (возможно пустой), разделенных точкой с запятой.

При записи формул допустимо использование следующих *ключевых слов*:

1. REM. Данное ключевое слово вводит комментарий, которым является текст после ключевого слова. Например:

```
REM "Это комментарий, вставленный в формулу";  
REM {Комментарий можно записать в фигурных скобках};
```

2. SELECT. Это ключевое слово служит для выбора множества документов и применяется при построении представлений. После ключевого слова следует формула, которая определяет выбираемые документы. Например:

```
SELECT Form = "Main Topic" | @Alldescendants
```

Данная формула выберет все документы, созданные с использованием формы "Main Topic", и всех *наследников* («ответы» и «ответы-на-ответы») этих документов.

3. DEFAULT. Синтаксис использования этого ключевого слова

```
DEFAULT varname := value
```

Если в документе нет поля с именем varname, то переменная varname получает значение value. Если документ содержит поле varname, то будет использоваться значение этого поля.

4. FIELD. Как и для DEFAULT, синтаксис использования

```
FIELD varname := value
```

Если в документе нет поля с именем varname, то такое поле создается и получает значение value. Если документ содержит поле varname, то поле получает новое значение, которое будет использоваться в дальнейшем.

5. ENVIRONMENT. Данное ключевое слово предназначено для работы с *переменными окружения* в файле Notes.ini. Используется в форме ENVIRONMENT varname := value, где varname – имя переменной окружения (если такой переменной нет, она создается), value – значение этой переменной.

Если при работе формулы необходимо использовать литералы, то они формируются по следующим правилам. Числовой литерал записывается как обычное число, строковый литерал записывается в кавычках, временные литералы записываются в квадратных скобках. Для организации *списков* из литералов применяется двоеточие. Например:

```
nNumber := 10.34
cText := "This is a text constant"
cTextList := "Red" : "Yellow" : "Green"
dDate := [08/09/98]
dDateTime := [08/09/98 11:30 PM]
```

В формуле допустимо использование арифметических операторов (+, -, *, /,), операторов сравнения (=, >, <, <>, !=). Для присваивания используется оператор :=. Логические операторы представлены оператором «и» (&), оператором «или» (|) и оператором «не» (!). Для получения элемента списка возможно использование индекса (в квадратных скобках, индексация ведется с единицы).

2.12.1. Функции для управления ходом выполнения в формуле



Рис. 81. Функции для управления ходом выполнения

Функция @If() – это условный оператор. Базовая форма функции @If() имеет три части: условие, выражение, возвращаемое, если условие истинно и выражение, возвращаемое, если условие ложно:

```
@If(A > 0; A + 1; A - 1)
```

Если требуется использование вложенных условий, то они размещаются на нечетных позициях аргументов функции @If(). У функции @If() всегда должно быть нечетное количество аргументов. Конечно, можно просто вкладывать функции @If() друг в друга:

```
@If(@IsNewDoc; "Blue";  
@If(cColor = "Green"; "Purple"; "Mauve"))
```

Функции @DoWhile, @While и @For служат для организации циклов. Циклы в функциях @DoWhile и @While выполняются, пока условие condition истинно. Использование функции @For пояснит следующий пример:

```
REM {Образуем список из 5 значений};
cTextList := "January": "February": "March": "April": "May";
REM { Перебираем и выводим элементы списка};
REM { В цикле For проводится инициализация счетчика (j := 1)};
REM { Затем записывается условие продолжения цикла};
REM { (j <= длина списка) };
REM { Третий параметр - шаг цикла (j := j+1)};
@For(j := 1;
     j <= @Elements(cTextList);
     j := j+1;
     @Prompt([Ok]; "Номер элемента " + @Text(j); cTextList[j])
    )
```

Назначение функции @Do – объединить несколько вызовов команд или функций в один. Это может потребоваться в тех случаях, где по синтаксису необходимо одно утверждение.

Функция @Return обеспечивает выход из формулы, при этом значение формулы – это значение аргумента данной функции.

Функция @Nothing возвращает пустое значение в формулах преобразования типов или специальное значение null при вызове вне этих формул.

Функция @Select (не путать с ключевым словом SELECT!) выбирает из списка своих аргументов значение, заданное первым аргументом (или последнее значение, если первый аргумент больше длины списка значений):

```
@Select(2, "Alex", "Mary", "Peter")
```

Данная формула вернет строку "Mary".

2.12.2. Функции для работы с полями



Рис. 82. Функции для работы с полями

Использование ключевых слов DEFAULT и FIELD обсуждалось ранее. Заметим, что строка следующего вида

FIELD fieldname := fieldname

часто используется для создания в документе нового поля (в данном случае создается поле с именем fieldname). Вызовы FIELD fieldname := @DeleteField и FIELD fieldname := @Unavailable уничтожают поле в документе³. Функция @SetField используется для установки значения поля, а функция @GetField – для чтения значения.

Если создается формула, срабатывающая при обработке конкретного поля (например, обработка событий поля), то можно использовать формулы @ThisName и @ThisValue, которые возвращают строку с именем поля и значение поля соответственно.

Функция @Abstract предназначена для получения значения поля в специально обработанном виде. Например, при помощи следующего вызова возвращается значение поля description, в котором отсутствуют гласные буквы (английские гласные):

```
@Abstract([DROPVOWELS]:[ABBREV]; 200; ""; "description")
```

2.12.3. Функции для работы со списками и строками

Списки в Lotus Notes возникают как результат работы некоторых функций выбора (@DbColumn, @DbLookup). Также в виде списка воспринимаются значения поля, содержащего множество значений.

Рассмотрим основные функции для работы со списками значений, перечисленные в табл. 19.

Таблица 19

Функции для работы со списками значений

Имя функции	Возвращаемое значение
@Contains()	True, если список содержит указанное значение
@Elements()	Количество элементов списка
@Explode()	Создает список по строке или диапазону
@Implode()	Создает строку, состоящую из значений списка
@IsMember()	True, если список содержит указанное значение
@IsNotMember()	True, если список не содержит указанное значение
@Member()	Позиция элемента в списке
@Replace()	Замена элементов списка
@ReplaceSubString()	Замена подсписка исходного списка другим списком
@Subset()	Получение подсписка. Первый параметр – список. Если второй параметр положительное число, то выбираются первые слева значения списка. Если этот параметр отрицательное число, то выбираются значения списка справа
@Sum()	Сумма всех элементов числового списка
@Trim()	Список, из которого удалены пустые элементы
@Unique()	Список уникальных значений

³ Данные функции уничтожают содержимое поля, но пустые поля в документе не сохраняются (если поле содержало значение по умолчанию, то оно будет установлено в поле).

Для работы со списками, кроме перечисленных функций, применяются некоторые операторы. Для конкатенации списков применяется оператор «двоеточие» (:). При помощи этого оператора можно построить список из переменных, констант или значений полей. Единственное условие – все элементы списка должны быть одного типа.

При работе со списками можно использовать такие операторы, как +, -, *, /, >, <, >=, <=, = и !=. Нормальное использование предполагает обработку элементов списков парами. Например, следующий фрагмент кода генерирует список с такими элементами: "Chapter 11"; "Chapter 22".

```
jcTextList1 := "Chapter 1" : "Chapter 2";
jcTextList2 := "1" : "2";
jcTextList1 + jcTextList2
```

Добавление «звездочки» (*) к операторам списка позволяет создать список, являющийся декартовым произведением исходных списков, причем обработанным согласно указанному оператору. Если в предыдущем примере использовать следующий оператор

```
jcTextList1 *+ jcTextList2
```

то будет получен такой результат:

```
"Chapter 11"; "Chapter 12"; "Chapter 21"; "Chapter 22"
```

Язык формул содержит большой набор функций для работы со строками. Строковые данные – один из основных типов данных при обработке баз документов. Некоторые строковые функции описаны в табл. 20.

Таблица 20

Строковые данные

Имя функции	Назначение	Пример
<i>Обнаружение подстрок</i>		
@Begins()	Устанавливает, начинается ли строка с подстроки	@Begins("Lotus"; "Lo") = True
@Contains()	Устанавливает, содержит ли строка подстроку	@Contains("Lotus"; "otu") = True
@Ends()	Устанавливает, оканчивается ли строка подстрокой	@Ends("Lotus"; "Lo") = False
<i>Выделение подстроки</i>		
@Left()	Выделение левой подстроки	
@LeftBack()	Выделение левой подстроки (поиск с правого конца)	@LeftBack("Lotus Notes"; "o")="Lotus N" @LeftBack("Lotus Notes"; 7) = "Lotu"
@Middle()	Выделение подстроки	@Middle("Lotus Notes"; 2; 1) = "t"
@MiddleBack()	Выделение подстроки (поиск с правого конца)	@MiddleBack("Lotus Notes"; "es"; "us") = " Not"

@Right()	Выделение правой подстроки	@Right("Lotus Notes"; " ") = "Notes" @Right("Lotus Notes"; 3) = "tes"
@RightBack()	Выделение правой подстроки (поиск с правого конца)	@RightBack("Lotus Notes"; "o") = "tes" @RightBack("Lotus Notes"; 7) = "otes"
<i>Сравнение строк</i>		
@Like()	Сравнение строки с шаблоном (с учетом регистра)	@Like("A big t"; "a?????t") = False
@Matches()	Сравнение строки с шаблоном	@Matches("A big t"; "a?????t") = True
<i>Манипуляция со строками</i>		
@Length()	Длина строки	@Length("Lotus Notes") = 11
@LowerCase()	Преобразование к нижнему регистру	@LowerCase("Lotus Notes") = "lotus notes"
@ProperCase()	Первые буквы слов – большие, остальные маленькие	@ProperCase("LOTUS NOTES") = "Lotus Notes"
@UpperCase()	Преобразование к верхнему регистру	@UpperCase("Lotus Notes") = "LOTUS NOTES"
@Repeat()	Повтор строки	@Repeat("Hello";2) = "HelloHello"
@ReplaceSubString()	Замена указанной подстроки	@ReplaceSubstring("I like apples"; "like"; "hate") = "I hate apples"
@Text()	Конвертирование в строку других типов данных (второй параметр – строка формата)	@Text(800;"S") = "8.00E+02"
@Trim()	Удаление концевых и начальных пробелов	@Trim(" Alex ") = "Alex"

2.12.4. Функции для выборки информации

Для построения формулы выбора документов в представлении или агенте используются формулы, построенные с ключевым словом SELECT.

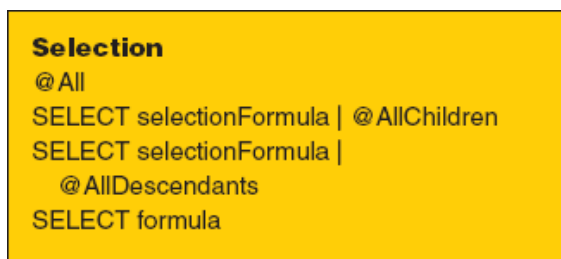


Рис. 83. Построение формул выбора

Параметры @AllChildren и @AllDescendants позволяют выбрать не только документы, но и все ответы на эти документы (@AllChildren) или ответы и «ответы на ответы» для документов (@AllDescendants).

Приведем несколько примеров формул выбора документов:

1. `SELECT @All` – выбор всех документов базы;
2. `SELECT @IsAvailable(Year) & Year>1995` – выбор документов, которые содержат поле `Year` и значение этого поля больше, чем 1995.
3. `SELECT @Author = "Volosevich" | @AllDescendants` – выбор документов, автором которых является `Volosevich`, а также ответы и «ответы на ответы» для этих документов.

Если база документов содержит некое представление, то возможна выборка данных из этого представления, удовлетворяющих некоторому критерию или ключу.

Функция `@DbColumn()` возвращает список значений из указанной колонки определенного представления. Это представление может размещаться как в текущей, так и во внешней базе документов. Синтаксис данной функции:

```
@DbColumn(class:cache; server:database; view; columnNumber)
```

Параметры:

- `class` – строка, указывает тип базы. Для баз `Dominio` можно использовать пустую строку или строку `"Notes"`.
- `cache` – строка, указывает на возможность использования кэширования при поиске. Если не планируется кэшировать данные, следует строку `"NoCache"`. Для включения кэша используется пустая строка `" "`.
- `server: database` – список строк, определяющий сервер и базу документов. Для поиска в текущей базе следует указать `":"`. Для поиска в локальной базе указывается пустое имя сервера, но записывается имя файла базы (если требуется, то с полным путем). Например, `":"DATABASE.NSF`. Для поиска в базе на сервере требуется дополнительно указать имя сервера: `"MYSERVER":"DATABASE.NSF"`.
- `view` – строка с именем представления, в котором производится поиск. Это имя должно совпадать с полным именем представления, указанным в его свойствах.
- `columnNumber` – число, номер колонки представления, данные из которой возвращает функция. При указании номера колонки, следует учитывать, что это должен быть номер колонки с данными, а не с константным значением или с результатом действия следующих функций: `@DocChildren`, `@DocDescendants`, `@DocLevel`, `@DocNumber`, `@DocSiblings`, `@DocParentNumber`, `@IsCategory`, `@IsExpandable`.

Приведем пример использования функции `@DbColumn`:

`@DbColumn("";"";"INVENTORY.NSF";"Inventory On Hand"; 2)` – поиск в текущей базе в представлении с именем `Inventory On Hand`, возвращается значение второй колонки.

Функция `@DbLookup()` возвращает список значений из указанной колонки определенного представления, удовлетворяющих ключу поиска. Синтаксис данной функции подобен синтаксису функции `@DbColumn`:

```
@DbLookup(class:cache; server:database; view; key;
columnNumber; keywords)
```

Параметры `class:cache`, `server:database`, `view`, `columnNumber` формируются по правилам, аналогичным правилам для `@DbColumn`. Параметр `key` (строка) задает ключ для выбора документов. Для работы функции `@DbLookup` представление должно содержать первую отсортированную колонку, которая и будет являться ключевой. В качестве ключа может выступать константа или редактируемое поле.

Необязательный список `keywords` может содержать следующие элементы:

[FAILSILENT] – если значения по ключу не найдены, возвращается пустая строка, а не генерируется ошибка;

[PARTIALMATCH] – для удовлетворения ключу достаточно частичного совпадения (с начала значения колонки);

[RETURNDOCUMENTUNIQUEID] – функция возвращает значение UNID документа, а не значение колонки или поля.

Для функции `@DbLookup` можно использовать следующий синтаксис:

```
@DbLookup(class:cache; server: database; view; key; fieldName; keywords)
```

В этом случае параметр `fieldName` содержит строку с названием поля документа, из этого поля и выбираются возвращаемые данные.

Примечание: функции `@DbColumn` и `@DbLookup` обладают следующим ограничением – они не могут вернуть объем данных, больший, чем 64 Кбайт.

2.12.5. Функции для осуществления диалога с пользователем

Для построения различных диалоговых окон служат следующие функции: `@DialogBox`, `@PickList` и `@Prompt`.

Функция `@Prompt` позволяет использовать различные виды окон диалога. Ее общий синтаксис выглядит следующим образом:

```
@Prompt([style]:[NOSORT]; windowtitle; promptText;
        defaultChoice; choiceList; fileType)
```

Возможные значения параметра `[style]` приведены в табл. 21.

Таблица 21

Значения параметра `[style]`

Значение <code>[style]</code>	Результат
[CHOOSEDATABASE]	Показывает окно открытия базы документов
[LocalBrowse]	Показывает окно выбора файлов на локальном диске
[OK]	Диалоговое окно с текстом и кнопкой OK
[OkCancelCombo]	Выпадающий список для выбора значений
[OkCancelEdit]	Окно с полем ввода и кнопками OK и Cancel
[OkCancelEditCombo]	Выпадающий список для выбора значений с возможностью ввода собственного значения
[OkCancelList]	Список для выбора одного из значений
[OkCancelListMult]	Список для выбора нескольких значений
[Password]	Окно с полем ввода пароля
[YesNo]	Диалоговое окно с текстом и кнопками Yes и No
[YesNoCancel]	Диалоговое окно с кнопками Yes, No и Cancel

Следующий пример демонстрирует использование функции @Prompt:

```
REM "Ввод количества часов";
jnHours := @Prompt([OKCANCELEDIT]; "Effort";
                  "Please enter the number of hours."; 0);
```

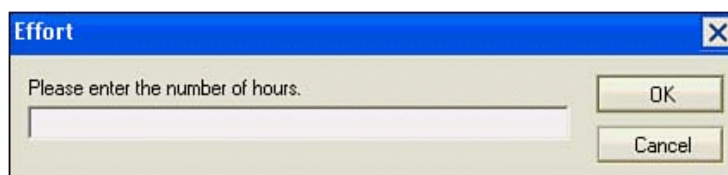


Рис. 84. Использование функции @Prompt

Функция @DialogBox позволяет организовать диалоговое окно для ввода информации на основе формы или подчиненной формы. Данная функция имеет следующий синтаксис:

```
@DialogBox(formname; [AutoHorzFit]: [AutoVertFit]: [NoCancel]:
           [NoNewFields]: [NoFieldUpdate]: [ReadOnly];
           [SizeToTable]; [NoOkCancel]: [OkCancelAtBottom];
           windowtitle)
```

Смысл параметров функции раскрывает табл. 22.

Таблица 22

Параметры функции @DialogBox

Параметр	Значение
formname	Имя формы, используемой для построения окна
[AutoHorzFit]	Подгонка горизонтального размера окна под размер формы
[AutoVertFit]	Подгонка вертикального размера окна под размер формы
[NoCancel]	Нет кнопки Cancel, только кнопка OK
[NoNewFields]	Новые поля не добавляются в связанный с формой документ
[NoFieldUpdate]	Изменения полей формы не переносятся в связанный с формой документ
[ReadOnly]	Запрещает редактирование полей
[SizeToTable]	Для отображения используется первая таблица формы
[NoOkCancel]	Нет кнопок Cancel и OK. Фактически, диалоговое окно просто показывает содержимое документа
[OkCancelAtBottom]	Кнопки размещаются внизу, а не с правой стороны окна
Windowtitle	Заголовок диалогового окна

Все параметры являются опциональными, за исключением параметра formname.

2.13. ЯЗЫК LOTUS SCRIPT

Система Lotus Notes/Domino предоставляет в распоряжение разработчика полноценный встроенный язык программирования – язык *LotusScript*. Язык может использоваться для написания обработчиков событий отдельных элементов базы документов, а также для создания агентов.

LotusScript является диалектом языка Visual Basic. Рассмотрим общие синтаксические правила LotusScript:

- Регистр при записи команд не учитывается.
- Разделителем команд (выражений) языка служит пробел.
- Выражения (кроме блочных операторов) следует располагать в одной строке или использовать символы пробела и подчеркивания (_) для переноса на другую строку.
- Несколько выражений в одной строке разделяются двоеточием (:).

Примеры:

```
'Одно выражение на одной строке
Print "One line"
'И на двух строках
Print "One" & _
"Two"
'Два выражения на одной строке
Print "One" : Print "Two"
```

Типы данных, поддерживаемые LotusScript и Visual Basic схожи. Принято разделять типы на *примитивные* (встроенные) и *сложные* (пользовательские). Информация о примитивных типах сведена в табл. 23.

Таблица 23

Типы данных, поддерживаемые LotusScript

Тип данных	Описание типа	Диапазон значений	Длина	Суффикс
<i>Integer</i>	числовой, целый	-32768 - 32767	2 байта	%
<i>Long</i>	числовой, целый	-2147483648 - 2147483647	4 байта	&
<i>Single</i>	числовой, вещественный	-3.402823E+38 - 3.402823E+38	4 байта	!
<i>Double</i>	числовой, вещественный	-1,7976931348623158E+308 - +1,1348623158E+308	8 байт	#
<i>Currency</i>	денежный (числовой с фиксированной точкой)	-922337203685477.5807 - +922337203685477.5807	8 байт	@
<i>String</i>	Строковый	Строка длиной от 0 байт до 4 Гбайт	4 байта (указатель)	\$
<i>Boolean</i>	Булевский	<i>True</i> или <i>False</i>	2 байта	нет
<i>Variant</i>	произвольный тип			нет

Тип данных отдельной переменной указывается при ее объявлении. Для указания типа можно использовать суффикс после имени переменной. Если тип данных не указан, переменная получает тип *Variant*. Следующие примеры показывают способы установки типа:

```
Dim rc 'Определили rc как Variant
Dim i% 'Определил i как Integer
Dim A As Currency 'Явно указали тип переменной
```

LotusScript позволяет использовать в коде переменные без предварительного объявления (в этом случае возможно задать тип суффиксом). Для того, чтобы запретить неявное определение переменной следует указать в

коде опцию **Option Declare**, после которой компилятор не допустит неявного определения.

Рассмотрим некоторые особенности работы со строковыми литералами. В LotusScript строковым литералом является последовательность символов, заключенная в следующие разделители:

- В пару двойных кавычек: "A quoted string"
- Между двумя вертикальными разделителями: |A bar string|
- Между открывающей и закрывающей фигурной скобкой: {A brace string}

Строковые литералы, заключенные между вертикальными чертами или фигурными скобками могут быть многострочными:

```
|A string  
on two lines|
```

Для включения строковых разделителей ", |, {, } в качестве текста в строку следует их удвоить: |A bar string with a bar || in it|

Рассмотрим правила построения идентификаторов:

1. Первый символ идентификатора – буква в нижнем или верхнем регистре. Остальные символы – буквы, цифры или символ подчеркивания.
2. Суффикс типа данных (% , & , ! , # , @ , \$) может быть добавлен, но он не является частью идентификатора.
3. Максимальная длина идентификатора – 40 символов, не включая суффикс.
4. Имена идентификаторов не зависят от регистра.
5. Если требуется включить в идентификатор запрещенный символ, то его следует экранировать тильдой (~).

Примеры использования идентификаторов:

```
'$ является запрещенным символом в идентификаторе  
Call ProductClass.LoMethod$ 'Так нельзя  
Call ProductClass.LoMethod~$ 'А вот так можно  
X = OLEClass.Hi@Prop 'Ошибка!  
X = OLEClass.Hi~@Prop 'Все ок!
```

LotusScript использует следующие операторы:

1. **Арифметические.** Print 3 + 4 'Печатает 7
2. **Битовые.** 2 And 3 'Битовое AND значений 10 и 11
3. **Булевы.** (4 > 0) And (4 < 10) 'Результат - True
4. **Операторы сравнения.** Print 7 <= 8 'Печатает True
5. **Строковая конкатенация.** Print "My cat " & "Geoffrey"
6. **Строковое сравнение.** Print "kid" < "kit" ' Prints True
7. **Присваивание.** newInt% = 8 + 12

Если выполняется присваивание объектов, то требуется использовать специальное ключевое слово **Set**:

```
Set objectVariable = objectReference
```

8. **Оператор Is**, для сравнения значений объектных ссылок.

```

Class ClassA
. . .
End Class

Dim X As New ClassA
Dim Y As ClassA
Set Y = X
Print X Is Y 'Результат: True

```

Рассмотрим возможности управления ходом выполнения в LotusScript. Операторы данной категории можно разделить на следующие функциональные группы:

1. Блочные операторы `If-Then-Else`, `If-Then-ElseIf` и `Select Case`.

```
If doCount% >= 1000 Then flagForm% = -1
```

Чтобы включить несколько инструкций в блок `Then`, следует разделить инструкции двоеточием:

```
If doCount% >= 1000 Then Print "Done." : flagForm% = -1
```

Следующий пример использует оператор `If-Then-ElseIf` для определения времени суток:

```

If timeTest! < 43200 Then
    Print "Morning"
ElseIf timeTest! < 64800 Then
    Print "Afternoon"
Else
    Print "Evening"
End If

```

Блочный оператор `Select Case` является аналогом оператора `switch` в языке C и подобен оператору `If-Then-ElseIf`. Синтаксис оператора:

```

Select Case selectExpr
[Case conditionList:
    [ statements ] ]
[Case conditionList:
    [ statements ] ]
. . .
[Case Else
    [ statements ] ]
End Select

```

В следующем примере добавляется правильный суффикс к количественному числительному:

```

Select Case anInt$
    Case "1", "21", "31", "41": printNum$ = anInt$ & "st"
    Case "2", "22", "32", "42": printNum$ = anInt$ & "nd"
    Case "3", "23", "33", "43": printNum$ = anInt$ & "rd"
    Case Else: printNum$ = anInt$ & "th"
End Select

```

2. Операторы ветвления `GoTo`, `If...GoTo...Else`, `On...GoTo`, `GoSub`, `On...GoSub`, `Return`. Синтакс:

```
GoTo label
On expression GoTo label, [, label ]...
```

Операторы ветвления `GoSub` и `On...GoSub` эмулируют вызов подпрограммы: передают управление в указанное место, выполняют последующие инструкции и, встретив `Return`, возвращают управление в точку вызова.

3. Итеративные блочные операторы `Do`, `For`, `ForAll`, `While`
Несколько синтаксических примеров. Это бесконечный цикл:

```
Do
    . . .
Loop
```

Условие проверяется перед итерацией:

```
Do While condition
    . . .
Loop
```

```
Do Until condition
    . . .
Loop
```

Условие проверяется после итерации:

```
Do
    . . .
Loop While condition

Do
    . . .
Loop Until condition
```

Цикл `For`:

```
For countVar = first To last [Step increment]
    [statements]
Next [countVar [ , countVar ]... ]
```

Итеративный блок `ForAll` выполняет операции один раз для каждого элемента массива или списка. Синтаксис:

```
ForAll refVar In container
    [statements]
End ForAll
```

Здесь `container` – это массив или список. Примечание: внутри блока `ForAll` переменной `refVar` можно присваивать значения. Пример использования:

```
Dim persStats List As String 'Определяем список из строк
'Присваиваем значение элементам списка
```

```

persStats("Name") = "Ian"
persStats("Age") = "36"
persStats("Home state") = "MD"

'Для каждого элемента persStats печатаем его tag и значение
ForAll idAttrib In persStats
    Print ListTag(idAttrib) & ": " & idAttrib
End ForAll

' Output:
' Name: Ian
' Age: 36
' Home state: MD

```

4. Операторы прерывания выполнения `End` и `Exit`

Перейдем к рассмотрению функций и процедур в LotusScript. Синтаксис объявления функции:

```

[Public|Private] [Static] Function Name [(params)] [As Type]
    [statements]
End Function

```

Модификатор доступа `Public` делает функцию доступной вне определяющего модуля или класса. Устанавливается по умолчанию для методов класса. Модификатор доступа `Private` делает функцию доступной только в пределах ее определения (в модуле или классе). Устанавливается по умолчанию для функций, определенных в модуле. Модификатор `Static` делает все переменные, определенные в функции, статическими: они сохраняют свои значения между вызовами функции. Пример:

```

'Определяем функцию с 3-мя параметрами: целой переменной,
'массивом и списком
Function FOver(a As Integer, b() As Integer, _
              c List As Integer) As Integer
.
.
.
End Function

Dim x As Integer
Dim y(5) As Integer
Dim z List As Integer
Call FOver(x, y, z)

```

Для возврата значения функции в теле функции возможно применение двух подходов. Первый заключается в использовании имени функции в левой части оператора присваивания, а возвращаемого значения – в правой. После такого оператора присваивания выхода из функции не происходит, ее выполнение продолжается. Второй подход – использование оператора `Return` в виде `Return <возвращаемое значение>`. При этом, в отличие от первого подхода, происходит немедленный выход из функции.

Синтаксис объявления процедуры в целом аналогичен объявлению функции:

```
[Public|Private] [Static] Sub Name [(parameters)]
    [statements]
End Sub
```

Пример объявления и вызова процедуры:

```
Sub PrintName2(someName As String)
    'Переводим строку в верхний регистр
    someName$ = UCase$(someName$)
    Print someName$
End Sub
```

```
firstName$ = "David"
Call PrintName2(firstName$)
```

Рассмотрим возможности LotusScript по работе с массивами и списками. LotusScript поддерживает *статические* и *динамические массивы*. Чтобы объявить статический массив используется следующий синтаксис:

```
'Статический одномерный массив:
Dim states(1 to 50) As String
states(1) = "Alabama"
```

```
'Статический двумерный массив:
Dim statesAnd10Cities(1 to 50, 1 to 10) As String
statesAnd10Cities(1,1) = "Alabama, Birmingham"
```

В LotusScript по умолчанию принята индексация элементов массива с нуля, если нижняя граница индекса не указана явно. Таким образом, инструкция `Dim a(5) As Integer` определяет массив из элементов `a(0)...``a(5)`. Чтобы изменить нижнюю границу по умолчанию с 0 на 1, следует использовать инструкцию `Option Base 1`.

Динамические массивы объявляются без указания индексов:

```
'Динамический массив строк:
Dim myDynamicArray() As String
```

Однако динамический массив нельзя использовать, пока не определены его измерения и границы с помощью команды `ReDim`. Синтаксис команды:

```
ReDim [Preserve] arrayName(bounds) [As dataType]
```

Ключевое слово `Preserve` указывает LotusScript, что следует сохранить текущие значения в массиве `arrayName`.

Чтобы освободить память занимаемую динамическим массивом, используют команду `Erase`. Примененная к статическому массиву, команда `Erase` только заново инициализирует элементы массива (нулями, пустыми строками, значениями `EMPTY` или `NOTHING`). Следующий фрагмент кода показывает пример работы с массивами:

```
Option Base 1
'Объявим динамический массив строк
Dim myNames() As String
Dim ans1 As Integer
```

```

Dim ans2 As Integer
Dim counter As Integer
Dim userInput As String

'Просим пользователя ввести число и заносим значение в ans1%
ans1% = CInt(InputBox$("How many names want to enter?"))

'Используем ans1% как верхнюю границу для массива
ReDim myNames(ans1%)

'Пользователь вводит имена, а мы заносим их в массив
For counter% = 1 To ans1%
    myNames(counter%) = InputBox$("Enter a name: ")
Next

'Печатаем содержимое массива, значения отделяются пробелами
For counter% = 1 To ans1%
    Print myNames(counter%) " "
Next

'Пользователь вводит другое число, сохраняем это число в ans2%
ans2% = CInt(InputBox$("How many more names?"))

'Изменяем размерность массива, сохраняя существующие элементы
If ans2% > 0 Then
    ReDim Preserve myNames(ans1% + ans2%)
    'Вводим новые значения в массив
    For counter% = 1 To ans2%
        myNames(counter% + ans1%) = InputBox$("Enter a name: ")
    Next

    'Выводим содержимое массива
    For counter% = 1 To ans1% + ans2%
        Print myNames(counter%) " "
    Next
End If

```

LotusScript поддерживает работу со списками. *Список* – это одномерная коллекция элементов одного типа данных. Во время выполнения в любой момент можно изменять размер списка (во время компиляции не выделяется место для размещения элементов списка). Список автоматически сжимается и расширяется при удалении или добавлении элемента. Доступ к элементу списка осуществляется по уникальному строковому имени, называемому *тегом*.

Следующий пример иллюстрирует как объявить список, добавить в него элементы и работать с этими элементами. Элементы списка имеют тип `String`.

```

'Сделаем сравнение строк не зависящим от регистра
Option Compare NoCase

Dim myList List As String
Dim newTag As String
Dim newValue As String

```



```

'Добавляем элементы в список
myList("A1234") = "Andrea"
myList("A2345") = "Vera"
myList("A3456") = "Isabel"

'Пользовательский ввод:
newTag$ = InputBox$("Please enter your ID:")
newValue$ = InputBox$("Please enter your first name:")
' Добавляем элемент
myList(newTag$) = newValue$
Print myList(newTag$) 'Вводим имя пользователя

```

LotusScript предоставляет большое количество функций для работы со списками. Вот некоторые из них:

TypeName(listName) – возвращает имя типа элемента списка и слово "LIST" в символах верхнего регистра.

IsList(listName) – возвращает **True** (-1) или **False** (0) в зависимости от того, является ли listName списком.

IsElement(listName(stringExpr)) – возвращает **True** (-1) или **False** (0) в зависимости от того, имеется ли у списка listName тег stringExpr.

ListTag(refVar) – возвращает имя текущего элемента из списка, поэлементно обрабатываемого командой **ForAll**. refVar – ссылка на переменную, определенную в заголовке цикла **ForAll**.

Erase listName – удаление всех элементов списка и освобождение выделенной памяти.

Erase listName(listTag) – удаление конкретного элемента списка и освобождение занимаемой элементом памяти.

2.14. DOMINO OBJECT MODEL

Объектная модель классов Domino (Domino Object Model, DOM) позволяет разработчику приложений получить из LotusScript доступ к базам Lotus Notes и содержащимся в них объектам (представлениям, документам, полям в документе), а также к объектам интерфейса пользователя. Общая структура системы классов представлена на схеме (рис. 85).

Встроенные классы можно разделить на две группы:

- классы интерфейса пользователя (User Interface, UI) или классы «переднего плана» (front-end),
- классы «заднего плана» (back-end).

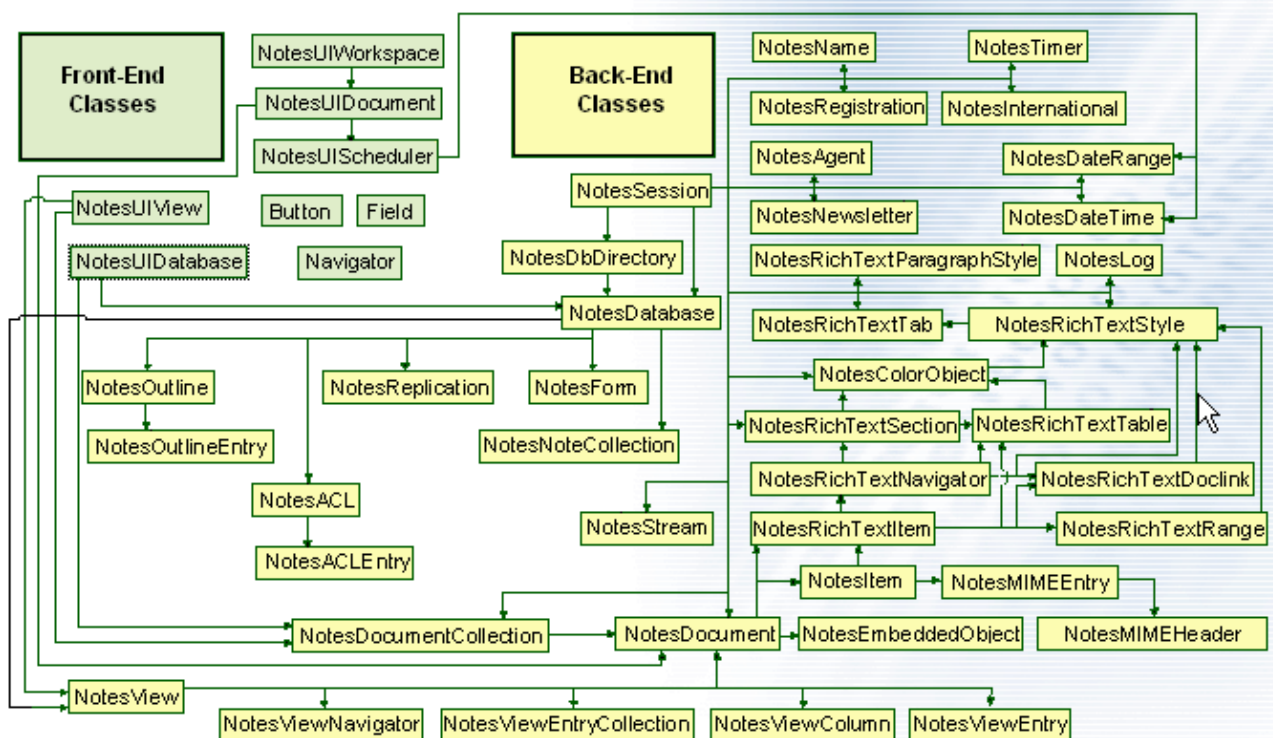


Рис. 85. Схема общей структуры системы классов

Рассмотрим классы интерфейса пользователя. Они обеспечивают доступ к таким объектам пользовательского интерфейса, как рабочее пространство Notes (Workspace), окно вида (View window), окно документа (Document window), поле в окне документа.

Класс NotesUIWorkspace

Класс NotesUIWorkspace является базовым для всех классов интерфейса пользователя. Обычно для получения доступа к другим пользовательским классам (например, NotesUIDocument или NotesUIView), необходимо создать объект класса NotesUIWorkspace. Основные свойства класса содержит табл. 24.

Таблица 24

Свойства класса NotesUIWorkspace

Свойство	Описание, пример
CurrentDatabase	Содержит объект NotesUIDatabase, представляющий текущую базу, с которой работает пользователь
CurrentDocument	Возвращает ссылку на объект класса NotesUIDocument, соответствующий текущему документу в рабочем пространстве. Текущим называют документ, который в настоящее время открыт (для просмотра или редактирования) или выбран («выделен» прямоугольной рамкой курсора) в представлении или папке. Часто используется для получения соответствующего back-end объекта класса NotesDocument <pre>Dim ws As New NotesUIWorkspace Dim uidoc As NotesUIDocument Dim doc As NotesDocument Set uidoc = ws.CurrentDocument Set doc = uidoc.Document</pre>

Укажем некоторые методы класса NotesUIWorkspace (табл. 25).

Таблица 25

Методы класса NotesUIWorkspace

Метод	Описание
ComposeDocument	Создает новый документ по форме form\$ из базы file\$ на сервере server\$, делает его текущим документом и показывает в рабочем пространстве
DialogBox	Выводит в модальном диалоговом окне текущий документ (открытый в окне или выбранный в представлении или папке) по указанной форме
EditDocument	Открывает текущий документ в заданном параметром editMode (тип Boolean) режиме. True задает режим редактирования, False – режим просмотра. Если параметр опущен, документ открывается в режиме редактирования
OpenDatabase	Открывает в рабочем пространстве пользователя указанный вид из указанной базы.
OpenFileDialog SaveFileDialog	Показывает пользователю диалог выбора\сохранения файла
ViewRefresh	Обновляет текущий вид

Класс NotesUIDocument

Объект класса NotesUIDocument представляет документ, который открыт в рабочем пространстве Notes. Свойства класса NotesUIDocument позволяют получить доступ к документу как объекту класса NotesDocument, определить, новый ли это документ, в каком поле находится курсор, какой заголовок окна и т.д. Методы класса NotesUIDocument позволяют выполнять типовые операции с буфером обмена, «категоризацию» документа, пересчет полей и т.д. Таблица 26 содержит описание некоторых свойств и методов.

Таблица 26

Свойства и методы класса NotesUIDocument

Имя	Описание
AutoReload	Если свойство включено, то документ в рабочем пространстве должен автоматически перезагружаться (в память) из базы, если в соответствующем документе в базе произошли изменения
CurrentField	Если документ находится в режиме редактирования, возвращается имя поля (тип String), в котором в настоящее время находится курсор. В режиме чтения всегда возвращается пустая строка
Document	Объект класса NotesDocument, представляющий документ в базе, который соответствует текущему документу в рабочем пространстве
EditMode	Позволяет выяснить или изменить режим документа в рабочем пространстве (редактирование или просмотр)
IsNewDoc	Позволяет проверить, является ли документ в рабочем пространстве новым. Возвращает True, если документ новый (еще ни разу не сохранялся в базе), или False в противном случае

FieldGetText\ FieldSetText	Считывает значение указанного поля или устанавливает его новое значение
Close	Закрытие документа
Copy, Cut, Paste	Копирует, вырезает и вставляет текущее выделение
Save	Сохраняет документ

Класс NotesUIDatabase

Класс предоставляет UI доступ к текущей открытой базе. Свойства класса представлены в табл. 27.

Таблица 27

Свойства класса NotesUIDatabase

Имя	Описание
Database	Возвращает соответствующий объект NotesDatabase (back-end представление базы)
Documents	Возвращает коллекцию NotesDocumentCollection, содержащую документы, относящиеся к текущему событию базы. Например, в обработчике события QueryDocumentDelete эта коллекция будет содержать документы, которые пользователь собирается удалить

Некоторые методы класса NotesUIDatabase приведены в табл. 28.

Таблица 28

Методы класса NotesUIDatabase

Имя	Описание
Close	Закрывает базу, в том числе все представления, документы и т.д.
OpenView	Открывает заданное представления и факультативно подсвечивает первое вхождение заданного ключа

Перейдем к рассмотрению классов заднего плана. Они представляют такие объекты Notes, как базу данных, представление или папку, агент, документ, заметку (Item) в документе. Программист с использованием объектов этих классов может манипулировать перечисленными объектами Notes из разработанных им скриптов. При этом выполняемые над объектами действия происходят без явного отображения в пользовательском интерфейсе, как бы «за сценой, на заднем плане».

Класс NotesSession

Класс NotesSession представляет в среде Notes текущую сессию (сеанс), обеспечивает доступ к переменным среды, адресным книгам, текущей базе данных и текущему агенту, позволяет получать информацию о текущем пользователе, текущей платформе и версии Notes и т.д. Для получения доступа к текущей сессии необходимо методом `New` создать объект класса NotesSession:

```
Dim variableName As New NotesSession
'или
Set notesSession = New NotesSession
```

Поскольку выполнение любого скрипта всегда происходит в рамках одной сессии, очередной вызов метода `New` для `NotesSession` будет всегда возвращать тот же самый объект.

Свойства класса `NotesSession` рассмотрены в табл. 29.

Таблица 29

Свойства класса `NotesSession`

Имя свойства	Описание
<code>AddressBooks</code>	Возвращает адресные книги (тип <code>Array of NotesDatabases</code>), известные во время выполнения скрипта
<code>CommonUserName</code>	Возвращается составляющая CN (canonical name, тип <code>String</code>) из полного имени текущего пользователя. Когда скрипт выполняется на станции, возвращается имя пользователя, под ID которого работает станция <code>Notes</code> . Когда скрипт выполняется на сервере, возвращается имя сервера. Если имя простое (не иерархическое), то составляющая CN для него совпадает с полным именем
<code>CurrentDatabase</code>	Объект класса <code>NotesDatabase</code> , представляющий базу, в которой находится данный скрипт. Эта база может оказаться как открытой, так и нет
<code>EffectiveUserName</code>	Если скрипт выполняется на станции, возвращается полное имя пользователя (тип <code>String</code>). Если скрипт выполняется на сервере, возвращается полное имя разработчика, который создал (или последним редактировал) этот скрипт – <i>владельца скрипта</i> . Свойство предназначено для определения лица, ответственного за последствия выполнения скрипта
<code>SavedData</code>	Возвращает <code>NotesDocument</code> , содержащий данные, которые можно сохранять между вызовами агента
<code>UserGroupNameList</code>	Возвращает массив строк всех групп, к которым принадлежит пользователь

Методы класса `NotesSession` описаны в табл. 30.

Таблица 30

Методы `NotesSession`

Имя метода	Описание
<code>GetDatabase</code>	Создает объект класса <code>NotesDatabase</code> , используя при этом заданные параметрами имя сервера и имя файла базы, и, если это удастся, открывает базу. Чтобы узнать, успешно ли было произведено открытие базы, можно использовать свойство <code>NotesDatabase.IsOpen</code>
<code>GetEnvironmentString</code> <code>GetEnvironmentValue</code>	Эти два метода возвращают значение (строковое или числовое) заданной переменной окружения
<code>SetEnvironmentVar</code>	Устанавливает значение переменной окружения
<code>CreateDOMParser</code>	Создает XML DOM объект по переданным XML данным
<code>CreateXSLTransformer</code>	Создает новый <code>NotesXSLTransformer</code> объект

Класс NotesDatabase

Объект класса NotesDatabase представляет базу документов Notes. Свойства объекта класса NotesDatabase позволяют получать доступ к содержащимся в базе объектам: списку управления доступом (ACL), агентам, коллекции из всех документов, представлениям и т.д., а также контролировать свойства самой базы: сервер размещения, путь и имя файла, размер, название базы, текущий уровень доступа пользователя.

Для доступа к текущей базе документов используйте свойство NotesSession.CurrentDatabase:

```
Dim s As New NotesSession
Dim db As NotesDatabase
Set db = s.CurrentDatabase
```

Для получения другой базы, можно поступить следующим образом:

```
Dim s As New NotesSession
Dim db As NotesDatabase
Set db = s.GetDatabase("MyServer/MyOrg", _
    "Customers\KewLDB.nsf")

If db.IsOpen Then
    . . .
End If
```

Не лишним будет проверить, удалось ли получить требуемую базу с помощью свойства NotesDatabase.IsOpen.

Свойства класса NotesDatabase представлены в табл. 31.

Таблица 31

Свойства класса NotesDatabase

Имя свойства	Описание
AllDocuments	Коллекция (объект класса NotesDocumentCollection) всех документов в базе
CurrentAccessLevel	Константа типа Integer , позволяющая узнать, каков уровень доступа текущего пользователя к базе. Возможные значения: ACLLEVEL_NOACCESS – нет доступа, ACLLEVEL_DEPOSITOR – депозитор, ACLLEVEL_READER – читатель, ACLLEVEL_AUTHOR – автор, ACLLEVEL_EDITOR – редактор, ACLLEVEL_DESIGNER – дизайнер, ACLLEVEL_MANAGER – менеджер
FileName, FilePath	Возвращают имя базы и полный путь к базе
IsFTIndexed	Возвращает True , если база имеет индекс полнотекстового поиска, или False , если не имеет. Пример: прежде чем выполнять запрос полнотекстового поиска, скрипт аккуратно проверяет, имеет ли база индекс поиска. Dim session As New NotesSession Dim db As NotesDatabase Dim collection As NotesDocumentCollection

	<pre> Set db = session.CurrentDatabase If db.IsFTIndexed Then Set collection= db.FTSearch("Clay tablets",0) End If </pre>
IsLink	Определяет, есть ли на базу ссылка с помощью линка. Файловый линк (file link) – это просто тестовый файл, содержащий действительный путь к базе и имеющий расширение .NSF
IsOpen	Определяет, открыта ли база (в этом случае можно получить доступ ко всем свойствам базы). Базу можно попытаться открыть с помощью метода NotesDatabase.Open
IsPrivateAddressBook, IsPublicAddressBook	Является ли база личной или общей адресной книгой. Часто используются совместно с NotesSession.AddressBooks
Server	Имя сервера (тип String), на котором размещается база, или пустая строка (" "), если база расположена локально

Методы класса NotesDatabase рассмотрены в табл. 32.

Таблица 32

Методы класса NotesDatabase

Имя метода	Описание
CreateDocument	Создает в базе, к которой применяется метод, новый документ, и возвращает объект класса NotesDocument, представляющий этот документ. Чтобы созданный документ сохранился в базе, не забудьте применить к нему метод Save
CreateView	Позволяет программно создать вид
FTSearch	Выполняет запрос полнотекстового поиска по всем документам в базе. Для поиска только по документам из заданного вида используют одноименный метод класса NotesView
GetDocumentByID, GetDocumentByUNID	По заданному идентификатору документа (NoteID) или универсальному идентификатору документа (UNID) находит документ в базе и возвращает объект класса NotesDocument, представляющий этот документ
GetView	По заданному имени возвращает вид или папку из базы, к которой применяется метод. Если база расположена на сервере, метод может вернуть только общий вид или папку. Если же база локальна, метод может вернуть как общий, так и личный вид или папку (личные виды и папки хранятся не в базе, а в файле DESKTOP.DSK пользователя)
Open	Открывает базу (более строго, объект класса NotesDatabase). Когда база открыта, доступны все ее свойства и методы. Возможны два случая использования метода Open: 1. Когда объект класса NotesDatabase уже связан с существующей базой. В этом случае вы должны либо задать оба параметра метода пустыми строками, например db.Open(" ", " "), либо указать их в точности такими, как они уже установлены в объекте класса NotesDatabase. Если же вы попытаетесь "переназначить" уже установленные в объекте сервер или файл, то получите сообщение об ошибке. 2. Когда объект класса NotesDatabase создан, но пока не

	<p>связан с существующей базой. В этом случае вы должны указать имя сервера (пустая строка вместо имени сервера в этом случае означает, что база расположена локально), и имя базы (оно не может быть в этом случае пустой строкой). Пример: скрипт открывает первую базу, найденную на текущем компьютере (сервере или станции):</p> <pre>Dim directory As New NotesDbDirectory("") Dim db As NotesDatabase Set db = directory.GetFirstDatabase(DATABASE) Call db.Open("", "")</pre>
OpenMail	<p>Открывает объект класса NotesDatabase, связанный с почтовым ящиком пользователя. Использование метода имеет смысл либо со станции пользователя (скрипт, выполняющийся на станции, может открыть базу на доступном сервере), либо на почтовом сервере владельца скрипта (поскольку скрипт, выполняющийся на сервере, не может открыть базу на другом сервере). Пример: скрипт определяет и выводит в диалоговом окне название и имя сервера почтового ящика пользователя, на станции которого скрипт выполняется.</p> <pre>Dim db As New NotesDatabase("", "") Call db.OpenMail Call db.Open("", "") MessageBox(db.Title & "on server " & db.Server)</pre>
Search	<p>Возвращает коллекцию документов, удовлетворяющих заданному формулой критерию отбора. Пример: скрипт отбирает в базе все документы-ответы (формула отбора – @IsResponseDoc), созданные или модифицированные после 1 декабря 1994 года.</p> <pre>Dim db As New NotesDatabase("Serv", "some-docs.nsf") Dim coll As NotesDocumentCollection Dim dateTime As New NotesDateTime("12/01/94") Set coll = db.Search("@IsResponseDoc", dateTime, 0)</pre>

Класс NotesView

Класс представляет вид (view) или папку (folder) в базе данных и обеспечивает доступ к содержащимся в виде или папке документам. Свойства класса NotesView позволяют определить название вида или папки, дату и время его создания и последней модификации, выяснить, представляет ли этот объект вид или папку и т.д. Методы класса NotesView, прежде всего, позволяют программисту свободно осуществлять навигацию по документам из вида или папки.

Свойства класса NotesView описаны в табл. 33

Свойства класса NotesView

Имя свойства	Описание
AutoUpdate	При изменении и сохранении документа соответствующий вид каждый раз обновляется, что может пагубно сказаться на производительности или вызвать ошибки, если изменения в документах вызывают их пересортировку в виде. Данное свойство (по умолчанию установлено в True) позволяет временно заблокировать данное автообновление
EntryCount	Возвращает количество документов в виде

Методы класса NotesView представлены в табл. 34.

Методы класса NotesView

Имя метода	Описание
FTSearch	Выполняет заданный запрос полнотекстового поиска по документам из данного вида и накладывает на вид соответствующую фильтрацию (т.е. отображает в виде только удовлетворяющие запросу документы)
GetAllDocumentsByKey	Возвращает коллекцию NotesDocumentCollection, содержащую все документы с совпадающими значениями в первой (и следующих) отсортированных или распределенных по категориям колонке и ключе. Документы возвращаются в неотсортированном виде. Свойство ColumnValues результирующих документов является недоступным
GetAllEntriesByKey	Возвращает коллекцию NotesViewEntryCollection объектов NotesViewEntry. Результирующие документы отсортированы так же, как и в исходном виде. Свойство ColumnValues доступно для использования
GetFirstDocument, GetLastDocument, GetNextDocument, GetPrevDocument	Используются для навигации по документам вида. Например: <pre>Dim s As New NotesSession Dim db As NotesDatabase Dim view As NotesView Dim doc As NotesDocument Set db = s.CurrentDatabase Set view = db.getView("Main") view.AutoUpdate = False Set doc = view.getFirstDocument Do While Not(doc Is Nothing) doc.ConcatValues = doc.ColumnValues(1) & "~" & _ doc.ColumnValues(2) Call doc.Save(True, False) Set doc = view.getNextDocument(doc) Loop</pre>

GetNthDocument	Возвращает документ, имеющий заданный параметром номер (тип <code>Integer</code>) в виде. Первый документ в виде имеет номер 1
Refresh	Данный метод приводит содержимое вида в соответствие с изменениями, произошедшими в базе с момента создания данного объекта класса <code>NotesView</code> или с момента предыдущего вызова метода <code>Refresh</code>

Класс `NotesDocumentCollection`

Этот класс представляет *коллекцию* (т.е. некоторое множество) документов из базы данных, выбранную согласно указанным критериям. Свойства класса `NotesDocumentCollection`, описанные в табл. 35, позволяют узнать, сколько в коллекции документов, в результате выполнения какого поискового запроса она была получена и является ли она упорядоченной.

Таблица 35

Свойства класса `NotesDocumentCollection`

Имя свойства	Описание
Count	Количество документов (тип <code>Long</code>) в коллекции. Следующий пример скрипта определяет количество документов в базе: <pre>Dim session As New NotesSession Dim db As NotesDatabase Dim collection As NotesDocumentCollection Set db = session.CurrentDatabase Set collection = db.AllDocuments MessageBox collection.Count</pre>

Методы класса `NotesDocumentCollection` рассмотрены в табл. 36.

Таблица 36

Методы класса `NotesDocumentCollection`

Имя метода	Описание
AddDocument, DeleteDocument	Добавляет или удаляет документ из коллекции. Важно, что нельзя создать пустую коллекцию и добавлять в нее документы. Коллекцию, возможно и пустую, должен вернуть метод объекта (например, <code>NotesView.GetAllDocumentsByKey</code>). После этого в коллекции можно добавлять и удалять документы
GetFirstDocument, GetLastDocument, GetNextDocument, GetPrevDocument	Используются для навигации по коллекции
GetNthDocument	Возвращает документ, имеющий в коллекции заданный номер. Номер документа в коллекции изменяется в диапазоне от 1 до <code>NotesDocumentCollection.Count</code> . Пустая коллекция содержит 0 документов
PutAllInFolder, RemoveAllFromFolder	Помещает (удаляет) все документы коллекции в указанную папку
RemoveAll	Удаляет из базы каждый документ в текущей коллекции.

StampAll	Позволяет установить значение поля сразу во всех документах коллекции
----------	---

Класс NotesDocument

Объект класса NotesDocument представляет документ в базе данных Notes. Для создания нового документа – нового объекта класса NotesDocument – можно использовать либо метод New, либо метод CreateDocument из класса NotesDatabase. Метод New получает в качестве параметра notesDatabase объект класса NotesDatabase, представляющий базу, в которой должен быть создан документ:

```
Dim variableName As New NotesDocument(notesDatabase)
'или
Set notesDocument = New NotesDocument(notesDatabase)
```

Для получения доступа к существующему документу необходимо использовать соответствующие методы классов NotesDatabase, NotesView и NotesDocumentCollection.

Свойства класса NotesDocument представлены в табл. 37.

Таблица 37

Свойства класса NotesDocument

Имя свойства	Описание
ColumnValues	Массив вычисленных значений (тип Array of Variants), каждое из которых «представляет» этот документ в соответствующем столбце родительского вида для документа. Первый элемент массива содержится в первом столбце вида, второй – во втором, и так далее
EmbeddedObjects, HasEmbedded	EmbeddedObjects – массив содержащихся в документе OLE-объектов (тип Array of NotesEmbeddedObjects). OLE-объекты могут содержаться в пунктах типа RichText документа, а также в составе формы, сохраненной в документе. Если в документе содержатся встроенные объекты, то свойство HasEmbedded вернет True
IsNewNote	True, если документ ни разу не сохранялся на диск
Items	Возвращает массив объектов класса NotesItem (тип Array of NotesItems), представляющих все заметки в документе. Порядок элементов в массиве соответствует порядку следования полей в документе. Нижняя граница массива равна 0
Responses	Возвращает коллекцию NotesDocumentCollection документов, которые являются прямыми документами-ответами на данный. Ответы на ответы не включаются в эту коллекцию

Методы класса NotesDocument рассмотрены в табл. 38.

Методы класса NotesDocument

Имя метода	Описание
ComputeWithForm	Проверяет документ путем вычисления для его пунктов, одноименных с полями сопоставленной документу формы, всех формул значений по умолчанию, трансляции и проверки ввода
CreateRichTextItem	Создает в документе, к которому применяется метод, новый пункт типа RichText с указанным параметром именем (тип String), и возвращает соответствующий созданному пункту объект класса NotesRichTextItem
GetFirstItem	Документ может содержать более одного пункта с одинаковым именем. Этот метод возвращает первый пункт (как объект класса NotesItem) с именем, заданным параметром name\$ (тип String)
GetItemValue	Возвращает из документа значение пункта с заданным именем
MakeResponse	<p>Делает документ, к которому применяется метод, документом-ответом на документ, который задан параметром метода. Учтите (хотя это и очевидно), что оба документа должны находиться в одной базе. Пример: скрипт делает второй документ из вида All documents ответным на первый документ из этого вида</p> <pre> Dim session As New NotesSession Dim db As NotesDatabase Dim view As NotesView Dim docA As NotesDocument Dim docB As NotesDocument Set db = session.CurrentDatabase Set view = db.GetView("All documents") Set docA = view.GetFirstDocument Set docB = view.GetNextDocument(docA) Call docB.MakeResponse(docA) docB.Form = "Response" Call docB.Save(True, True) </pre>
RemoveItem	Удаляет пункт из документа. В документе из базы изменения произойдут только после вызова метода Save
RemovePermanently	Удаляет документ из базы
ReplaceItemValue	Заменяет все пункты с заданным именем одним новым пунктом с заданным значением value. Если пункт с заданным именем не содержался в документе, метод создает новый пункт и добавляет его в документ
Save	Сохраняет изменения, выполненные в документе, находящемся «в виртуальной памяти», в базе данных

2.14.1. Обработка событий в LotusScript

LotusScript является событийно-ориентированным языком. События в базе данных можно подразделить на несколько уровней, таких как уровень базы данных, уровень вида, формы и поля. Большинство событий имеют префикс Query или Post. Код в Query-обработчиках выполняется до того, как произой-

дет событие. Post-события (обработчики) исполняют код после факта наступления события.

Имеется два события общего характера:

- Initialize – событие происходит при загрузке объекта в оперативную память. Например, при открытии базы возникает Initialize-событие для базы данных.
- Terminate – событие возникает при выгрузке объекта (например, при закрытии базы данных). При закрытии документа это событие происходит для каждого поля документа.

Опишем события уровня базы документов при помощи табл. 39.

Таблица 39

События в базе документов

Событие	Описание
PostOpen	Возникает после открытия базы данных, и после ее события Initialize
QueryDocumentDelete	Возникает при нажатии Delete на выбранном документе или виде и до того, как документ будет помечен как удаленный
QueryDocumentUndelete	Возникает при нажатии Delete на отмеченном «как удаленный» документе – для отмены удаления
PostDocumentDelete	Возникает при нажатии F9 на отмеченном «как удаленный» документе. Это событие возникает после того, как документ удален
QueryDragdrop	Возникает при «перетаскивании» документа из базы в вид или папку (с помощью мыши)
PostDragdrop	Возникает после операции «перетаскивания»
QueryDropToArchive	Возникает до того, как документ попадает в базу-архив
PostDropToArchive	Возникает после того, как документ попадает в базу-архив
QueryClose	Возникает как раз перед закрытием базы данных

События уровня формы возникают при операциях с документами, таких как открытие и закрытие документа, изменение состояния документа и пересчете формул в документе. Эти события перечислены в табл. 40.

Таблица 40

События уровня формы

Событие	Описание
QueryOpen	Возникает при открытии документа, до того как он открыт
PostOpen	Возникает после открытия документа, но до Initialize-события
QueryModeChange	Возникает при изменении состояния документа. Например, документ был в состоянии для чтения, а потом стал редактируемым
PostModeChange	Возникает после изменения состояния документа
QueryRecalc	Возникает до обновления формул в документе
PostRecalc	Возникает после того, как все формулы в документе обновлены
QuerySave	Возникает до сохранения документа
PostSave	Возникает после сохранения документа
QuerySend	Возникает как раз перед посылкой документа по электронной почте

PostSend	Возникает после посылки документа по электронной почте
QueryClose	Возникает перед закрытие документа

События уровня поля документа возникают в ответ на действия с полями документа, например при перемещении фокуса между полями. Обработчики этих события используются для размещения скриптов, обрабатывающих значение поля.

2.15. СОЗДАНИЕ АГЕНТОВ ДЛЯ БАЗ LOTUS/DOMINO

Агенты (agents) – это средство автоматизации для баз документов Lotus. Агенты представляют логически выделенные фрагменты кода, реализующие некоторую бизнес-логику. Они могут выполняться при наступлении в базе определенного события, по расписанию или запускаться вручную. Код агентов может быть создан на языке @-формул, LotusScript или Java.

Создавать агенты может как дизайнер при первоначальном создании базы документов, так и пользователь при эксплуатации базы. Существуют две категории агентов: *разделяемые* (shared) и *частные* (private). Частный агент может быть запущен тем человеком, который его создал, либо любым другим с правом доступа не ниже Designer. Когда агент создается и сохраняется, вместе с ним сохраняется и ID пользователя, который применяется для определения, может ли частный агент быть запущен любым заданным пользователем. Среди списка прав ACL имеются два правила, связанных с созданием агентов: Create Private Agents и Create LotusScript/Java. Последнее правило разрешает писать код агента с использованием языков LotusScript и Java. В табл. 41 перечислены права на создание агентов.

Таблица 41

Права на создание агентов

Для создания...	Необходимо право доступа...
Частные агенты	Право доступа Reader или выше. Право создания частных агентов должно быть определено в ACL
Использование частных агентов	Право доступа Reader или выше
Частные агенты с кодом LotusScript или Java	Право создания частных агентов и LotusScript/Java-агентов должно быть определено в ACL
Разделяемые агенты с помощью простых действий и формул	Право доступа Designer или выше
Разделяемые агенты с кодом LotusScript или Java	Право доступа Designer или выше. Право создания частных агентов и LotusScript/Java-агентов должно быть определено в ACL.

Рассмотрим процесс создания агентов в Domino Designer. Для создания агента требуется выполнить команду меню Create | Design... | Agent или выбрать в дереве дизайна базы пункт Shared Code | Agents. Настройка агента выполняется при помощи окна свойств агента (рис. 86).

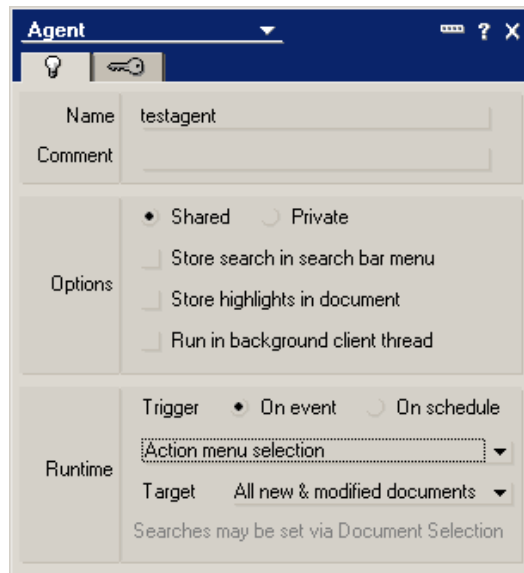


Рис. 86. Окно свойства агента

Любой агент должен иметь имя (Name). Кроме имени, агент может иметь псевдоним, записываемый через вертикальную черту. Поле Comment содержит произвольный текстовый комментарий к агенту.

В секции Options можно указать категорию агента (Shared или Private), а также указать, что код агент должен выполняться в отдельном потоке (Run in background client thread).

При помощи секции Runtime настраиваются такие важные параметры агента как *условие срабатывания* (Trigger) и *множество обрабатываемых документов* (Target). Если значение Trigger установлено в On schedule, то мы получаем агент, *запускаемый по расписанию*. В этом случае настраиваются такие очевидные параметры расписания как периодичность срабатывания и диапазон времени работы. Естественно, чтобы агент запустился в назначенное время, база данных должна находиться на работающем сервере Domino⁴.

В случае если значение Trigger установлено в On event, создается агент, *запускаемый по событию*. В выпадающем списке предлагается выбрать событие запуска агента:

1. Action menu selection – агенты, которые запускаются пользователем базы вручную из меню *Действия* (Action) на указанном множестве документов.
2. Agent list selection – агенты, вызываемые из кода обработчиков событий и других агентов. Они недоступны обычному пользователю из меню. В коде их можно получить по имени из объекта текущей базы:

```
Set agent=session.currentDatabase.GetAgent("Ulialia")
Call agent.run 'или call agent.runOnServer
```

При запуске агенту можно передать документ, на котором он должен работать, и делается это следующим образом: передается в качестве параметра NoteID документа в базе:

⁴ Особую категорию образуют агенты, запускаемые по расписанию, у которых в настройках частоты срабатывания указано Never. Это агенты, которые вызываются из других агентов.

```
Call agent.runOnServer(doc.NoteID)
```

В самом агенте переданный параметр и документ по параметру берутся так:

```
Dim session As New NotesSession  
Param = session.CurrentAgent.parameterDocID  
Set paramDoc=session.CurrentDatabase.GetDocumentbyID(param)
```

3. Before new mail arrives и After new mail has arrived – агенты, запускаемые прямо перед и сразу после прихода почты в базу документов.

4. After documents are created or modified – агенты, срабатывающие при создании нового или модификации существующего документа в базе.

5. When documents are pasted – агенты, запускаемые при вставке документа в базу.

Параметр Target определяет множество документов, на которых запускается агент. В отношении гибкости это весьма неудачный вариант. Вторым вариантом задания множества документов является определение этого множества в самом агенте. Это значит, что в коде агента необходимо взять нужные документы, например, из специально заготовленного представления, или воспользоваться поиском, который создаст то же представление, только динамически. Можно также в самом агенте взять документ, например, как переданный параметр.

После определения основных настроечных параметров агента можно переходить к написанию кода агента. Для этого используются

- Простые действия (Simple Actions);
- @-формулы;
- LotusScript;
- Java.

В случае использования простых действий можно отметить, что «программирование» происходит аналогично процессу создания Действий (Actions) и заключается в выборе и минимальной настройке команды из фиксированного списка.

Для создания агентов, базирующихся на формулах, нужно использовать предложение SELECT для указания обрабатываемых документов, причем это предложение должно быть первым в формуле. Если таковое не присутствует, Domino добавляет SELECT @All в начало формулы. Если агент включает предложение SELECT, которое определяет условия выбора, необходимо протестировать агента путем использования Select Documents в представлении (View), чтобы удостовериться в том, что Вы выбираете подходящие документы перед их модификацией.

Формулы для агентов могут легко манипулировать значениями полей, используя ключевое слово FIELD либо функцию @SetField(). Использование одной из этих двух опций позволяет устанавливать значение поля в значение другого поля, в комбинацию полей и текста и т.д. Также можно создавать и уда-

лять поля в документах. Ничего из вышеуказанного нельзя сделать, используя лишь простые действия.

Если тело агента создается с помощью LotusScript, то фактически пишется обработчик события Initialize агента. В случае применения Java автоматически генерируется следующая «заготовка» кода:

```
import lotus.domino.*;

public class JavaAgent extends AgentBase {
    public void NotesMain() {
        try {
            Session session = getSession();
            AgentContext agentContext = session.getAgentContext();
            // (Your code goes here)
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Как мы видим, агент на Java – это класс, производный от класса AgentBase. В коде Java используются классы, в основном соответствующие классам из Domino Object Model. Подробное описание данных классов можно найти в справочном руководстве.

2.16. ПУБЛИКАЦИЯ БАЗ LOTUS/DOMINO В WEB

Система Lotus Notes/Domino предоставляет в распоряжение разработчика набор средств, которые облегчают публикацию баз документов в сети Internet. В данном параграфе будут рассмотрены такие элементы дизайна баз как *страницы* (pages), *схемы* (outlines), *фреймы* и *фреймсеты* (framesets).

2.16.1. Страницы

Термин *страница* (page) взят из мира Internet-разработок, где для создания страниц, отображаемых в Web-браузерах, используется HTML. Страница Domino подобна форме, однако в отличие от последней, страница способна лишь отображать информацию (формы могут как отражать, так и собирать информацию). Страницы могут содержать текст, вычисляемый текст и встроенные объекты Domino, такие как представления, схемы, навигаторы, панели папок. Страницы также могут включать графику, карты изображений, HTML, таблицы, секции, ссылки, действия и апплеты. Однако на страницу нельзя помещать поля.

Создать новую страницу довольно просто; к настоящему времени вы должны быть достаточно знакомы с IDE, чтобы знать, что для создания страницы необходимо открыть БД в Domino Designer и в списке Design выбрать Pages. Это приведет к созданию новой, совершенно пустой страницы, которая по внешнему виду напоминает новую пустую форму. Подобно форме,

страница – документ, который можно наполнять всеми видами объектов, исключая поля.

Рассмотрим окно свойств страницы и назначение отдельных закладок окна:

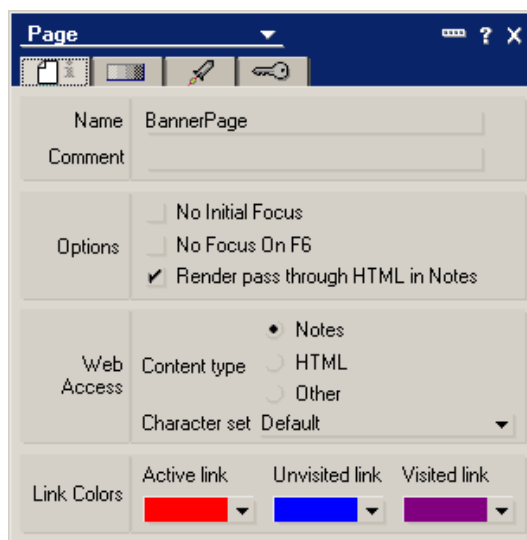


Рис. 87. Свойства страницы

Page Info – эта закладка позволяет дать странице имя, устанавливать опции фокуса, определять Web-доступ и устанавливать цвета ссылок. На самой верхней секции закладки вводятся *имя*, *псевдоним* (через вертикальную черту, после имени) и *комментарий* к странице. Секция Options контролирует фокус страницы и определяет, обрабатывает ли Notes содержимое страницы как встроенный HTML-код. Если установить флажок Treat Page Contents as HTML, можно создать всю страницу полностью посредством HTML-кода. Секция Web Access определяет, каким образом обрабатывается содержимое страницы при доступе к нему Web-клиента. Вы можете определить тип содержания и набор символов. Секция Link Colors позволяет определять цвета для активных, непосещенных и посещенных ссылок для Web-клиентов. По умолчанию цвета соответственно красный, синий и пурпурный.

Background – эта закладка позволяет выбирать графику фона или цвет для страницы.

Launch – закладка позволяет определять опции запуска (Launch) и фрейма (Frame) страницы. Подобно закладке Launch формы, данная закладка содержит две секции: Auto Launch и Auto Frame. Список Auto Launch для страницы имеет меньшее количество выборов, нежели аналогичный список для формы:

- None (по умолчанию) – не имеет особых свойств запуска;
- First Attachment – запускает первый аттачмент файла на странице;
- First Document Link – запускает первую ссылку на документ на странице;
- First OLE Object – запускает первый OLE-объект на странице.

Security – эта закладка содержит единственный флажок – Available to Public Access Users, разрешающий доступ к странице неавторизованных пользователей.

При разработке страницы можно воспользоваться прямым добавлением HTML-кода к странице. Для этого есть несколько способов. Первый способ заключается в создании всей страницы при использовании HTML-кода, а затем установке свойства Content Type в секции Web Access диалога свойств страницы в значение HTML. Второй способ состоит в создании фрагментов HTML-кода непосредственно на странице. Несмотря на тот факт, что нельзя помещать на страницу поля в качестве элементов дизайна, можно на странице создать форму ввода, которая может включать поля. Вам для этого просто необходимо вручную написать HTML-код для создания формы на странице с методом FORM METHOD=post и поля с тегом INPUT.

Страница часто используется как средство навигации по различным частям сайта. Создание объектов на странице, которые ссылаются на другие места сайта, такие как базы данных, представления, страницы, документы и даже другие сайты, является нормой при написании страниц. В сайтах, написанных строго с использованием HTML, все эти ссылки должны быть прописаны вручную – они не динамичны. Преимущество использования Domino заключается в том, что Вы можете предоставить вид Web-клиенту, а документы могут динамически добавляться и удаляться из представления путем автоматического обновления вида для Web-браузера. И Вам не придется при этом писать ни строчки HTML-кода!

Иногда Вам придется вставлять ссылки на представления, другие БД, документы или страницы. Создать ссылку нетрудно, а сделать это можно несколькими способами:

1. Вы можете скопировать ссылку на объект Domino, такой как БД, документ, вид или страница в буфер обмена, а затем вставить на страницу. Для копирования объекта дизайна, такого как страница или представление, Вы выделяете его на рабочей панели (Work pane) и выбираете пункт меню Edit | Copy as Link | Named Element. Пока ссылка находится в буфере, Вы можете вернуться на страницу, выделить текстовый или графический объект, который Вы хотите использовать в качестве ссылки, и выбрать пункт меню Create | Hotspot | Link Hotspot. Затем нажимаете кнопку Paste в диалоге свойств, и объект будет вставлен в Hotspot Resource Link box.

2. Также можно создать ссылку посредством *диалога свойств ссылки* (Link properties box). Например, для создания ссылки из текста на представление в БД вначале необходимо выделить текст. Затем выбрать пункт меню Create | Hotspot | Link Hotspot. Автоматически открывается диалог свойств ссылки (Hotspot Resource Link properties box). Определите Named Element (именованный элемент) в поле Type и выберите View. Нажмите кнопку Browse и выберите представление. Имя представления автоматически вводится в поле Value. Ввод имени фрейма в поле Frame заставляет страницу запускаться в определенном фрейме фреймсета.

Как и в случае форм, Вы можете внедрять объекты дизайна Domino в страницы путем выбора пункта меню Create | Embedded Element | element type (тип элемента). Встраивание элементов в страницы дает Вам большую гибкость и возможность управлять их представлением. Когда виды, схемы, устройства выбора даты и другие объекты Domino предоставляются Web-клиентам, их функциональность очень близка той, которая доступна Notes-клиентам. Это на один шаг приближает Вас к проектированию единственного интерфейса, который функционирует одинаково хорошо для обоих видов клиентов.

Выбрать элемент для внедрения можно из следующего списка:

- Outlines (Схемы)
- Views (Представления/Виды)
- Navigators (Навигаторы)
- Date pickers (Устройства выбора даты)
- Folder panes (Панели папок).

2.16.2. Схемы

Добавленные в версии 5 *схемы* (Outlines) обеспечивают возможность навигации по сайтам и БД. Схемы являются частью пользовательского интерфейса, которая не только перечисляет различные части приложения либо сайта, но и обеспечивает связи с данными частями. Схемы могут содержать ссылки на представления, формы, фреймсетов, другие БД и т.д. Схемы одинаково хорошо работают как в Web-, так и в Notes-клиентах.

Рассмотрим основные моменты, связанные с созданием схем. Поскольку схемы определяют структуру приложений либо сайтов, с ними можно осуществлять работу двумя способами. Первый способ заключается в том, что создать схему можно до создания любого другого элемента, а затем добавить связи и действия так, как строится остальная часть приложения. Второй способ заключается в том, что схему можно добавить к БД с элементами, которые уже существуют. В любом случае, для того чтобы создать схему, необходимо открыть Domino Designer и выбрать базу данных на панели Design. Нажимаете на Outlines под Shared Code в списке Design и затем нажимаете кнопку New Outline (новая схема).

Диалог свойств схемы показан на рис. 88.

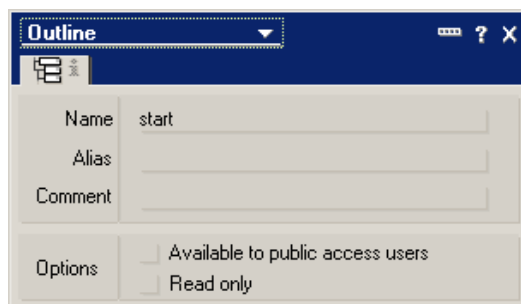


Рис. 88. Диалог свойств схемы

Только одна закладка присутствует в диалоге свойств схемы – закладка Outline Info. Она содержит базовую информацию: имя схемы (Name), псевдоним (Alias), комментарий к схеме (Comment). При установке флажка Available to Public Access Users к схеме получают доступ неавторизованные пользователи; они могут использовать схему для доступа к другим элементам, определенным для общего (public) доступа.

Сердцем схемы является *элемент* (entry). Элементы выполняют действия, соединяются с другими БД или Web-сайтами, открывают представления и папки и т.д. Подобно другим объектам дизайна, элемент имеет диалог свойств, который состоит из закладок Info и Hide When (рис. 89).

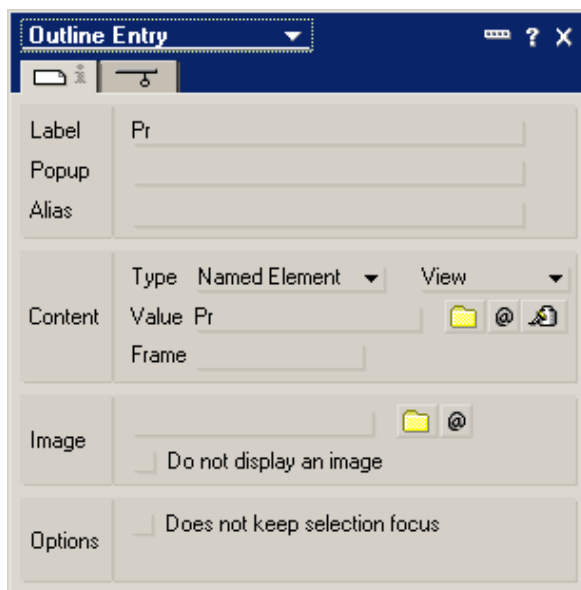


Рис. 89. Диалог свойств элемента

Закладка Info позволяет задать имя (Name), содержание (Content), изображение (Image) и опции (Options) элемента. Подробнее рассмотрим секцию Content. В этой секции три поля: Type, Value и Frame. Выбор конкретного значения в поле Type определяет значения остальных полей секции Content. Возможны пять значений поля Type:

None – может использоваться для метки-заполнителя, типа заголовка иерархической категории;

Action – связывает с элементом действие, базирующееся на @-формулах;

Link – вставляет в элемент связь, ранее скопированную в буфер обмена;

Named Element – список выбора, который позволяет выбрать в качестве значения элемента один из следующих объектов базы: страница, форма, фреймсет, представление, папка, навигатор;

URL – значением элемента является ссылка URL.

Разработанную схему можно внедрить в другие элементы дизайна, такие как страница или форма. Первый способ внедрения заключается в том, что схема внедряется в страницу путем нажатия кнопки Use Outline на рабочей панели схемы. Этот способ можно применить, чтобы задействовать схему в новой странице. Второй способ состоит во внедрении схемы в уже существующую

страницу либо форму. Для этого необходимо выбрать пункт меню Domino-дизайнера Create | Embedded Element | Outline.

Первое, что можно заметить, внедрив схему в страницу, – это то, что страница не обязательно включает все элементы схемы. Для изменения этого эффекта нужно будет поработать со свойствами внедряемой схемы.

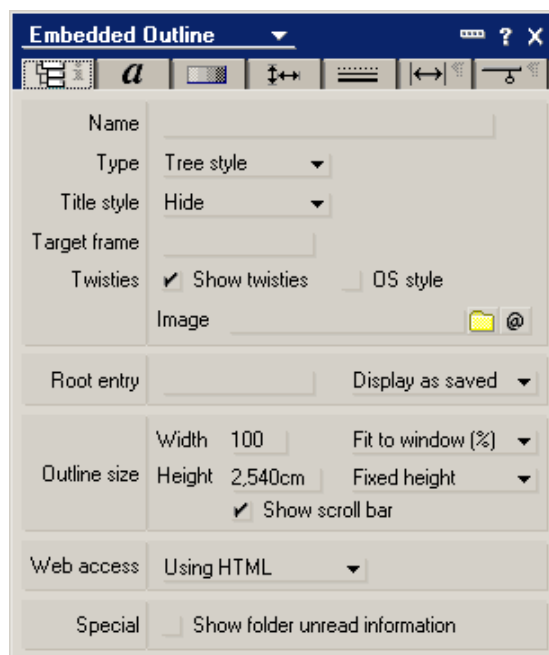


Рис. 90. Свойства внедряемой схемы

Диалог свойств внедренной схемы содержит такие закладки:

1. Info – настройки, определяющие имя, стиль и внешний вид схемы;
2. Font – настройки шрифта каждого уровня схемы, а также цвета каждого из трех состояний: Normal, Selected и Moused;
3. Background – устанавливает цвета фона или изображения для каждого уровня схемы;
4. Layout – определяет высоту, смещение и выравнивание элемента схемы, метку элемента схемы и связанное изображение (если таковое есть) для каждого уровня схемы;
5. Border – устанавливает стиль границ схемы, цвет, эффекты и толщину;
6. Paragraph Margins – содержит стандартные настройки для полей, табуляции и разбиения на страницы;
7. Paragraph Hide When – стандартные настройки, которые включают сокрытие параграфа типом клиента, состоянием документа и формулой.

2.16.3. Фреймсеты

Фреймсеты (Framesets) – это сущности, задающие логику расположения окошек, в которых отображаются навигаторы, страницы, представления, формы и прочие элементы интерфейса пользователя. Фреймсеты взяли свое начало в Web как средство представления содержания более чем одной HTML-страницы одновременно. Фреймосодержащие Web-сайты становятся достаточно распро-

страненным явлением, так как вы можете связать один фрейм (окошко) фрейм-сета с содержанием другого фрейма. Это обеспечивает прекрасный способ навигации по Web-сайту.

Фреймсет делит клиентское окно на два или более фрейма. *Фреймы* – это секции окна, которые могут представлять содержание независимо друг от друга. Каждый фрейм может содержать страницу либо элемент дизайна.

Рассмотрим процесс создание фреймсетов. Прежде чем начать проектирование фреймсета, необходимо его спланировать. При этом следует учесть *схему расположения (layout)* фреймов для фреймсета. Выбор возможных вариантов расположений доступен в диалоге создания нового фреймсета (Create New Frameset). Выпадающий список диалога позволяет выбрать число фреймов для размещения (от 2 до 4). Каждому выбору соответствует 4 возможных варианта размещения выбранного числа фреймов, среди которых предстоит выбрать нужный вариант.

Рассмотрим пример создания фреймсета с тремя фреймами. Для создания нового фреймсета выберите Framesets из списка проекта базы документов в Domino Designer. Нажмите кнопку New Frameset. Откроется диалог создания нового фреймсета (рис. 91).

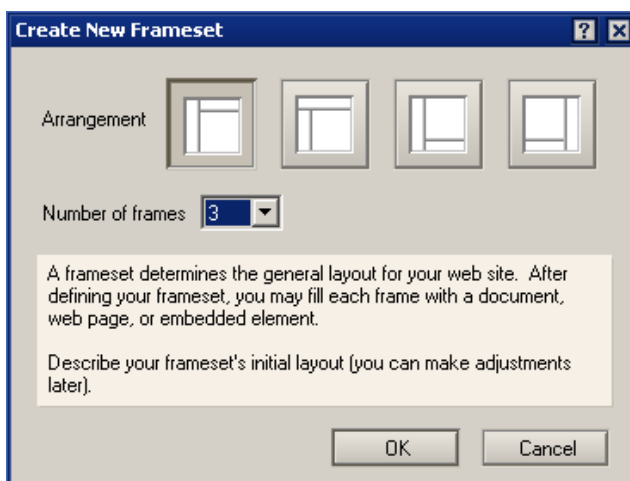


Рис. 91. Создание нового фреймсета

В выпадающем списке Number of frames (число фреймов) выберите значение 3, после чего выберите требуемую схему. Нажмите ОК, и перед Вами появится трехфреймовый фреймсет.

Хотя существует ровно 12 базовых размещений фреймов на фреймсете, отдельный фрейм сам может быть разделен на подфреймы. Создав базовый фреймсет, вы можете разделить любой фрейм на подфреймы либо по вертикали (Split into Columns), либо по горизонтали (Split into Rows). Например, создав базовый трехфреймовый фреймсет, можно добавить еще один фрейм в верхний правый угол с целью размещения на нем логотипа.

Дизайнер фреймсета достаточно прост в использовании. В конечном счете, большая часть действий разворачивается непосредственно в самих фреймах.

Однако стоит хотя бы знать о существовании диалога свойств и некоторых пунктов меню перед переходом к проектированию фреймов.

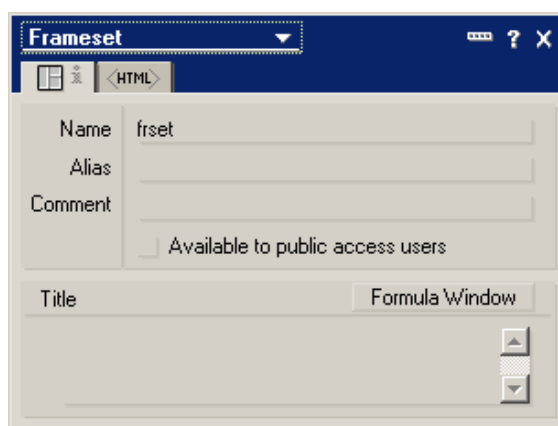


Рис. 92. Проектирование фрейма

Перед тем как сохранить новый фреймсет, нужно придумать ему имя (Name), псевдоним (Alias) и заголовок (Title). В качестве заголовка может выступать обычный текст. Также для задания заголовка может использоваться формула. Если не обеспечить фреймсет заголовком, последний будет в качестве заголовка отображать "Untitled".

После настройки фреймсета требуется выполнить настройку отдельного фрейма (рис. 93).

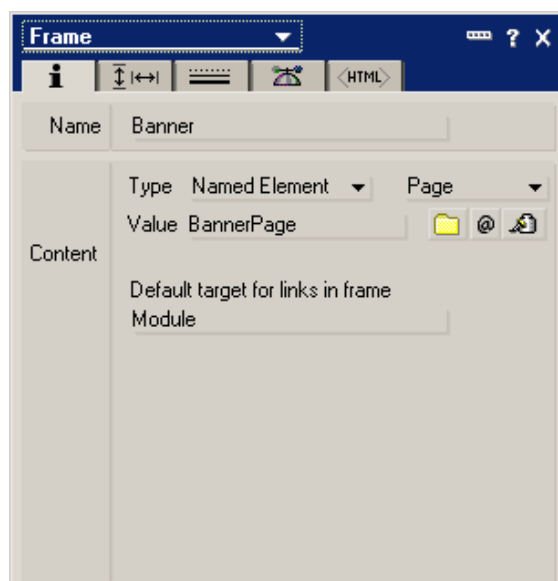


Рис. 93. Настройка фрейма

Любой фрейм имеет имя – Name и содержание (Content). Настройка содержания фрейма напоминает настройку одноименного свойства для схемы. Однако поле Type принимает одно из трех значений:

1. Link – вставляет во фрейм связь, ранее скопированную в буфер обмена;
2. Named Element – список выбора, который позволяет выбрать в качестве значения фрейма один из следующих объектов базы: страница, форма, фреймсет, представление, папка, навигатор;

3. URL – значением фрейма является ссылка URL.

Поле `Default target for links in frame` задает фрейм, в котором будут показываться элементы, ссылки на которые имеются в настраиваемом фрейме.

После разработки фреймсета и настройки отдельных его элементов можно указать данный фреймсет как исходный при открытии базы. Для этого требуется использовать закладку `Launch` свойств базы:

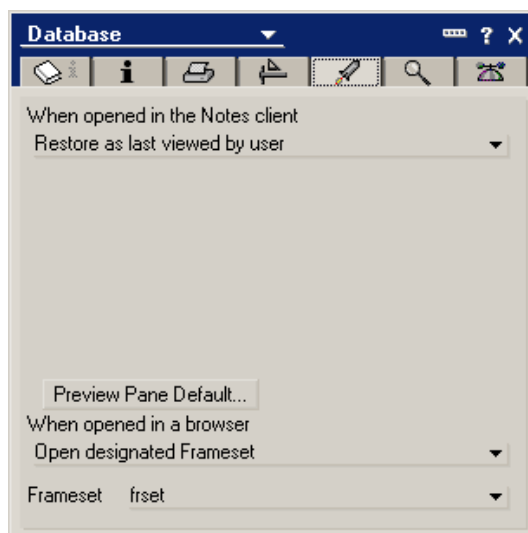


Рис. 94. Указание стартового фреймсета

В одном (или обоих) полях `When Opened...` задайте `Open Designated Frameset`. Выберите название фреймсета. Теперь при открытии базы будет открыт заданный фреймсет.

ЛИТЕРАТУРА

1. Керн С., Линд Д., Разработка приложений в среде Lotus Notes и Domino 6. Подробное руководство. Platinum Edition. . – М.: ДиаСофт, 2005.
2. Клейтон Р., Lotus Notes 5. Учебный курс. – СПб.: Питер, 2000.
3. Мармел Э., Библия пользователя Project 2002. – Вильямс, 2003.
4. Пайрон Т., Использование Project 2002. Специальное издание. – Вильямс, 2003.
5. Фишвик К., Lotus Notes и Domino 6: сертификация для системного администратора. – М.: Кудиц-образ, 2005.