М. Г. Радченко
Е. Ю.Хрусталева

Архитектура и работа с данными «1С:Предприятия 8.2»

Москва
2011

УДК 658.012.011.56:004.42

ББК 65.29 Р15

Р15 Радченко М. Г. Архитектура и работа с данными «1С:Предприятия 8.2» / М. Г. Радченко, Е. Ю. Хрусталева. -М: ООО «1С-Паблишинг», 2011.-268 е.: ил. - (Профессиональная разработка).

ISBN 978-5-9677-1555-6

Книга посвящена углубленному изучению вопросов создания и модификации прикладных решений на платформе системы «1С:Предприятие 8.2». Она является частичной переработкой популярной книги «Профессиональная разработка в системе «1С:Предприятие 8».

В книгу включены материалы, которые описывают общие архитектурные решения, используемые в технологической платформе «1С:Предприятие 8.2». Перечисляются имеющиеся в системе средства разработки, администрирования, а также прикладная функциональность, которая может быть использована в приложениях на этой платформе. Подробно рассматривается система типов и идеология работы с данными. Отдельная глава посвящена клиент-серверному варианту работы. В ней описывается структура кластера серверов, средства управления кластером и принципы исполнения программного кода на сервере и на клиенте.

Книга является очередным изданием в серии «Профессиональная разработка», в которой уже вышли пособия «Реализация прикладных задач в системе «1С:Предприятие 8.2», «Технологии интеграции «1С:Предприятия 8.2», «Разработка управляемого интерфейса», «Инструменты для создания тиражируемых приложений «1С:Предприятия 8.2».

Книга рассчитана на разработчиков, обладающих некоторым навыком создания и модификации прикладных решений в системе «1С:Предприятие 8.2» и желающих повысить свой профессиональный уровень.

Также книга будет интересна IT-специалистам, не занимающимся разработкой, но желающим получить представление о возможностях системы, ее идеологии, архитектуре и реализации конкретных механизмов.

Рассматриваемые в книге инструменты и механизмы описаны исходя из возможностей, предоставляемых версией 8.2.14.519 технологической платформы «1С:Предприятие 8.2».

Книга выпущена под редакцией Максима Радченко.

Группа подготовки издания:

Дизайн обложки - Кира Федотова.

Литературное редактирование, корректура - Елена Семененко. Верстка,

препресс - Ольга Шестакова.

© ООО «1С-Паблишинг», 2011

ISBN 978-5-9677-1555-6

Оглавление

Введение **9**
Глава 1. Архитектура «1С:Предприятия» **11**
Платформам прикладные решения 12
Средства разработки в составе платформы 12
Метаданные - способ описания прикладного решения 13
Метаданные и встроенный язык 14
Подсистемы 15
Создание прикладных решений на основе модели 16
Объектные и необъектные данные 18
Три способа представления данных 19
Сквозная типизация 20
Поддержка составных типов данных 20
Смешанный подход к манипулированию данными 21
Встроенный язык 22
Язык запросов 23
Прикладные решения, не зависимые от используемого хранилища данных 24
Управляемый интерфейс 24
Командный интерфейс 28
Использование информации из метаданных 31
Расширения форм и элементов управления 31
Генерация форм по умолчанию 31
Клиентские приложения 32

Подключение через Интернет ... 33
Файловый и клиент-серверный варианты работы ... 34
Кластер серверов «1С:Предприятия» ... 35
Выполнение основной функциональности на сервере ... 36
Использование встроенного языка на клиенте ... 37
Поддержка различных хранилищ данных ... 38
Работа под управлением различных операционных систем ... 38
Обновление прикладных решений ... 40
Интернационализация ... 40
Глава 2. Функциональность «1С:Предприятия» ... 41
Обзор функциональности прикладных решений ... 41
Справочники ... 41
Документы ... 42
Механизм характеристик ... 43
Механизм сведений ... 43
Механизм учета движения средств ... 44
Механизм бухгалтерского учета ... 44
Механизм сложных периодических расчетов ... 45
Механизм бизнес-процессов ... 45
Механизм анализа данных и прогнозирования ... 46
Сложные отчеты, использующие систему компоновки данных ... 46
Механизмы презентации текстовых и аналитических данных ... 47
Средства интеграции и механизмы обмена данными ... 52
Web-сервисы ... 53
Web-расширение ... 53
Механизм полнотекстового поиска ... 54
Механизм криптографии ... 55
Механизм разделения данных ... 56
Внешние источники данных ... 56
Обзор функциональности средств разработки ... 57
Объекты конфигурации и инструменты для их редактирования ... 57
Конструкторы и редакторы ... 59
Синтакс-помощник ... 61
Отладчик и режим замера производительности ... 61
Проверка конфигурации ... 61
Сравнение/объединение конфигураций ... 62
Редактирование текстов интерфейса ... 62
Групповая разработка прикладных решений ... 62
Поставка и поддержка прикладных решений ... 63
Создание дистрибутивов ... 63
Функциональные опции ... 63
Показатели производительности и режим имитации задержек при вызове сервера ... 64

1С:Библиотека стандартных подсистем 8.2 ... 65
Обзор функциональности средств администрирования ... 65
Список пользователей ... 65
Механизмы аутентификации ... 66
Механизм заданий ... 67
Система прав доступа ... 67
Ограничение прав на уровне записей и полей ... 68
Журнал регистрации ... 68
Загрузка/выгрузка информационной базы ... 69
Утилита администрирования кластера серверов ... 69
«1С:Корпоративный инструментальный пакет» ... 70
Глава 3. Использование встроенного языка ... 71
Введение ... 71
Модули конфигурации ... 74
Контекст исполнения модулей ... 76
Общий модуль ... 77
Модуль управляемого приложения ... 78
Модуль сеанса ... 82
Модуль внешнего соединения ... 82
Модуль объекта ... 83
Модуль менеджера объекта ... 86
Модуль управляемой формы ... 88
Модуль команды ... 94
Работа с метаданными ... 95
Глава 4. Работа с данными ... 103
Объектные и необъектные данные ... 103
Объектные данные ... 104
Модель хранения данных ... 104
Ссылка ... Ю6
Объект ... 112
Необъектные данные ... 125
Модель хранения данных ... 125
Набор записей ... 127
Система типов ... 133
Типы значений и типы данных ... 133
Типы значений ... 135
Примитивные типы ... 135
Типы, образуемые в прикладном решении ... 160
Типы данных ... 177
Числовые типы данных ... 178
Строковые типы данных ... 179
Типы данных, описывающие значения типа «Дата» ... 181

Составной тип данных ... 182
Наборы типов ... 187
Транзакции ... 188
Использование явного вызова транзакций ... 190
Обработка ошибок базы данных в транзакции ... 191
Вложенный вызов транзакций ... 194
Влияние транзакций на работу программных объектов ... 197
Восстановление признака проведенности документа ... 201
Очистка ссылки ... 202
Очистка кода/номера объекта ... 202
Механизм управляемых блокировок ... 203
Возможные проблемы при многопользовательском доступе к одним и тем же данным 203
Уровни изоляции транзакций ... 206
Режим управляемых блокировок ... 207
Установка режима управления блокировкой данных в конфигурации ... 209
Установка управляемых блокировок ... 209
Глава 5. Клиент-серверный вариант работы ... 217
Общие сведения о клиент-серверном варианте работы ... 217
Программные компоненты клиент-серверной архитектуры «ЮПредприятия» ... 218
Варианты использования ... 221
Работа под управлением различных операционных систем ... 224
Кластер серверов ... 226
Состав кластера ... 226
Сеансы и соединения ... 228
Виды соединений ... 228
Виды сеансов ... 232
Масштабируемость кластера ... 232
Перераспределение сервисов между менеджерами кластера ... 232
Использование нескольких рабочих процессов ... 234
Использование нескольких рабочих серверов ... 234
Отказоустойчивость кластера ... 234
Резервирование кластера ... 234
Резервирование рабочих процессов ... 235
Устойчивость к обрыву канала связи ... 236
Управление кластером серверов ... 237
Утилита администрирования кластера серверов ... 237
Работа со списком администраторов центрального сервера ... 238
Работа со списком кластеров центрального сервера ... 238
Работа со списком менеджеров кластера ... 240
Работа со списком администраторов кластера ... 241
Работа со списком рабочих серверов кластера ... 242
Работа со списком рабочих процессов ... 242
Работа со списком информационных баз ... 244

Работа со списком сеансов 245
Работа со списком соединений 246
Динамическое перераспределение нагрузки 247
Программные средства администрирования кластера серверов 248
Безопасность 249
Многоуровневая аутентификация 249
Шифрование передаваемых данных 251
Работа встроенного языка на сервере 255
Организация выполнения кода на сервере или на клиенте 255
Клиентские и серверные процедуры в модуле формы, команды 255
Клиентские и серверные обработчики событий формы, команды 258
Вызов процедур из серверных модулей 263
Передача параметров на сервер и возврат значений 265
Привилегированный и безопасный режимы исполнения кода 265
Привилегированный режим 266
Безопасный режим 268

Введение

В книге собрана и систематизирована наиболее важная информация, которая может понадобиться разработчику прикладных решений «^Предприятия 8.2».

Несмотря на то, что в одной книге невозможно рассмотреть все ситуации, возникающие при разработке прикладных решений, на большинство вопросов в книге можно найти ответы. Причем читать будет одинаково интересно как начинающим пользователям, так и продвинутым разработчикам.

При написании мы стремились к тому, чтобы книга стала серьезным инструментом для разработчиков: к ней всегда можно было бы обратиться в случае затруднений, узнать что-то новое о хорошо известной предметной области или познакомиться с новым взглядом на привычные вещи.

При подготовке материала были использованы различные источники информации:

- опыт преподавания на учебных курсах по платформе и прикладным решениям «1С:Предприятия 8»;
- опыт внедрения прикладных решений;
- опыт, накопленный разработчиками фирмы «1С»;
- материалы информационно-технологической поддержки (ИТС);
- материалы форума партнеров-разработчиков на сайте http://partners.v8.lc.ru;
- общение на партнерских семинарах, проводимых фирмой «1С».

Глава 1. Архитектура «1 С: Предприятия»

Система «1С:Предприятие» в широком смысле представляет собой совокупность четырех составляющих (рис. 1.1):

- технологической платформы;
- прикладных решений различного масштаба и различной направленности, созданных на основе технологической платформы;
- методики создания прикладных решений;
- информационно-технологической поддержки пользователей и разработчиков.



Рис. 1.1. Структура «1С:Предприятия»

Такая архитектура продиктована прежде всего задачами, которые призвана решать система «1С:Предприятие».

Во-первых, система должна обеспечивать высокий уровень адаптируемости прикладных решений под требования заказчика.

Во-вторых, система должна обеспечивать изменение готового прикладного решения разработчиком, не участвовавшим в его создании. Это особенно важно для прикладных решений в сфере экономических задач, где существенная часть разработчиков не создают собственные прикладные решения, а дорабатывают и развивают существующие типовые решения.

В-третьих, система должна обеспечивать эффективное использование компьютерных технологий и платформ, не требуя при этом глубоких специальных знаний от разработчика.

В-четвертых, система должна обеспечивать стандартизацию разработки.

Таким образом, можно сказать, что «1С:Предприятие» не является универсальным средством программирования. Система обладает достаточно широкими возможностями, однако ее архитектура и конкретная реализация механизмов и технологий платформы продиктованы прежде всего необходимостью решения специализированных задач по созданию бизнес-приложений и требованиями, предъявляемыми к самой системе.

Платформа и прикладные решения

Основным концептуальным решением, отличающим систему «1С:Предприятие» от универсальных средств программирования, является четкое разделение на платформу и прикладное решение.

Прикладное решение «1С:Предприятия» является самостоятельной сущностью и может выступать в качестве отдельного программного продукта. Однако создание, модификация и собственно функционирование прикладного решения невозможны без использования технологий и механизмов платформы. Поэтому платформа поставляется с каждым комплектом «1С: Предприятия».

Средства разработки в составе платформы

Прикладные решения «ЮПредприятия» являются открытыми. Благодаря этому клиент с помощью разработчика или собственными силами может модифицировать и настраивать любое прикладное решение под себя. Исключение составляют прикладные решения, которые выпускаются в так называемой «базовой версии». Кроме прочих ограничений это означает невозможность самостоятельной модификации прикладного решения пользователем.

Для модификации прикладных решений не требуется использовать какие- либо отдельные программные продукты - все средства разработки входят в состав технологической платформы.

Можно сказать, что платформа состоит из двух составляющих (рис. 1.2):

- среда исполнения,
- среда разработки.



Рис. 1.2. Структура технологической платформы «1С:Предприятия»

Таким образом обеспечивается высокий уровень адаптируемости прикладных решений под требования заказчика.

Метаданные - способ описания прикладного решения

Прикладное решение «1С:Предприятия» не пишется в прямом смысле слова на языке программирования. При создании прикладных решений «1С:Предприятия» используется более абстрактная технология -технология метаданных.

Метаданные представляют собой иерархическую структуру объектов, полностью описывающую все прикладное решение. Эта структура называется «дерево объектов конфигурации» (рис. 1.3).

Среда исполнения «1С:Предприятия» исполняет, «проигрывает» метаданные, аналогично тому, как операционная система исполняет код привычной программы.

Отличительной особенностью технологии метаданных является использование визуального конструирования прикладного решения. Вместо кропотливого написания кода разработчик просто добавляет визуальными средствами новый объект конфигурации в прикладное решение и получает сразу же описание нужных типов, структур данных, описание наборов прав, связей между объектами, информацию об особенностях их поведения, визуального представления и т.д.



Рис. 1.3. Дерево объектов конфигурации прикладного решения

Метаданные и встроенный язык

Все прикладное решение представляется не в виде строк с инструкциями на языке программирования, а в виде иерархической структуры объектов конфигурации. При этом разработчик использует встроенный язык и язык запросов для того, чтобы описать специфические алгоритмы поведения тех или иных объектов конфигурации в различные моменты исполнения прикладного решения.

Использование встроенного языка при разработке прикладных решений ограничено в основном решением тех задач, которые действительно требуют алгоритмического описания, например, расчета налогов, проверки корректности введенных данных и пр. Основная же структура прикладного решения описывается структурой объектов конфигурации.

Почти все объекты конфигурации содержат модули, в которых и могут быть описаны алгоритмы на встроенном языке. Эти модули будут вызываться средой исполнения в конкретные, заранее определенные моменты работы прикладного решения - события. Таким образом, можно сказать, что использование встроенного языка в прикладных решениях носит событийный характер.

Подсистемы

Платформа «1С: Предприятие» позволяет выделить в прикладном решении отдельные функциональные части - подсистемы, в совокупности представляющие все прикладное решение. Подсистемы могут иметь иерархическую структуру, т.е. одна подсистема может включать в себя несколько других подсистем.

Структура подсистем определяет всю функциональность прикладного решения и служит основой для построения глобального командного интерфейса приложения (рис. 1.4).

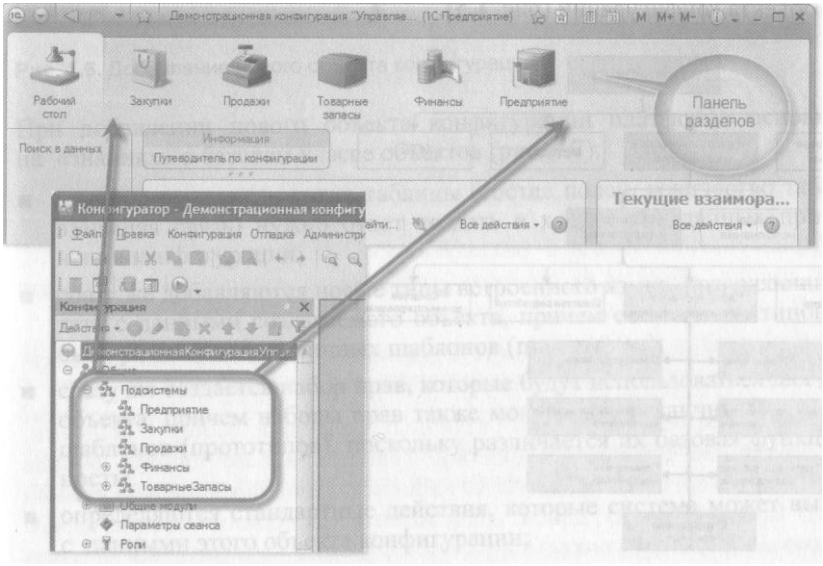


Рис. 1.4. Глобальный интерфейс прикладного решения

Разработка практически любого прикладного решения начинается с проектирования состава подсистем, так как структура подсистем (разделов в интерфейсе приложения) - это первое, что увидит пользователь при знакомстве с прикладным решением.

Каждый объект конфигурации, описывающий данное прикладное решение, . может быть отнесен к одной или нескольким подсистемам. Таким образом, наличие подсистем позволяет распределить объекты конфигурации по логически связанным с ними разделам и предоставить пользователю удобный интерфейс, в котором ему будут видны только те объекты (т.е. та функциональность прикладного решения), с которыми он работает.

Создание прикладных решений на основе модели

Важной особенностью системы «1С:Предприятие» является то, что для описания структуры прикладного решения разработчик использует не произвольные, а строго определенные объекты конфигурации.

Платформа «1С:Предприятие» содержит ограниченный набор прототипов (шаблонов) объектов конфигурации. Среди этих шаблонов есть, например, шаблон справочника, документа, регистра накопления, бизнес-процесса и т. д.

Каждый такой шаблон (прототип) содержит определенную базовую реализацию объекта конфигурации (рис. 1.5).

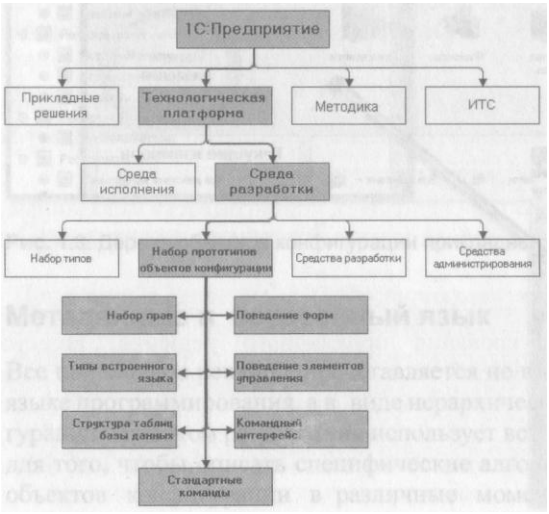


Рис. 1.5. Состав прототипа (шаблона) объекта конфигурации

■

Когда разработчик добавляет в дерево объектов конфигурации новый объект конфигурации, этот объект наследует базовую реализацию прототипа (рис. 1.6):



Рис. 1.6. Добавление нового объекта конфигурации

При добавлении нового объекта конфигурации платформа основывается на «знаниях» о базовом классе объектов (рис. 1.7):

- платформа «знает», какие таблицы (состав полей, количество таблиц, их взаимная связь) нужно будет создать в хранилище данных при сохранении конфигурации;
- сразу же добавляются новые типы встроенного языка, позволяющие работать с данными создаваемого объекта, причем состав этих типов может быть разным для различных шаблонов (прототипов);
- сразу же создается набор прав, которые будут использоваться для данного объекта, причем наборы прав также могут быть различными для разных шаблонов (прототипов), поскольку различается их базовая функциональность;
- определяются стандартные действия, которые система может выполнять с данными этого объекта конфигурации;
- и т. д.

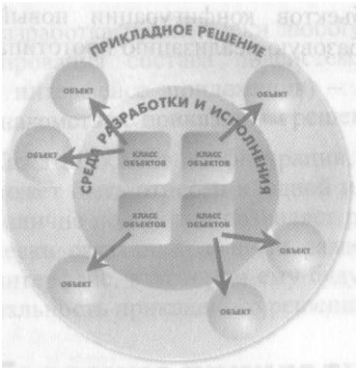


Рис. 1.7. Добавление нового объекта конфигурации

Благодаря этому разработчик, не производя никаких дополнительных действий, тут же может запустить прикладное решение и работать с только что добавленным объектом - базовая реализация объекта, унаследованная от прототипа (шаблона), обеспечит выполнение всех необходимых типовых действий.

Таким образом, несмотря на то, что каждое прикладное решение обладает собственной «индивидуальностью», все они созданы по определенной модели, с использованием объектов конфигурации, которые обладают одинаковой базовой реализацией. Такой подход значительно упрощает модификацию прикладных решений разработчиками, которые не участвовали в их создании.

Объектные и необъектные данные

В «1С:Предприятии» принято разделять все прикладные данные на те, которые имеют объектную природу (объектные данные) и не имеют объектной природы (необъектные данные).

Примерами объектных данных могут служить данные справочников, документов. Необъектными данными являются, например, данные регистров.

Подобное деление определяет два различных подхода к работе с данными. Данные, имеющие объектную природу, хранятся в базе данных в виде объектов. Например, объектом является элемент справочника или конкретный документ. Каждый объект ценен для системы уже одним фактом своего существования и имеет уникальный идентификатор - ссылку. Могут измениться любые данные объекта, но это будет все тот же объект. Удалив объект,

его нельзя создать заново. Даже если для нового объекта будут установлены те же данные, это будет уже другой объект с точки зрения «1С:Предприятия», обладающий другим уникальным идентификатором.

Данные, имеющие необъектную природу, хранятся в базе данных в виде записей. Каждая запись полностью описывается значениями своих полей и не имеет какого-либо уникального идентификатора. Поэтому для «^Предприятия» собственно факт существования записи не важен. Можно удалить запись, а затем создать новую, с такими же значениями полей - состояние базы данных с точки зрения логики прикладного решения от этого не изменится.

Подробнее об объектных и необъектных данных можно прочитать в разделе «Объектные и необъектные данные» на стр. 103.

Три способа представления данных

Для всех прикладных данных (как объектных, так и необъектных) «1С:Предприятие» поддерживает три способа представления данных (рис. 1.8):

- хранение в базе данных,
- представление во встроенном языке,
- отображение в формате XML.



Рис. 1.8. Три способа представления данных

В базе данных информация хранится в виде объектов базы данных или в виде отдельных записей (в зависимости от ее природы - объектной или необъектной).

Данные, хранимые в базе данных, могут быть считаны в объекты встроенного языка для их просмотра или изменения и записаны обратно в базу данных.
В то же время объекты встроенного языка могут быть сериализованы в/из элементы/ов XML.
Представление данных в формате XML используется

при обмене данными в распределенных информационных базах, а также может использоваться при взаимодействии с другими информационными системами.

Важно отметить, что все три способа представления данных используют одну и ту же систему понятий, благодаря чему от разработчика не требуется специальных усилий для преобразования данных из одного представления в другое.

Сквозная типизация

Важной особенностью работы с данными является то, что в «1С:Предприятии» реализована общая система типов встроенного языка, полей баз данных и интерфейса. Иными словами, разработчик одинаковым образом определяет поля базы данных, переменные встроенного языка, реквизиты форм и одинаковым образом работает с ними (рис. 1.9).



Рис. 1.9. Общая система типов

В результате разработчику не приходится заботиться о преобразованиях между типами данных, поддерживаемыми той или иной СУБД, типами, поддерживаемыми встроенным языком, и типами, используемыми для построения интерфейсных решений.

Подробнее о типах данных, используемых системой «1С:Предприятие», можно прочитать в разделе «Система типов» на стр. 133.

Поддержка составных типов данных

Важной особенностью модели данных «1С:Предприятия» является возможность использования реквизитов объектов конфигурации, имеющих составной тип. Например, в расходной накладной в качестве покупателя может быть указано либо юридическое лицо из справочника организаций, либо физическое лицо из справочника частных лиц. Соответственно, при проекти

ровании базы данных разработчик может определить реквизит, который будет хранить значение любого из этих типов.
Подробнее о составном типе можно прочитать в разделе «Составной тип данных» на стр. 182.

Смешанный подход к манипулированию данными

«1С: Предприятие» обеспечивает два способа доступа к данным - объектный (для чтения и записи) и табличный (только для чтения).
В объектной модели разработчик оперирует объектами встроенного языка. В этой модели обращения к объекту, например документу, происходят как к единому целому - он полностью загружается в память, вместе с вложенными таблицами, к которым можно обращаться средствами встроенного языка как к коллекциям записей и т.д. (рис. 1.10).

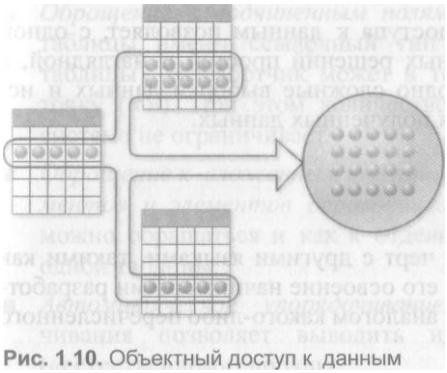


Рис. 1.10. Объектный доступ к данным

При манипулировании данными в объектной модели обеспечивается сохранение целостности объектов, кеширование объектов, вызов соответствующих обработчиков событий и т. д.
В табличной модели все множество объектов того или иного класса представляется как совокупность связанных между собой таблиц, к которым можно обращаться при помощи запросов - как к отдельной таблице, так и к нескольким таблицам во взаимосвязи (рис. 1.11).

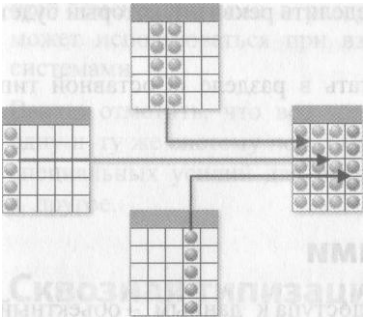


Рис. 1.11. Табличный доступ к данным

В этом случае разработчик получает доступ к данным сразу нескольких объектов, что очень удобно для анализа больших объемов данных, например, при создании отчетов. Однако в силу того, что данные, выбираемые таким способом, содержат не все, а лишь некоторые реквизиты анализируемых объектов, табличный способ доступа не позволяет изменять эти данные.

Сочетание табличного и объектного доступа к данным позволяет, с одной стороны, сделать разработку прикладных решений простой и наглядной, а с другой стороны, получать сколь угодно сложные выборки данных и использовать возможности агрегирования полученных данных.

Встроенный язык

Встроенный язык имеет много общих черт с другими языками, такими как Pascal, Java Script, Basic, что облегчает его освоение начинающими разработчиками. Однако он не является прямым аналогом какого-либо перечисленного языка.

Вот лишь некоторые наиболее значимые особенности встроенного языка:

- мягкая типизация (тип переменной определяется типом значения, которое она содержит, и может изменяться в процессе работы);
- отсутствие программного описания прикладных типов (они создаются при добавлении объектов метаданных);
- не требуется предварительное описание процедур/функций, если их вызов выполняется раньше их описания;
- событийная ориентированность встроенного языка;
- поддерживается обработка исключительных ситуаций;
- все операторы имеют как русское, так и английское написание, которое можно использовать одновременно;

- используется интерпретатор с предварительной компиляцией (перед исполнением модули, содержащие текст на встроенном языке, преобразуются во внутренний код);
- кеширование скомпилированных модулей в памяти.

Язык запросов

Язык запросов основан на SQL, но при этом содержит значительное количество расширений, ориентированных на отражение специфики финансово-экономических задач и на максимальное сокращение усилий по разработке прикладных решений.

Важной особенностью языка запросов является то, что он предоставляет доступ к данным только на чтение и использует те же типы данных, что и встроенный язык.

Можно перечислить наиболее существенные возможности, реализуемые языком запросов:

- *Обращение к подчиненным полям через точку.* Если поля какой-либо таблицы имеют ссылочный тип (хранят ссылки на объекты другой таблицы), разработчик может в тексте запроса ссылаться на них через точку («.»). При этом количество уровней вложенности таких ссылок система не ограничивает.
- *Обращение к вложенным таблицам, (например, табличным частям документов и элементов справочников).* К вложенным табличным частям можно обращаться и как к отдельным таблицам, и как к целым полям одной таблицы.
- *Автоматическое упорядочивание.* Режим автоматического упорядочивания позволяет выводить информацию в наиболее правильном («естественном») порядке.
- *Многомерное и многоуровневое формирование итогов.* Итоги и подитоги формируются с учетом группировки и иерархии, обход уровней может выполняться в произвольном порядке с подведением подитогов. Обеспечивается корректное построение итогов по временным измерениям.
- *Поддержка виртуальных таблиц.* Виртуальные таблицы, предоставляемые системой, позволяют получить практически готовые данные для большинства прикладных решений без необходимости составления сложных запросов. Например, такая виртуальная таблица может предоставить данные по остаткам товаров в разрезе периодов на какой-то момент времени. При этом виртуальные таблицы максимально используют хранимую информацию, например, ранее рассчитанные итоги и т. д.

- *Стандартные SQL операции.* В языке запросов поддерживаются стандартные для SQL операции, такие как объединение (Union), соединение (Join) и т. д.
- *Использование временных таблиц.* Временные таблицы позволяют повысить производительность запросов, в некоторых случаях снизить количество блокировок и сделать текст запроса более легким для восприятия.
- *Пакетные запросы.* Пакетные запросы позволяют более удобно работать с временными таблицами за счет того, что создание временной таблицы и ее использование помещаются в один запрос. Пакетный запрос представляет собой последовательность запросов, разделенных символом «;». Запросы исполняются один за другим. Результатом выполнения пакетного запроса в зависимости от используемого метода будет являться либо результат, возвращаемый последним запросом пакета, либо массив результатов всех запросов пакета в той последовательности, в которой следуют запросы в пакете.

Прикладные решения, не зависимые от используемого хранилища данных

Платформа изолирует разработчика от понятий и подробностей более низкоуровневых технологий. При создании прикладных решений разработчик «1С:Предприятия» не обращается к базе данных напрямую. Непосредственно он работает с платформой «1С:Предприятие». При этом он может:

ш описывать структуры данных в конфигураторе,

- манипулировать данными с помощью объектов встроенного языка,
- составлять запросы к данным, используя язык запросов.

Платформа «1С:Предприятие» обеспечивает операции исполнения запросов, описания структур данных и манипулирования данными, транслируя их в соответствующие команды. Это могут быть команды одной из сторонних СУБД, поддерживаемых «1С:Предприятием», в случае клиент-серверного варианта работы или команды собственной СУБД для файлового варианта работы.

Управляемый интерфейс

В платформе «1С:Предприятие» реализован современный эргономичный пользовательский интерфейс, рассчитанный на комфортную работу с системой в течение продолжительного времени, в том числе и неподготовленных пользователей, имеющих небольшой опыт работы с компьютером. Удобный и интуитивно понятный интерфейс позволяет пользователям выпол-

пять большой объем функционала, даже не пользуясь при этом инструкциями и документацией.

Пользовательский интерфейс «1С:Предприятия» построен по принципу декларативного описания. Он называется «управляемым» потому, что с помощью прав и ролей пользователей, функциональных опций и настроек пользователя платформа автоматически строит командный интерфейс, управляемые формы и отчеты.

Пользовательский интерфейс «1С:Предприятия» можно охарактеризовать как «задаче-ориентированный», позволяющий в каждом отдельном окне приложения (основном или вспомогательном) решать конкретную задачу.

Основное окно приложения предназначено для навигации по прикладному решению и вызова различных команд.

Вспомогательное окно предназначено для работы с объектами информационной базы (например, с документами или элементами справочников), построения отчетов или выполнения обработок данных.

В основном окне приложения пользователю предоставляется вся структура прикладного решения (панель разделов) и рабочий стол (рис. 1.12).

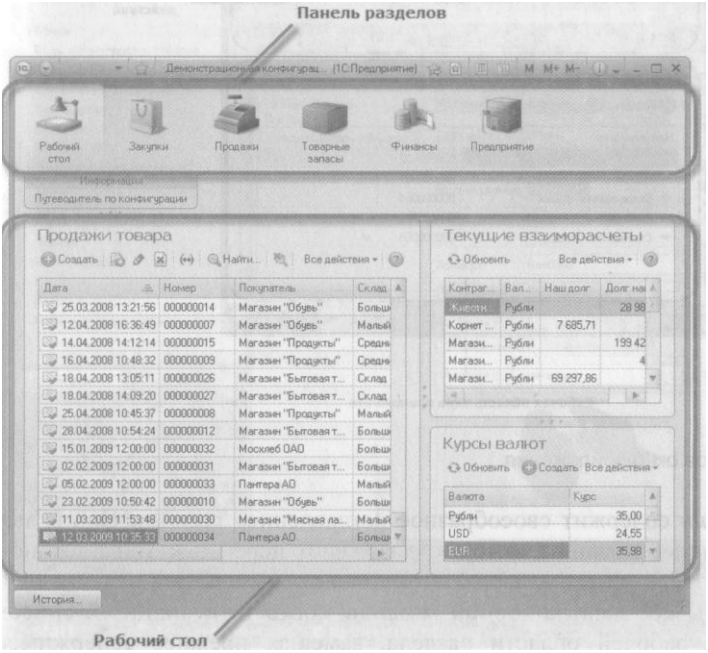


Рис. 1.12. Основное окно приложения

Рабочий стол - это стандартный раздел программы, содержащий часто используемые документы, отчеты, справочники и т. п. Этот раздел автоматически активизируется при запуске прикладного решения. Это своеобразный «помощник» пользователя. Рабочий стол вводит пользователя в курс дел, отвечает на его вопросы и позволяет ему быстро включиться в работу.

Панель разделов — это наиболее крупное разделение функциональности прикладного решения. Она расположена в верхней части основного окна и соответствует верхнему уровню подсистем, добавленных в конфигурацию. С ее помощью осуществляется переход к другим разделам программы.

При активизации раздела вся функциональность соответствующей подсистемы, включая вложенные подсистемы, представляется пользователю в виде команд в двух панелях: панели навигации и панели действий (рис. 1.13).

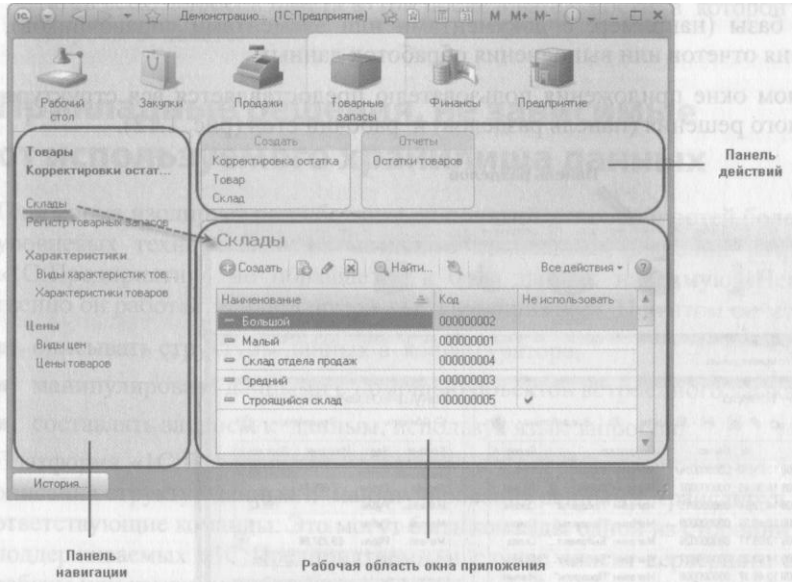


Рис. 1.13. Основное окно приложения

Панель навигации содержит своеобразное «оглавление» раздела. Она включает навигационные команды, которые позволяют перейти к той или иной точке этого раздела. Как правило, это команды перехода к каким-либо спискам. При вызове навигационной команды соответствующий ей список отображается в рабочей области раздела, замещая прежнее содержимое рабочей области.

Панель действий содержит наиболее востребованные в повседневной работе команды, позволяющие быстро создавать новые объекты, выполнять типовые обработки или строить популярные отчеты. Такие команды называются командами действия, поскольку их вызов приводит к открытию нового, вспомогательного окна приложения и на некоторое время переключает пользователя на выполнение другой задачи.

Вспомогательные окна приложения открываются при вызове команд ввода новых и редактирования существующих объектов, а также при открытии отчетов и обработок. Эти окна отображаются на экране компьютера независимо от основного окна приложения (рис. 1.14).

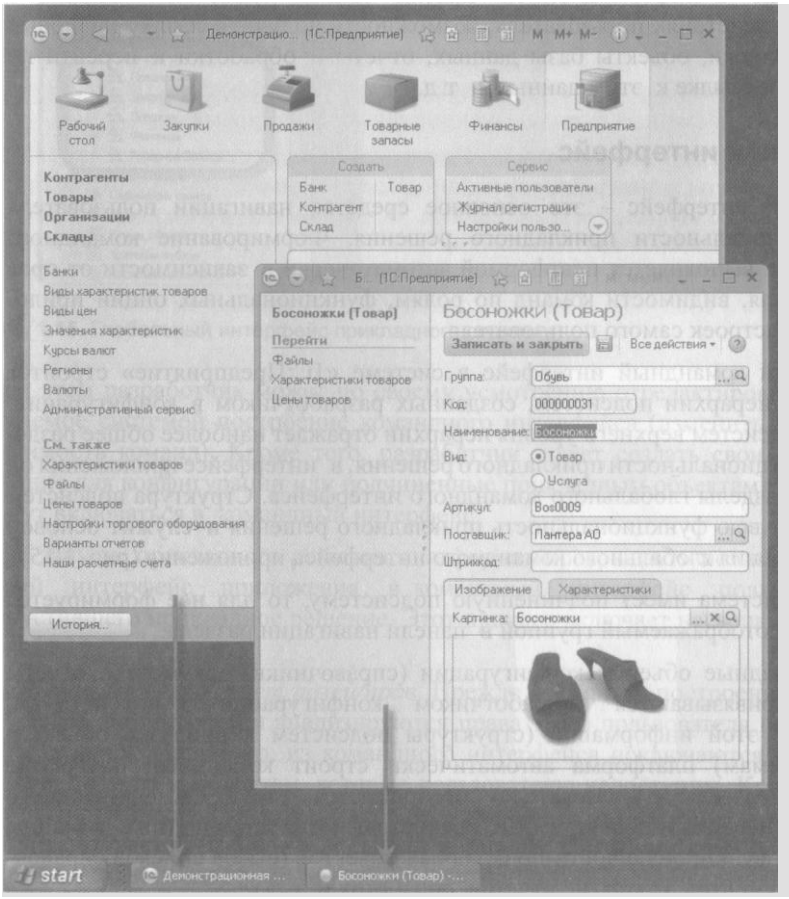


Рис. 1.14. Основное и вспомогательное окно приложения

Пользователь может открывать любое необходимое ему количество вспомогательных окон. Для удобства ввода данных и защиты от ошибок вспомогательное окно может быть открыто в режиме блокирования окна- владельца.

Основное окно приложения также включает в себя главное меню системы и набор вспомогательных команд (калькулятор, календарь и пр.). А также предоставляет различные сервисные возможности, такие как: просмотр истории работы пользователя и возможность перемещаться по разделам прикладного решения, с которыми уже работал пользователь; добавление разделов, списков, объектов базы данных, отчетов и обработок в избранное; отображение в информационной панели (внизу окна) оповещений о последних действиях пользователя; возможность получить ссылку на любые разделы, списки, объекты базы данных, отчеты и обработки и перейти по полученной ссылке к этим данным и т. д.

Командный интерфейс

Командный интерфейс - это основное средство навигации пользователя по функциональности прикладного решения. Формирование командного интерфейса выполняется платформой автоматически в зависимости от прав пользователя, видимости команд по ролям, функциональных опций приложения и настроек самого пользователя.

Глобальный командный интерфейс в системе «1С:Предприятие» строится на основе иерархии подсистем, созданных разработчиком в конфигурации. Состав подсистем верхнего уровня иерархии отражает наиболее общее разделение функциональности прикладного решения, в интерфейсе приложения он образует разделы глобального командного интерфейса. Структура подсистем определяет всю функциональность прикладного решения и служит основой для построения глобального командного интерфейса приложения (рис. 1.15).

Если подсистема имеет подчиненную подсистему, то для нее формируется подраздел, отображаемый группой в панели навигации раздела.

Все прикладные объекты конфигурации (справочники, документы, отчеты и т.д.) привязываются разработчиком конфигурации к подсистемам. На основе этой информации (структуры подсистем и привязки объектов к подсистемам) платформа автоматически строит командный интерфейс для пользователя. Пользователю отображается структура прикладного решения (иерархия подсистем) и предоставляются стандартные команды доступа к функциональности прикладных объектов (вызов списков справочников, документов, открытие отчетов, обработок и т. д.).

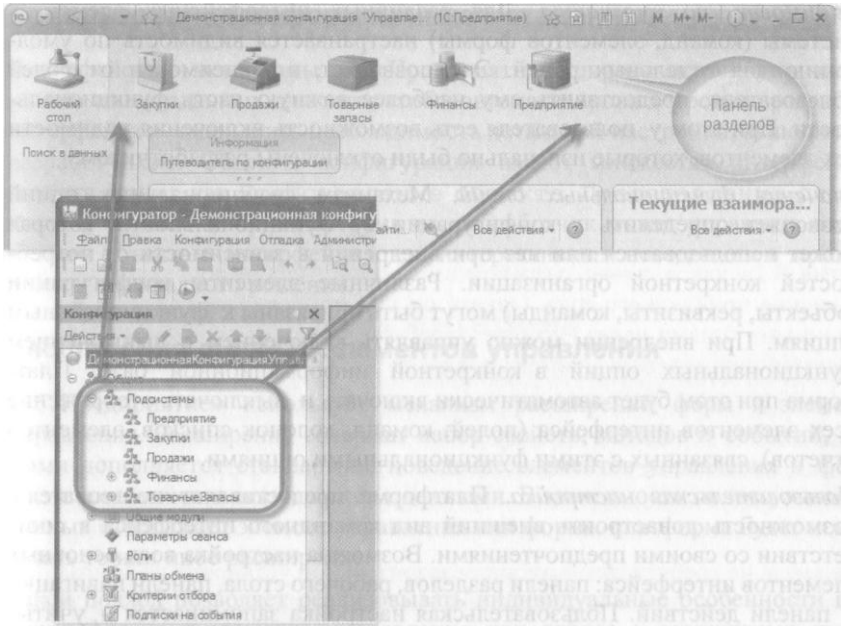


Рис. 1.15. Глобальный интерфейс прикладного решения

При этом разработчик может по своему усмотрению отредактировать предлагаемое системой построение командного интерфейса (изменить порядок, видимость команд). Кроме того, разработчик может создать свои команды (общие для конфигурации или подчиненные прикладным объектам), которые будут включаться в командный интерфейс.

При запуске прикладного решения платформа «превращает» полный командный интерфейс приложения в командный интерфейс пользователя, запустившего прикладное решение. Этот процесс включает несколько этапов (рис. 1.16):

- *Автоматический учет ролей/прав.* Прежде всего, при построении интерфейса автоматически анализируются права этого пользователя на доступ к данным. Например, из командного интерфейса исключаются команды для доступа к объектам, которые пользователю недоступны. Кроме того, в системе существует возможность установки прав на отдельные реквизиты объектов. Используя эту информацию, платформа сама исключает или делает недоступными для редактирования соответствующие поля в формах, а также исключает колонки из списков и отчетов.

- *Видимость по умолчанию.* Для различных интерфейсных элементов системы (команд, элементов формы) настраивается видимость по умолчанию для отдельных ролей. Это позволяет, в зависимости от ролей пользователя, предоставить ему наиболее важную часть функциональности. При этом у пользователя есть возможность включения видимости тех элементов, которые изначально были отключены разработчиком.
- *Значения функциональных опций.* Механизм функциональных опций позволяет определить в конфигурации ту функциональность, которая может использоваться или нет при внедрении в зависимости от потребностей конкретной организации. Различные элементы конфигурации (объекты, реквизиты, команды) могут быть привязаны к функциональным опциям. При внедрении можно управлять включением и выключением функциональных опций в конкретной информационной базе. Платформа при этом будет автоматически включать и выключать отображение всех элементов интерфейса (полей, команд, колонок списков, элементов отчетов), связанных с этими функциональными опциями.
- *Пользовательская настройка.* Платформа предоставляет пользователю возможность донастроить внешний вид командного интерфейса в соответствии со своими предпочтениями. Возможна настройка всех основных элементов интерфейса: панели разделов, рабочего стола, панели навигации и панели действий. Пользовательская настройка запоминается и учитывается при следующем запуске приложения этим пользователем.

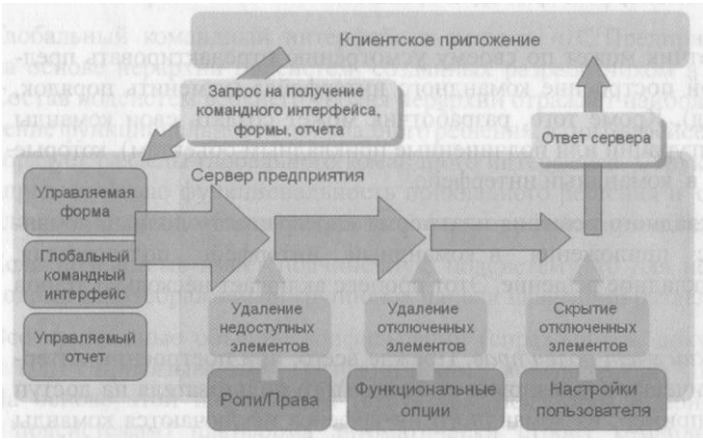


Рис. 1.16. Автоматическое формирование платформой командного интерфейса для пользователя

Подробнее об управляемом интерфейсе можно прочитать в книге «Разработка управляемого интерфейса» из серии «Профессиональная разработка».

Использование информации из метаданных

Интерфейсные механизмы «1С:Предприятия» широко используют информацию из метаданных для того, чтобы автоматизировать процесс разработки интерфейса и уменьшить необходимость детальной настройки, выполняемой разработчиком. Объекты конфигурации имеют множество интерфейсных свойств (таких как `Представление объекта`, `Представление списка`, `Маска`, `Формат` и пр.). Настроив их, разработчик может управлять отображением объекта в интерфейсе и его интерактивным поведением в процессе работы прикладного решения.

Расширения форм и элементов управления

«1С:Предприятие» использует механизм расширений форм и элементов управления. Расширения содержат набор свойств, методов и событий, которыми дополняется стандартное поведение элементов управления и формы после того, как они связываются с данными. В зависимости от того, какой тип данных отображает элемент управления или форма, платформа будет использовать то или иное расширение.

Такой подход позволяет реализовывать индивидуальные особенности поведения формы и элементов управления при работе с различными типами данных.

Командный интерфейс также наряду с командами, общими для данного элемента управления или формы, содержит и индивидуальный набор команд, определяемый конкретным типом данных, с которыми связан элемент управления или форма. Например, в форме документа содержится команда Провести, а в форме справочника — нет.

Генерация форм по умолчанию

Важной особенностью системы «1С:Предприятие» является механизм форм по умолчанию. Этот механизм освобождает разработчика от необходимости создания всех возможных форм для каждого объекта прикладного решения. Разработчику достаточно создать новый объект прикладного решения, а система сама сгенерирует в нужные моменты работы пользователя необходимые формы по умолчанию для работы с данными, содержащимися в этом объекте. Таким образом, разработчику нужно создавать собственные формы объектов прикладного решения лишь в том случае, если они должны иметь отличия (другой дизайн или специфическое поведение) от форм, генерируемых системой по умолчанию.

Клиентские приложения

Клиентское приложение - это программа, работающая на компьютере пользователя и обеспечивающая интерактивное взаимодействие системы «1С:Предприятие 8» с пользователем. В системе «1С:Предприятие 8» существует три клиентских приложения: толстый клиент, тонкий клиент и веб-клиент.

Толстый клиент позволяет реализовывать полные возможности «^Предприятия 8.2» как в плане разработки, администрирования, так и в плане исполнения прикладного кода. Однако он не поддерживает работу с информационными базами через Интернет, требует предварительной установки на компьютер пользователя и имеет довольно внушительный объем дистрибутива.

Тонкий клиент не позволяет разрабатывать и администрировать прикладные решения, однако может работать с информационными базами через Интернет. Он также требует предварительной установки на компьютер пользователя, но имеет значительно меньший размер дистрибутива, чем толстый клиент.

Веб-клиент не требует какой-либо предварительной установки на компьютер. В отличие от толстого и тонкого клиентов, он исполняется не в среде операционной системы компьютера, а в среде интернет-браузера (Windows Internet Explorer, Mozilla Firefox, Google Chrome или Safari). В веб-клиенте невозможна разработка прикладных решений.

В сводном виде возможности этих клиентских приложений можно представить следующим образом (табл. 1.1).

Таблица 1.1. Возможности клиентских приложений

	Толстый клиент	**Тонкий клиент**	**Веб-клиент**
Разработка прикладных решений	Да	Нет	Нет
Работа в локальной сети	Да	Да	Да
Работа через Интернет	Нет	Да	Да
Необходимость предварительной установки	Да, большой дистрибутив	Да, маленький дистрибутив	Нет

Независимо от режима работы (толстый, тонкий, веб-клиент) вся разработка прикладного решения ведется полностью в конфигураторе «^Предприятия», серверный и клиентский код пишется на встроенном языке. Однако работа в конфигураторе доступна только в режиме толстого клиента, тонкий и веб-клиент поддерживают работу только в пользовательском режиме 1С: Предприятие.

Подробнее о работе клиентских приложений можно прочитать в разделе «Общие сведения о клиент-серверном варианте работы» на стр. 217.

Подключение через Интернет

Подключение через Интернет позволяет обеспечить удаленную on-line работу пользователей с информационными базами. Это возможно благодаря использованию тонкого клиента и веб-клиента. Они подключаются к специальным образом настроенному веб-серверу, который осуществляет их взаимодействие с кластером серверов «1С:Предприятия» или с файловой информационной базой (рис. 1.17).



Рис. 1.17. Подключение к Интернету через веб-сервер в клиент-серверном варианте работы

Прикладные решения не требуют какой-либо доработки для того, чтобы работать с ними через Интернет. И тонкий, и веб-клиент самостоятельно обеспечивают функционирование интерфейса «1С:Предприятия 8» на компьютере пользователя.

Клиенты могут использовать различные способы выхода в Интернет. Это могут быть высокоскоростные подключения по выделенным линиям или через локальную сеть. А могут быть низкоскоростные подключения, например, через мобильное GPRS-соединение.

Специально для таких случаев, когда работа с информационной базой осуществляется через низкоскоростные каналы связи, тонкий клиент и веб- клиент имеют специальный режим запуска - режим низкой скорости соединения. В результате даже на низкоскоростных каналах связи платформа обеспечивает приемлемую скорость работы пользователей.

Файловый и клиент-серверный варианты работы

«1С:Предприятие» может работать в двух вариантах:

- файловый,
- клиент-серверный.

И в том, и в другом варианте все прикладные решения работают полностью идентично, что позволяет выбирать один или другой вариант работы без изменения существующего прикладного решения.

Файловый вариант работы рассчитан на персональную работу одного пользователя или работу небольшого количества пользователей в локальной сети. В этом варианте все данные информационной базы (конфигурация, база данных, административная информация) располагаются в одном файле - файловой базе данных. Работу с этой базой данных осуществляет файловая СУБД, разработанная фирмой «1С» и являющаяся частью платформы (рис. 1.18).



Рис. 1.18. Файловый вариант работы

Такой вариант работы обеспечивает легкость установки и эксплуатации прикладного решения. При этом для работы с информационной базой не требуются дополнительные программные средства, достаточно иметь операционную систему и «1С:Предприятие».

Клиент-серверный вариант предназначен для использования в рабочих группах или в масштабе предприятия. Он реализован на основе трехуровневой архитектуры «клиент-сервер» (рис. 1.19).

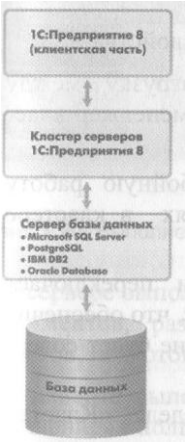


Рис. 1.19. Клиент-серверный вариант работы

Клиентское приложение, работающее у пользователя (толстый клиент, тонкий клиент или веб-клиент), взаимодействует с кластером серверов «1С:Пред- приятия 8», а кластер при необходимости обращается к серверу баз данных (Microsoft SQL Server, PostgreSQL, IBM DB2 или Oracle Database).

При этом физически кластер серверов «1С:Предприятия 8» и сервер баз данных могут располагаться как на одном компьютере, так и на разных. Это позволяет администратору при необходимости распределять нагрузку между серверами.

Подробнее о клиент-серверном варианте работы можно прочитать в главе «Клиент-серверный вариант работы» на стр. 217.

Кластер серверов «1С:Предприятия»

Кластер серверов «1С:Предприятия 8» - основной компонент платформы, обеспечивающий взаимодействие между пользователями и системой управления базами данных в клиент-серверном варианте работы. Наличие кластера

позволяет обеспечить бесперебойную, отказоустойчивую, конкурентную работу большого количества пользователей с крупными информационными базами.

Использование кластера серверов позволяет сосредоточить на нем выполнение наиболее объемных операций по обработке данных. Например, при выполнении даже весьма сложных запросов программа, работающая у пользователя, будет получать только необходимую ей выборку, а вся промежуточная обработка будет выполняться на сервере.

Кластер серверов «1С:Предприятия 8» отличают следующие свойства:

- *Масштабируемость* — позволяет перераспределить нагрузку между менеджерами кластера и тем самым разгрузить главный менеджер кластера;
- *Отказоустойчивость* - позволяет обеспечить бесперебойную работу пользователей при программных и аппаратных сбоях в кластере серверов;
- *Динамическое распределение нагрузки* - автоматически переключает пользователя на более производительный рабочий процесс, что обеспечит его более эффективную работу, причем такое переключение будет совершенно незаметно для пользователя.

Подробнее о кластере серверов можно прочитать в разделе «Кластер серверов» на. стр. 226.

Выполнение основной функциональности на сервере

Вся работа с прикладными объектами, чтение и запись базы данных выполняются только на сервере. Функциональность форм и командного интерфейса также реализована на сервере. Это позволяет увеличить производительность и надежность работы прикладного решения.

На сервере выполняется подготовка данных форм, расположение элементов, запись данных форм после изменения. На клиенте отображается уже подготовленная на сервере форма, выполняется ввод данных и вызовы сервера для записи введенных данных и других необходимых действий.

Командный интерфейс и отчеты также полностью формируются на сервере и отображаются на клиенте (рис. 1.20).

При этом механизмы платформы ориентированы на минимизацию объема данных, передаваемых на клиентский компьютер. Например, данные списков, табличных частей и отчетов передаются с сервера не сразу, а по мере просмотра их пользователем или использования во встроенном языке.



Рис. 1.20. Выполнение основной функциональности на сервере

На сервере выполняются: запросы к базе данных, запись данных, проведение документов, различные расчеты, выполнение обработок, формирование отчетов, подготовка форм к отображению.

На клиенте выполняется: получение и открытие форм, отображение форм, «общение» с пользователем (предупреждения, вопросы...), небольшие расчеты в формах, требующие быстрой реакции (например, умножение цены на количество), работа с локальными файлами, работа с торговым оборудованием.

Использование встроенного языка на клиенте

Управлять функциональностью форм можно не только на сервере, но и на клиенте. На клиенте также поддерживается работа встроенного языка, но в строго ограниченном объеме. Он используется в тех случаях, когда необходимо провести расчеты, связанные с отображенной на экране формой, например, быстро (бея обращения к серверу) подсчитать сумму строки документа на основе цены и количества, задать пользователю вопрос и обработать ответ, прочитать файл из файловой системы компьютера и отправить его на сервер.

Клиентские процедуры в модулях в явном виде отделяются от серверных, и в них используется ограниченный состав объектной модели встроенного языка.

На клиенте не допускается непосредственная работа с базой данных, а также работа непосредственно с прикладными объектами, например, недоступны такие типы встроенного языка, как СправочникОбъект.<имя>.

На клиенте невозможно использовать запросы. При необходимости вызова действий с данными в клиентском коде нужно вызывать серверные процедуры, которые, в свою очередь, будут обращаться к данным.

Поддержка различных хранилищ данных

В различных вариантах работы (файловый или клиент-серверный) «1С:Предприятие» использует различные хранилища данных.

В файловом варианте работы все данные информационной базы хранятся в одном файле - 1Cv8.1CD. Этот файл имеет специальный формат, поддерживаемый системой «1С: Предприятие».

В клиент-серверном варианте работы все данные хранятся в базах данных Microsoft SQL Server, PostgreSQL, IBM DB2 или Oracle Database.

Бесспорным преимуществом файловой базы данных является простота ее использования и обслуживания, однако она не рассчитана на интенсивную конкурентную работу большого числа пользователей и хранение значительных объемов данных.

В то же время использование сторонних СУБД, поддерживаемых «1С:Предприятием», позволяет обеспечить большую пропускную способность системы, хранение значительных объемов данных и надежность использования. Оборотной стороной этих преимуществ является более сложное, по сравнению с файловой базой данных, обслуживание и необходимость использования дополнительного программного обеспечения (в качестве сервера баз данных используются сторонние СУБД, поддерживаемые «1С:Предприятием»).

Работа под управлением различных операционных систем

Архитектура «1С:Предприятия» поддерживает многоплатформенность, то есть способность системы работать под управлением различных операционных систем.

Все основные компоненты платформы способны функционировать как под управлением операционной системы Windows, так и под управлением операционной системы Linux. Благодаря тому, что взаимодействие процессов между собой осуществляется по протоколу TCP/IP, в составе системы могут присутствовать компоненты с различными операционными системами. В общем случае архитектура «1С:Предприятия» позволяет смешивать

к одной системе элементы, работающие под управлением различных операционных систем, например, в одной системе могут работать тонкий клиент под управлением операционной системы Windows, веб-клиент под управлением Linux в интернет-браузере Mozilla Firefox и веб-клиент под управлением Windows в браузере Internet Explorer (рис. 1.21).

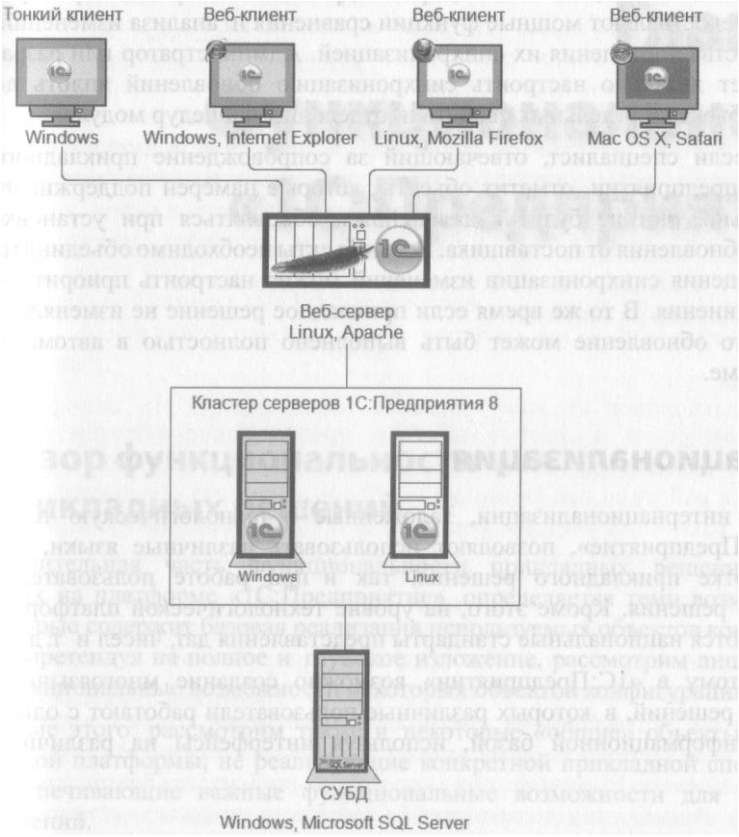


Рис. 1.21. Элементы системы «1С:Предприятие»,
работающие под управлением различных операционных систем

Обновление прикладных решений

Одним из важных архитектурных решений «1С:Предприятия» является наличие механизмов обновления прикладных решений. Эти механизмы обеспечивают синхронизацию изменений, сделанных поставщиком прикладного решения, с изменениями, внесенными при внедрении на конкретном предприятии. Они предоставляют мощные функции сравнения и анализа изменений, а также средства управления их синхронизацией. Администратор или разработчик может детально настроить синхронизацию обновлений вплоть до отдельных объектов, отдельных свойств и отдельных процедур модулей.

Например, если специалист, отвечающий за сопровождение прикладного решения на предприятии, отметит объекты, которые намерен поддерживать самостоятельно, они не будут в дальнейшем обновляться при установке очередного обновления от поставщика. Если объекты необходимо объединить, то для упрощения синхронизации изменений можно настроить приоритеты такого объединения. В то же время если прикладное решение не изменялось у клиента, то обновление может быть выполнено полностью в автоматическом режиме.

Интернационализация

Механизмы интернационализации, заложенные в технологическую платформу «1С:Предприятие», позволяют использовать различные языки, как при разработке прикладного решения, так и при работе пользователей прикладного решения. Кроме этого, на уровне технологической платформы поддерживаются национальные стандарты представления дат, чисел и т. д.

Благодаря этому в «1С:Предприятии» возможно создание многоязычных прикладных решений, в которых различные пользователи работают с одной и той же информационной базой, используя интерфейсы на различных языках.

Глава 2. Функциональность «1 С:Предприятия»

Обзор функциональности прикладных решений

Значительная часть функциональности прикладных решений, создаваемых на платформе «1С:Предприятие», определяется теми возможностями, которые содержит базовая реализация используемых объектов конфигурации. Не претендуя на полное и глубокое изложение, рассмотрим лишь основные функциональные возможности некоторых объектов конфигурации.

Кроме этого, рассмотрим также и некоторые «общие» объекты технологической платформы, не реализующие конкретной прикладной специфики, но обеспечивающие важные функциональные возможности для прикладных решений.

Справочники

Справочники описывают каталоги, содержимое которых более или менее постоянно. Это может быть, например, перечень выпускаемой продукции, список клиентов компании, перечень валют и т.д. Справочники обеспечивают поддержку иерархических структур, позволяют относить данные к отдельным объектам и их группам, предоставляют ряд других сервисных возможностей.

Многоуровневая иерархия, поддерживаемая справочниками, включается простой активизацией соответствующего свойства в метаданных. При этом поддержка иерархии распространяется сразу на все аспекты использования прикладного объекта.

Например, прототип (шаблон) справочника обеспечивает поддержку необходимых свойств и методов в объектной модели манипулирования данными (определение уровня объекта, контроль зацикливания иерархии и т.д.). В интерфейсных механизмах реализуется представление данных в виде иерархического списка или дерева с навигацией по уровням и интерактивным изменениям иерархии. В механизмах отчетов обеспечивается формирование иерархических документов такого рода и получение многоуровневой иерархии итогов в любых отчетах, в которых объекты этого типа выступают в качестве параметров для группировки.

Подробнее об использовании справочников можно прочитать в книге «Реализация прикладных задач в системе «1С:Предприятие 8.2» из серии «Профессиональная разработка».

Документы

Документы отражают в системе события, происходящие в жизни предприятия: поступление материалов, перечисление денег через банк, прием сотрудника на работу и т.д. Прототип (шаблон) документа обеспечивает их отражение в различных учетных механизмах, поддерживает контроль последовательности обработки событий, реализует сквозную нумерацию объектов разного типа и т. д.

Одним из важных функциональных механизмов системы является механизм проведения документов. Он предлагает разработчику стандартную модель организации связи между информацией о событиях, происходящих на предприятии, и различными учетными механизмами. Любая вводимая пользователем в виде документов информация может отражаться в любых учетных механизмах (планировании, управленческом учете, бухгалтерском учете и т.д.). Разработчик должен только указать в свойствах метаданных связь между документами и учетными механизмами, а также описать алгоритм проведения документа.

Все необходимые действия по проведению и отмене проведения система будет выполнять автоматически. При этом системой предоставляются дополнительные возможности, такие как поддержка отражения событий в реальном времени, поддержка восстановления последовательности отражения событий, происходящих на предприятии, при изменении их задним числом и т.д. В результате предоставляется единая модель связи исходных данных

|| учетных механизмов, которая не просто облегчает разработку, но и обеспечивает единообразное предсказуемое поведение всех прикладных решений, что существенно облегчает их освоение и поддержку.

11одробнее об использовании документов можно прочитать в книге «Реализация прикладных задач в системе «1С:Предприятие 8.2» из серии «Профессиональная разработка».

Механизм характеристик

Механизм описания характеристик позволяет организовать хранение свойств объектов (справочников, документов и т.д.), которые еще не известны на момент разработки прикладного решения. Таким образом, например, для номенклатуры пользователь сможет самостоятельно вводить новые свойства: цвет, размер, габариты, мощность и т.д. Для каждой группы номенклатуры может быть создан свой набор свойств: для холодильников - объем морозильной камеры, число компрессоров, уровень шума; для компьютеров - объем оперативной памяти, объем жесткого диска; для одежды - размер, рост, цвет.

Платформа «1С:Предприятие» позволяет описать дополнительные характеристики объектов конфигурации прямо в дереве метаданных один раз и хранить их в свойстве `Характеристики.` В дальнейшем на основе этих характеристик можно строить отчеты, анализировать объемы продаж, получать другую информацию для принятия решений.

Подробнее об использовании механизма характеристик можно прочитать в книге «Реализация прикладных задач в системе «1С:Предприятие 8.2» из серии «Профессиональная разработка».

Механизм сведений

Механизм хранения сведений, реализованный в платформе «1С:Предприятие», позволяет хранить в прикладном решении произвольные данные в разрезе нескольких измерений. Например, можно хранить курсы валют в разрезе валют или цены предприятия в разрезе номенклатуры и типа цен.

Кроме этого, может быть задана различная периодичность хранения сведений, что позволяет хранить не только сами значения, но и историю их изменения.

Подробнее об использовании механизма сведений можно прочитать в книге «Реализация прикладных задач в системе «1С:Предприятие 8.2» из серии «Профессиональная разработка».

Механизм учета движения средств

Механизм учета движения средств (финансов, товаров, материалов и т.д.) позволяет автоматизировать такие направления, как складской учет, взаиморасчеты, планирование. Основу этого механизма составляют регистры накопления.

Регистр накопления образует многомерную систему измерений и позволяет «накапливать» числовые данные в разрезе нескольких измерений. Например, в таком регистре можно накапливать информацию об остатках товаров в разрезе номенклатуры и склада или информацию об объемах продаж в разрезе номенклатуры и подразделения компании.

Информация из регистров используется в основном в различных экономических и аналитических отчетах. Для более быстрого и эффективного формирования отчетов в платформе реализован механизм агрегатов для оборотных регистров накопления. Это особенно важно для больших информационных баз, содержащих сотни тысяч и миллионы записей в регистрах. Используя механизм агрегатов, система автоматически оценивает интенсивность работы пользователей с теми или иными разрезами информации и на основе накопленной статистики выбирает оптимальный состав поддерживаемых агрегатов.

Подробнее об использовании механизма учета движения средств можно прочитать в книге «Реализация прикладных задач в системе «1С:Предприятие 8.2» из серии «Профессиональная разработка».

Механизм бухгалтерского учета

Механизмы бухгалтерского учета позволяют реализовать систему двойной записи бухгалтерского учета. Они не навязывают разработчику собственно принципов ведения бухгалтерского учета и позволяют создавать модели учета, применимые как в России, так и в других странах.

Можно перечислить следующие основные возможности, реализуемые механизмами бухгалтерского учета:

- ведение многоуровневых планов счетов с произвольной иерархией, в которых поддерживается фиксированная или переменная разрядность кодов счетов;
- ведение аналитического учета в нескольких разрезах и уровнях;
- ведение учета одновременно по нескольким планам счетов;
- ведение консолидированного учета по нескольким юридическим лицам;
- возможность указания для отдельных разрезов аналитики произвольного числа видов учета, таких как количественный, суммовой, валютный учет и т.д.

11одробнее об использовании механизма бухгалтерского учета можно прочитать в книге «Реализация прикладных задач в системе 1С:Предприятие 8.2» из серии «Профессиональная разработка».

Механизм сложных периодических расчетов

Механизм сложных периодических расчетов позволяет реализовывать различные модели расчета заработной платы. Работа механизма основана на двух составляющих.

С одной стороны, механизм сложных периодических расчетов содержит средства для описания различных видов расчета, которые будут использоваться в прикладном решении. Например, это могут быть такие виды расчета, как оклад, алименты, штраф и т. д. Помимо собственно описания этих видов расчета, существует возможность задать правила, по которым одни виды расчета будут влиять на другие виды расчета.

С другой стороны, этот механизм предоставляет возможность хранения промежуточных данных, которые используются для выполнения расчетов, и конечных результатов расчетов.

Расчет зарплаты является наиболее типичным применением данного механизма, но сам механизм не имеет ориентации именно на эту задачу и успешно используется для решения других задач, требующих описания периодических расчетов со сложными взаимосвязями, например, расчета дивидендов, стоимости коммунальных услуг и т. д.

Подробнее об использовании механизма сложных периодических расчетов можно прочитать в книге «Реализация прикладных задач в системе ^Предприятие 8.2» из серии «Профессиональная разработка».

Механизм бизнес-процессов

Механизм бизнес-процессов позволяет описывать, создавать и управлять выполнением бизнес-процессов в прикладных решениях. Целью этого механизма является автоматизация цепочек связанных операций, направленных на достижение общей цели, обычно в контексте организационной структуры, определяющей функциональные роли и связи.

Этот механизм включает средства для описания в прикладном решении схем бизнес-процессов и их ролевой маршрутизации для формирования заданий, выполняющихся в каждой точке маршрута, для управления бизнес-процессом и организации его связи с другими функциями прикладного решения.
Важно отметить, что данный механизм предлагает готовую стратегию автоматизации совместной деятельности работников предприятия. Для описания простейших бизнес-процессов достаточно визуально задать схемы маршрута

и указать условия ветвления в их узловых точках. Все остальные действия выполняются системой автоматически. При реализации сложных бизнес- процессов усилия разработчика требуются в основном для тесной их увязки с функциями прикладного решения.

Механизм анализа данных и прогнозирования

Механизм анализа данных и прогнозирования позволяет реализовывать в прикладных решениях инструменты для выявления закономерностей, которые обычно скрываются за большими объемами информации.

Например, проанализировав данные о продажах товаров, можно выявить группы товаров, которые обычно покупаются вместе, и при очередной покупке рекомендовать клиенту дополнительные товары, исходя из найденных закономерностей и тех товаров, которые клиент уже выбрал.

Этот механизм поддерживает выполнение нескольких типов анализа данных, таких как общая статистика, поиск ассоциаций, дерево решений, поиск последовательностей, кластерный анализ.

Сложные отчеты, использующие систему компоновки данных

Для формирования сложных экономических и аналитических отчетов платформа «1С:Предприятие» содержит мощный и гибкий механизм - *система компоновки данных*. Отчеты строятся на основе схемы компоновки данных - специального макета отчета, содержащего его декларативное описание.

Схема компоновки данных создается разработчиком и содержит описание наборов данных (источников информации для отчета) и связей между ними, параметров получения данных, полей отчета и т.п., а также в ней разработчик задает стандартные настройки компоновки данных - структуру отчета, порядок, отбор и др.

У каждого отчета разработчик может предусмотреть несколько вариантов настроек, представляющих данные в виде диаграммы, таблицы или группировок, имеющих другой состав полей, итогов и т.п. Разработчик может описать, какие настройки отчета будут доступны пользователю (пользовательские настройки) и какие из этих настроек будут присутствовать прямо в форме отчета (быстрые пользовательские настройки). С их помощью пользователь может настроить вариант отчета под себя и сохранить его для дальнейшего использования.

Система компоновки данных предоставляет широкие возможности для настройки оформления отчета и его отдельных элементов - выбор макета оформления (в том числе и собственного), настройка вывода группировок, итогов, заголовка отчета и многое другое.

Таким образом, большинство отчетов может быть описано и настроено разработчиком в схеме компоновки данных и не требует написания программного кода.

На основе описания в схеме компоновки данных и стандартных настроек система компоновки данных компонует отчет и выводит его пользователю в виде табличного документа (программным образом данные также можно вывести в таблицу значений, дерево значений и др.).

Пользователь может выбрать вариант отчета, являющийся с его точки зрения наиболее информативным, а может, при желании и соответствующей квалификации, задать собственные настройки и получить другой отчет, основанный на той же схеме компоновки данных.

Таким образом, система компоновки данных предоставляет пользователям возможность получать разнообразные необходимые им отчеты быстро, не обращаясь к разработчикам.

Система компоновки данных позволяет использовать в отчетах произвольные характеристики объектов информационной базы, предоставляет возможность расшифровывать информацию, содержащуюся в отчете, выводить в отчет иерархические данные и многое другое.

Подробнее об использовании системы компоновки данных можно прочитать в книге «Разработка сложных отчетов в «1С:Предприятии 8». Система компоновки данных».

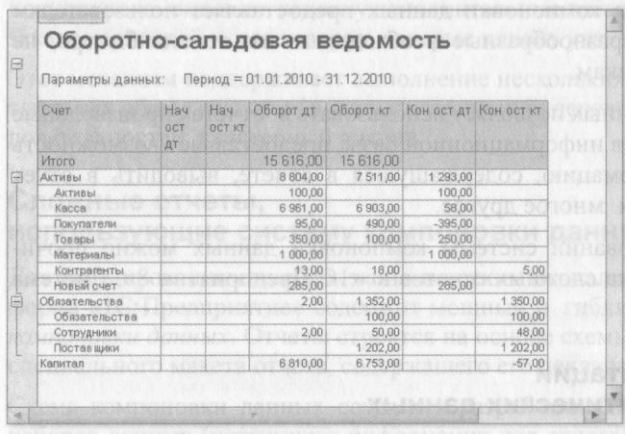
Механизмы презентации текстовых и аналитических данных

Технологическая платформа «1С Предприятие» содержит целый ряд объектов, которые позволяют представлять итоговую информацию в удобном для пользователя виде, обеспечивая при этом в большей или меньшей степени интерактивное взаимодействие.

Одним из мощных средств презентации любой информации и вывода ее на печать является *табличный документ* (рис. 2.1). Он обеспечивает не только эффективную подготовку печатных документов, но и просмотр их на экране в удобном для пользователя виде. Перечислим основные возможности табличного документа:

- оформление отчета, включая тип и размер шрифта, цвет текста и фона, тип и цвет рамки, рисунки и т.д.;
- использование группировок (как вертикальных, так и горизонтальных), с помощью которых можно отражать промежуточные итоги;

- поддержка механизма расшифровок, когда при щелчке на строке или ячейке отчета формируется более детальный отчет или открывается объект базы данных;
- использование примечаний, содержащих дополнительную информацию о данных, расположенных в ячейке или области документа;
- поддержка сводных таблиц, которые позволяют отобразить многомерные данные в виде плоской таблицы с вложенными заголовками;
- поддержка диаграмм различного вида для наглядного представления экономической информации в графическом виде;
- возможность сохранения табличного документа в различных форматах;
- и др.



Оборотно-сальдовая ведомость

Параметры данных: Период = 01.01.2010 - 31.12.2010

Счет	Нач ост дт	Нач ост кт	Оборот дт	Оборот кт	Кон ост дт	Кон ост кт
Итого			15 616,00	15 616,00		
Активы			8 804,00	7 511,00	1 293,00	
Активы			100,00		100,00	
Касса			6 961,00	6 903,00	58,00	
Покупатели			95,00	490,00	-395,00	
Товары			350,00	100,00	250,00	
Материалы			1 000,00		1 000,00	
Контрагенты			13,00	18,00		5,00
Новый счет			285,00		285,00	
Обязательства			2,00	1 352,00		1 350,00
Обязательства				100,00		100,00
Сотрудники			2,00	50,00		48,00
Поставщики				1 202,00		1 202,00
Капитал			6 810,00	6 753,00		-57,00

Рис. 2.1. Табличный документ

Используя *форматированный документ,* можно создавать фрагменты текста, содержащие различное шрифтовое, цветовое оформление, выравнивание, включающие картинки и гиперссылки. В основном он используется для редактирования в формах в тех случаях, когда текст должен содержать различное шрифтовое, цветовое оформление, гиперссылки и картинки. Например, в формах электронных писем (рис. 2.2).

Для представления итоговых данных в разрезе их географического положения в «1С:Предприятии» используется специальный объект *-географическая схема* (рис. 2.3). Она позволяет создавать отчеты, иллюстрирующие, например, объемы продаж тех или иных товаров в различных регионах страны. Также географическая схема может быть использована просто для отображения тех или иных географических данных, например, схемы проезда к офису или маршрута движения транспортного средства.

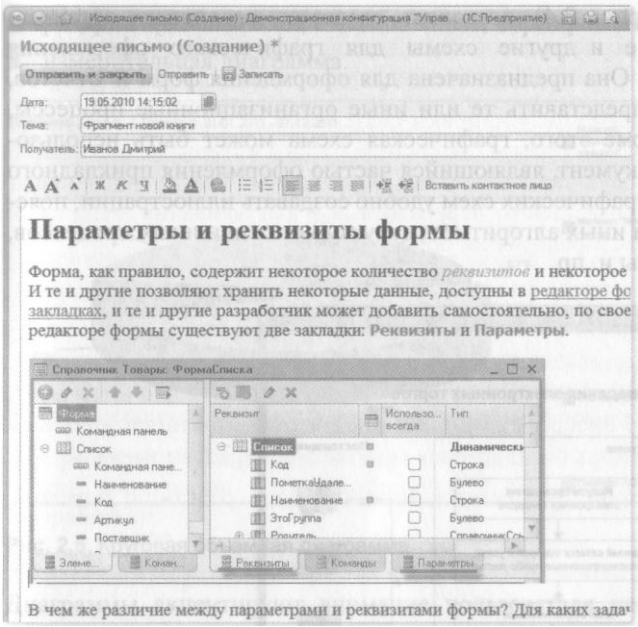


Рис. 2.2. Форматированный документ

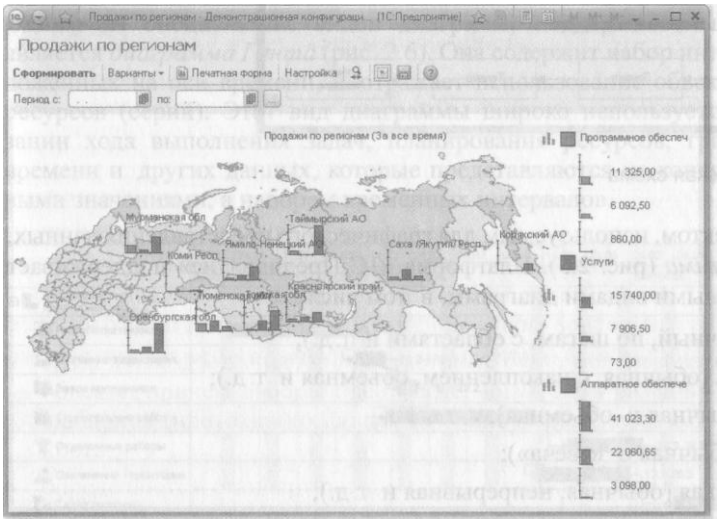


Рис. 2.3. Географическая схема

Графическая схема (рис. 2.4) позволяет создавать различные организационные, структурные и другие схемы для графического оформления прикладного решения. Она предназначена для оформления форм и отчетов, в которых требуется представить те или иные организационные процессы, блок-схемы и пр. Кроме этого, графическая схема может быть использована как отдельный документ, являющийся частью оформления прикладного решения. С помощью графических схем удобно создавать иллюстрации, поясняющие состав тех или иных алгоритмов, структуру тех или иных процессов, организационные схемы и пр.



Рис. 2.4. Графическая схема

Еще одним объектом, используемым для графического представления данных, является *диаграмма* (рис. 2.5). Платформа «1С:Предприятие» поддерживает работу с различными видами диаграмм, в том числе:

- график (обычный, по шагам, с областями и т. д.);
- гистограмма (обычная, с накоплением, объемная и т. д.);
- круговая (обычная и объемная);
- биржевая (обычная и «свеча»);
- изометрическая (обычная, непрерывная и т. д.);
- поверхностная (каркасная, выпуклая, вогнутая поверхность, сотовая и др.);

- радарная (с областями, с накоплением, нормированная и др.);
- измерительная диаграмма.

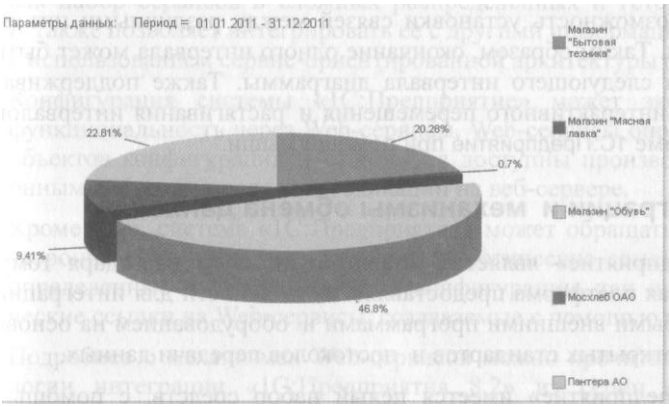


Рис. 2.5. Круговая объемная диаграмма

Диаграммы акцентируют внимание пользователя на динамике изменения данных и помогают быстро производить относительное сравнение данных. Кроме этого, специализированные виды диаграмм могут отражать закономерности, обычно скрытые за большими объемами данных.

Одним из специальных видов диаграмм, поддерживаемых платформой, является *диаграмма Ганта* (рис. 2.6). Она содержит набор интервалов, расположенных на оси времени, и отражает использование объектами (точками) ресурсов (серий). Этот вид диаграммы широко используется для визуализации хода выполнения задач, планирования ресурсов, графика рабочего времени и других данных, которые представляются не конкретными числовыми значениями, а набором временных интервалов.

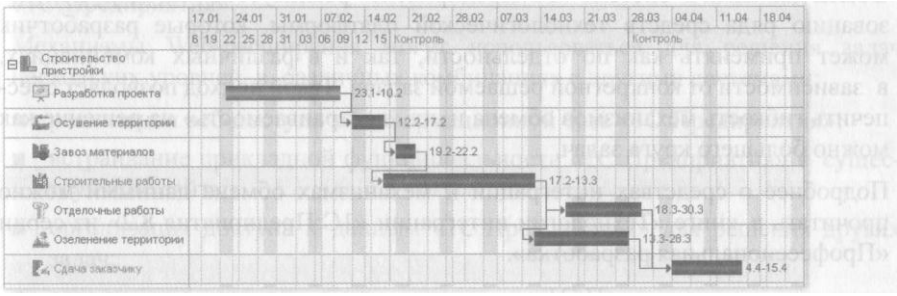


Рис. 2.6. Диаграмма Ганта

Диаграмма Ганта имеет гибкую структуру данных. Как точки, так и серии представляют собой иерархические коллекции, что позволяет, например, представить проект как набор связанных, иерархических задач.

Поддерживается возможность установки связей между различными интервалами диаграммы. Таким образом, окончание одного интервала может быть связано с началом следующего интервала диаграммы. Также поддерживается возможность интерактивного перемещения и растягивания интервалов диаграммы в режиме 1С:Предприятие при помощи мыши.

Средства интеграции и механизмы обмена данными

Система «1С:Предприятие» является открытой системой благодаря тому, что технологическая платформа предоставляет возможности для интеграции практически с любыми внешними программами и оборудованием на основе общепризнанных открытых стандартов и протоколов передачи данных.

В системе «1С:Предприятие» имеется целый набор средств, с помощью которых можно:

- создавать, обрабатывать и обмениваться данными различных форматов;
- осуществлять доступ ко всем объектам системы «1С:Предприятие», реализующим ее функциональные возможности;

я поддерживать различные протоколы обмена;

- поддерживать стандарты взаимодействия с другими подсистемами;
- разрабатывать собственные интернет-решения.

Механизмы обмена данными позволяют создавать территориально распределенные информационные системы на основе:

- «1С:Предприятия 8»;
- других информационных систем, не основанных на «1С:Предприятии 8». Обмен данными в системе «1С:Предприятие» реализуется благодаря использованию ряда средств технологической платформы, которые разработчик может применять как по отдельности, так и в различных комбинациях, в зависимости от конкретной решаемой задачи. Такой подход позволяет обеспечить гибкость механизмов обмена и их «настраиваемость» на решение как можно большего круга задач.

Подробнее о средствах интеграции и механизмах обмена данными можно прочитать в книге «Технологии интеграции «1С:Предприятия 8.2» из серии «Профессиональная разработка».

Web-сервисы

Механизм Web-сервисов позволяет использовать систему «1С:Предприятие» как набор сервисов в сложных распределенных и гетерогенных системах, а также позволяет интегрировать ее с другими информационными системами с использованием сервис-ориентированной архитектуры (SOA).

Конфигурация системы «1С:Предприятие» может экспортировать свою функциональность через Web-сервисы. Web-сервисы определяются в дереве объектов конфигурации и становятся доступны произвольным информационным системам после их публикации на веб-сервере.

Кроме того, система «1С:Предприятие» может обращаться к Web-сервисам сторонних производителей как через статические ссылки на Web-сервисы, определенные в дереве объектов конфигурации, так и используя динамические ссылки на Web-сервисы, создаваемые с помощью встроенного языка.

Подробнее о механизмах Web-сервисов можно прочитать в книге «Технологии интеграции «1С:Предприятия 8.2» из серии «Профессиональная разработка».

Web-расширение

Одним из средств интеграции, позволяющим расширить сферу применения «1С:Предприятия», является Web-расширение. Данный программный продукт позволяет организовать доступ через веб-интерфейс к функциональности прикладных решений новых категорий пользователей, в том числе и тех, у которых на компьютерах не установлена платформа «1С:Предпри- ятие». Это могут быть мобильные пользователи, сотрудники территориально удаленных подразделений, посетители интернет-магазинов и веб-порталов.

Web-расширение позволяет встраивать доступ к данным «1С:Предприятия» в существующие веб-сайты и веб-приложения и создавать готовые веб-приложения и веб-сервисы, использующие информационную базу «1С: Предприятия».

Механизмы Web-расширения могут использоваться для решения задач нескольких уровней, в различных комбинациях с другими системами:

- реализация веб-доступа к информационной базе «1С:Предприятия»;
- встраивание прикладной функциональности «1С:Предприятия» в существующие сайты;
- организация доступа к данным «1С:Предприятия» для решения других задач;
- организация программного доступа к «1С:Предприятию» из других систем.

Подробнее о Web-расширении можно прочитать в книге «Технологии интеграции «1С:Предприятия 8.2» из серии «Профессиональная разработка».

Механизм полнотекстового поиска

Механизм полнотекстового поиска предоставляет возможность пользователю быстро находить нужную информацию в информационной базе и в справочной системе. Например, позволяет найти все вхождения заданного слова или сочетания слов в данных информационной базы, при этом найденные слова выделяются цветом (рис. 2.7).

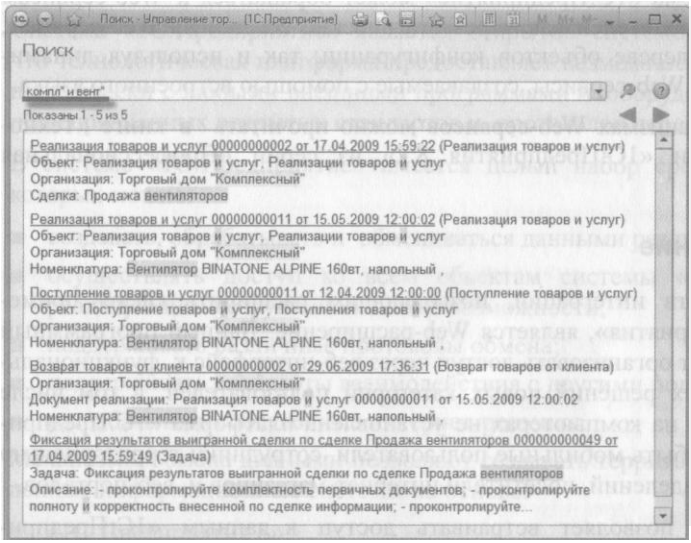


Рис. 2.7. Поиск данных в информационной базе

Механизм полнотекстового поиска основан на использовании двух составляющих:

- полнотекстового индекса,
- средств выполнения полнотекстового поиска.

Поиск осуществляется по тем данным, которые содержатся в полнотекстовом индексе. Полнотекстовый индекс создается один раз и затем должен периодически обновляться (в зависимости от частоты модификации информационной базы). Создание и обновление полнотекстового индекса может выполняться как интерактивно, в режиме 1С: Предприятие, так и программно, средствами встроенного языка.

В процессе работы информационной базы система отслеживает факт изменения данных в тех объектах конфигурации, которые могут участвовать к полнотекстовом поиске. Такими объектами являются, например, планы обмена, справочники, документы, планы видов характеристик, планы счетов, планы видов расчета, регистры (сведений, накопления, бухгалтерии, расчета), иинзес-процессы и задачи.

Впоследствии при создании или обновлении полнотекстового индекса система анализирует данные, содержащиеся в реквизитах этих объектов, и включает)ги данные в полнотекстовый индекс. При этом система «1С:Предприятие» позволяет осуществлять выборочное включение данных прикладных объектов и их реквизитов в полнотекстовый поиск. Также существует возможность ограничить область поиска данными только указанных объектов конфигурации.

Собственно полнотекстовый поиск выполняется средствами встроенного языка, например, с помощью обработки.

Полнотекстовый поиск выполняется в соответствии с правами пользователя. Таким образом, если какая-то информация недоступна данному пользователю, он не сможет получить ее и при помощи полнотекстового поиска.

Результаты полнотекстового поиска возвращаются порциями, и, кроме того, они отсортированы в определенном порядке. Это сделано для того, чтобы с большой долей вероятности пользователь получал требуемые ему данные в начале первой порции результатов поиска.

Механизм полнотекстового поиска предоставляет возможность нечеткого поиска с указанием порога нечеткости, выполнение поиска с учетом синонимов русского, английского и украинского языков, возможность использования дополнительных словарей полнотекстового поиска и многое другое.

Механизм криптографии

Механизм криптографии позволяет прикладным решениям использовать криптографические операции для обработки данных, хранящихся в информационной базе. Криптографические операции включают методы зашифровывания/расшифровывания информации, системы электронной цифровой подписи и др. Таким образом, с помощью механизма криптографии система «1С:Предприятие» обеспечивает высокую конфиденциальность и аутентичность информации, хранящейся в информационной базе.

Механизм криптографии не содержит собственных алгоритмов криптографии, но при этом обеспечивает набор объектов, позволяющих взаимодействовать с внешними модулями криптографии сторонних производителей - крип- топровайдерами.

Для взаимодействия с криптопровайдерами в операционной системе Windows используется интерфейс CryptoAPI. Таким образом, прикладные решения могут взаимодействовать с любыми криптопровайдерами, поддерживающими этот криптографический интерфейс.

Для взаимодействия с модулями криптографии в операционной системе Linux используется непосредственное взаимодействие с установленными компонентами.

Механизм разделения данных

Этот механизм позволяет, например, хранить в одной информационной базе данные нескольких независимых друг от друга организаций. При этом для работы с этими данными используется одна и та же конфигурация, находящаяся в этой же информационной базе.

Механизм позволяет разделять данные как полностью, так и частично. Например, какая-то информация может быть общей для всех организаций (справочник банков), а какая-то - индивидуальной, видимой и доступной только данной конкретной организации. Например, информация о хозяйственной или финансовой деятельности.

Такой подход значительно облегчает администрирование и обновление прикладных решений в тех случаях, когда одно и то же прикладное решение используется для ведения учета разных организаций. Например, когда фирма оказывает услуги по ведению бухгалтерского учета для многих небольших организаций, не имеющих собственной бухгалтерии.

Внешние источники данных

Платформа «1С:Предприятие» позволяет работать с внешними источниками данных. Это может быть необходимо для использования средств платформы для работы с базами данных, уже существующими на предприятии или для разработки прикладных решений, которым нужно получать данные из внешних баз данных.

Механизм работы с внешними источниками данных позволяет получать данные из различных ODBC-источников. При этом в случае получения данных из СУБД, поддерживаемых «1С:Предприятием» (SQL Server, PostgreSQL, IBM DB2 или Oracle Database), обеспечиваются полные возможности платформы в построении запросов, отчетов и т.д. В случае использования ODBC-источников для других СУБД эти возможности могут быть ограничены стандартами самой СУБД.

Внешние источники данных описываются в конфигураторе в отдельной ветви объектов конфигурации. Каждый объект, описывающий внешний источник данных, включает таблицы, поля и связи между таблицами.

Внешние источники данных поддерживаются в большинстве механизмов платформы: запросы, отчеты, основанные на системе компоновке данных, динамические списки, управляемые формы, встроенный язык и др. Данные внешних источников могут использоваться только на чтение и не могут быть изменены из встроенного языка. Реквизиты объектов конфигурации (например, реквизит справочника) могут иметь ссылочный тип и ссылаться на объектную таблицу внешнего источника данных.

При подключении к внешнему источнику данных используются параметры соединения, которые задаются в параметрах сеанса, параметрах пользователя или в общих параметрах. Для управления процессом подключения/отключения и настройкой параметров соединения используется стандартная обработка Управление внешними источниками данных. Кроме того, подключение к внешнему источнику данных выполняется автоматически при обращении к данным внешнего источника, например, при выполнении запроса.

Обзор функциональности средств разработки

Состав средств разработки достаточно широк и разнообразен. Это позволяет выполнять полный цикл действий, от создания конфигурации и до получения тиражируемого дистрибутива прикладного решения, не прибегая к помощи каких-либо продуктов сторонних производителей (рис. 2.8).

Перечислим основные средства, используемые при разработке прикладных решений «1С:Предприятия».

Объекты конфигурации и инструменты для их редактирования

Прежде всего, это, конечно же, сами объекты конфигурации и инструменты для их редактирования - дерево объектов конфигурации, окно редактирования объекта конфигурации, палитра свойств и окно Дополнительно. С помощью этих инструментов выполняется добавление, удаление объектов конфигурации, изменение их свойств, установка связей с другими объектами конфигурации и пр.

Дерево объектов конфигурации является одним из основных инструментов для работы с конфигурацией. Оно представляет состав конфигурации в виде древовидной структуры, каждая ветвь которой описывает определенную составляющую конфигурации и позволяет выполнять различные действия над отдельными объектами конфигурации.

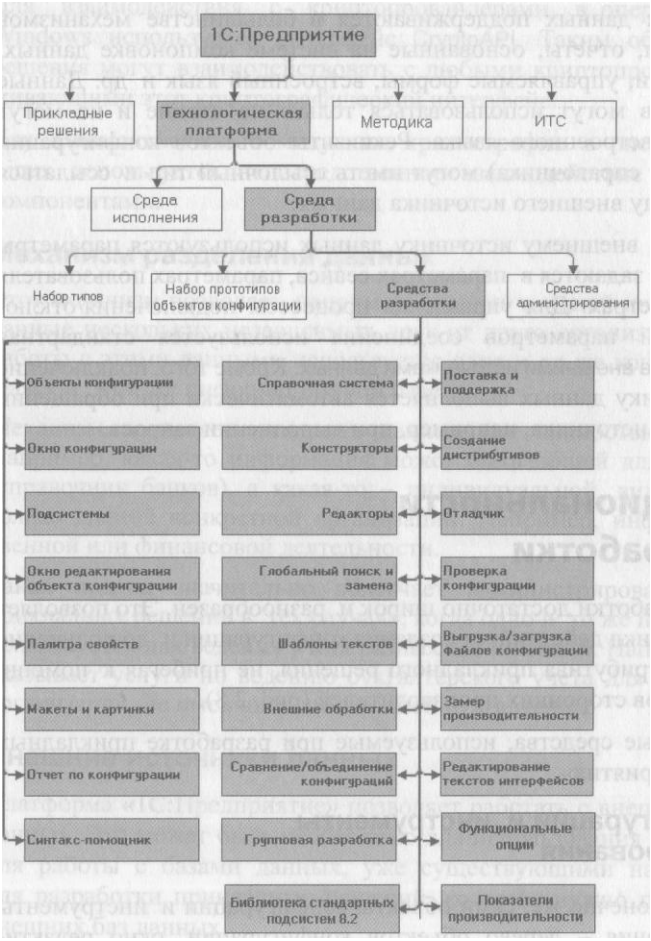


Рис. 2.8. Состав средств разработки

Окно редактирования объекта конфигурации позволяет выполнять в числе прочего последовательную установку свойств объекта, что удобно (особенно для начинающих разработчиков), т. к. позволяет задать в первую очередь те свойства объекта, которые могут определять наличие или отсутствие других свойств объекта.

Палитра свойств также предоставляет возможность изменять свойства объекта и получать быстрый доступ к информации, связанной с данным объектом. Палитра свойств удобна в использовании для опытных разработчиков,

а также тогда, когда необходимо просмотреть или изменить одноименные свойства у различных объектов конфигурации.

Окно Дополнительно служит для отображения в едином списке свойств объекта, относящихся к различным объектам конфигурации. В этом окне можно настраивать вхождение команд объекта в командный интерфейс, просматривать и изменять включение объекта конфигурации в подсистемы, редактировать связи с функциональными опциями, настраивать права и выполнять другие действия.

Конструкторы и редакторы

В состав средств разработки входит довольно большое количество конструкторов и редакторов. Конструкторы позволяют автоматизировать и облегчить создание некоторых часто используемых элементов прикладного решения, модификация которых может быть формализована. Как правило, эти элементы прикладного решения могут быть созданы и без использования конструктора, но конструктор позволяет сделать это быстрее и легче.

Например, текст запроса может быть полностью написан самим разработчиком. Для этого разработчик должен хорошо знать синтаксис языка запросов и понимать назначение различных предложений языка запросов. В то же время текст запроса можно создать с помощью конструктора запросов. При этом используется визуальное конструирование запроса, когда с помощью мыши необходимо выбрать и перетащить нужные таблицы, поля, установить связи между ними и т. д. После нажатия кнопки ОК конструктор запроса создаст синтаксически верный текст запроса. Кроме того, можно использовать конструктор запроса с обработкой результата, который позволяет создать готовый фрагмент кода для получения данных с помощью запроса и вывода их, например, в табличный документ.

Можно перечислить, например, следующие конструкторы, присутствующие в интерфейсе разработки и администрирования:

- конструктор запроса,
- конструктор запроса с обработкой результата,
- конструктор движений регистров,
- конструктор печати,
- конструктор ввода на основании,
- конструктор форм объектов конфигурации, ш
  конструктор макета,
- конструктор форматной строки,

- конструктор строк на разных языках,
- конструктор агрегатов,
- конструктор схемы компоновки данных,
- конструктор настроек компоновки данных,
- конструктор макета оформления компоновки данных,
- конструктор ограничения доступа к данным.

Редакторы позволяют создавать и изменять элементы прикладного решения, модификацию которых затруднительно формализовать или для которых подобная формализация не имеет практического смысла.

Например, конструктор форм объектов конфигурации позволяет создать некоторую типовую форму объекта, разместить на ней необходимые поля, назначить источники данных и т.д. Этот процесс легко формализуется и потому может быть выполнен конструктором. Однако дальнейшее редактирование формы, как правило, является процессом творческим и формализации не поддается. Поэтому платформа предоставляет в распоряжение разработчика редактор форм, с помощью которого он придает форме нужный вид, располагает дополнительные элементы управления, настраивает при необходимости привязки и т.д. в соответствии с конкретным назначением формы.

Можно перечислить, например, следующие редакторы, реализованные в интерфейсе разработки и администрирования:

- редактор форм,
- редактор текстов и модулей,
- редактор командного интерфейса конфигурации,
- редактор рабочей области рабочего стола,
- редактор командного интерфейса рабочего стола,
- редактор командного интерфейса,
- редактор «Все подсистемы»,
- редактор роли,
- редактор «Все роли»,
- редактор «Все ограничения доступа»,
- редактор табличных документов,
- редактор графической схемы,
- редактор картинок,
- HTML-редактор.

Синтакс-помощник

Синтакс-помощник позволяет получать справку по использованию конструкций встроенного языка, свойствам, методам и событиям объектов встроенного языка (как поставляемых непосредственно платформой, так и создаваемых наследованием от прототипов объектов конфигурации), а также справку по составу таблиц, являющихся источниками данных в языке запросов. Синтакс-помощник может быть использован как самостоятельно, гак и в связке с редактором текстов и модулей. Находясь в тексте модуля, можно, используя сочетание клавиш, перейти к описанию соответствующего свойства, метода, функции и т. д. в синтакс-помощнике, а конструкции встроенного языка, описанные в синтакс-помощнике, в свою очередь, могут быть перенесены в текст модуля с помощью мыши.

Отладчик и режим замера производительности

Среди средств разработки, используемых наиболее часто и интенсивно, можно отметить также отладчик и режим замера производительности. Отладчик позволяет проанализировать работу прикладного решения, выполнить остановку в указанном месте кода на встроенном языке, просмотреть значения переменных, состояние используемых объектов в определенные моменты работы, выполнить фрагмент кода «по шагам» и т. д. Режим замера производительности используется для того, чтобы проанализировать эффективность выполнения участка кода, узнать время, затраченное на выполнение каждой строки этого участка кода (как абсолютное, так и относительное, в процентах к общему времени выполнения всего участка). Таким образом можно определить операторы, выполнение которых занимает основное количество времени, и попытаться оптимизировать используемый алгоритм. Подробнее об использовании режима замера производительности можно прочитать в книге «Разработка управляемого интерфейса» (ч. 4) из серии «Профессиональная разработка».

Проверка конфигурации

Режим проверки конфигурации позволяет выявить ошибки, которые не являются критичными для функционирования прикладного решения в принципе, но наличие которых может существенно снизить скорость работы прикладного решения или даже привести к возникновению ошибок при работе в некоторых специальных режимах. Использование этого режима не является обязательным, но может оказаться очень полезным, например, для проверки конфигурации перед поставкой заказчику, перед выпуском тиражного решения, для проверки после массированного удаления объектов или после объединения конфигураций.

Сравнение/объединение конфигураций

Механизм сравнения/объединения конфигураций позволяет выполнять, например, детальный анализ отличий между разными версиями одного и того же прикладного решения. Также с помощью этого механизма можно выполнять переход от одной версии прикладного решения к другой версии того же самого или другого прикладного решения. При этом разработчик имеет возможность задать различные правила объединения для различных элементов прикладного решения, указывая, какой элемент должен быть взят из новой версии, какой оставлен без изменений, а какие элементы, например, должны быть объединены и содержать как старый, так и новый варианты. После нажатия кнопки Выполнить платформа автоматически выполнит объединение в соответствии с правилами, заданными разработчиком. Использование механизма сравнения/объединения конфигураций позволяет избежать ручной модификации конфигурации (и связанного с этим большого количества ошибок) при переносе изменений из одной конфигурации в другую, а также значительно ускорить этот процесс.

Редактирование текстов интерфейса

Платформа «1С:Предприятие» позволяет создавать многоязычные прикладные решения, в которых каждый пользователь может работать с интерфейсом на своем родном языке. Для того чтобы помочь разработчику при создании многоязычных прикладных решений, в состав средств разработки входит механизм редактирования текстов интерфейса. Этот механизм позволяет найти и сгруппировать все вхождения той или иной строки в конфигурации и в одно нажатие заменить их или добавить аналог этой строки на другом языке. Кроме этого, механизм позволяет переносить строки на разных языках из одного прикладного решения в другое. Подробнее о механизме редактирования текстов интерфейса можно прочитать в книге «Инструменты для создания тиражируемых приложений «1С:Предприятия 8.2» из серии «Профессиональная разработка».

Групповая разработка прикладных решений

Механизм групповой разработки позволяет вести и версионировать разработку прикладных решений группе разработчиков, внося изменения в конфигурацию одновременно, по мере выполнения каждым из них своего участка работы. Подробнее о механизме групповой разработки можно прочитать в книге «Инструменты для создания тиражируемых приложений «1С: Предприятия 8.2» из серии «Профессиональная разработка».

Поставка и поддержка прикладных решений

Механизм поставки и поддержки прикладных решений служит для автоматизации процесса поддержки разработчиками прикладного решения, используемого пользователем. Задача поставки и поддержки заключается в создании новых версий прикладного решения и обновления той версии прикладного решения, которая находится у пользователей. Подробнее о механизме поставки и поддержки можно прочитать в книге «Инструменты для создания тиражируемых приложений «1С:Предприятия 8.2» из серии «Профессиональная разработка».

Создание дистрибутивов

Механизм создания дистрибутивов позволяет разработчику создать комплект поставки - набор файлов, предназначенных для установки на компьютере пользователя. Комплект поставки включает в себя программу установки Setup.exe и набор файлов поставки, сжатых в архив. Для установки прикладного решения пользователю достаточно запустить на своем компьютере программу установки, входящую в комплект поставки, и следовать инструкциям, появляющимся на экране. Программа установки имеет стандартный интерфейс и помогает пользователю установить все компоненты прикладного решения. Подробнее о механизме создания дистрибутивов можно прочитать в книге «Инструменты для создания тиражируемых приложений «1С:Предприятия 8.2» из серии «Профессиональная разработка».

Функциональные опции

Функциональные опции позволяют разработчику выделить некоторую часть функциональности прикладного решения, которую можно оперативно включать или выключать на этапе внедрения и/или в процессе работы системы.

Объекты конфигурации и их реквизиты можно привязать к функциональным опциям, добавленным в конфигурацию. Отображение этих объектов в интерфейсе приложения зависит от того, включена связанная с ними функциональная опция или нет.

Функциональные опции могут использоваться с параметрами. В этом случае вид конкретной формы может зависеть от значения параметра, выбранного в форме.

Таким образом, в зависимости от потребностей конкретной организации внедренец или администратор прикладного решения в режиме 1С:Предприятие может включить или выключить определенные функциональные опции, ограничив тем самым функциональность прикладного решения задачами, решаемыми конкретным пользователем.

Показатели производительности и режим имитации задержек при вызове сервера

В процессе отладки и работы конфигурации разработчик может использовать показатели производительности и режим имитации задержек при вызове сервера в качестве инструментов для оптимизации клиент-серверного взаимодействия в разрабатываемом прикладном решении.

Показатели производительности прикладного решения позволяют разработчику оценить, насколько эффективно работает прикладное решение. Они отражают информацию о количестве текущих и накопленных вызовов сервера, длительности вызовов сервера, объеме принятых и отправленных данных.

Окно показателей производительности всплывает в левом нижнем углу монитора при выполнении каких-либо действий, приводящих к вызову сервера в процессе работы «1С:Предприятия». Это окно можно мышью перетащить в любое другое место экрана.

При запуске сеанса «1С:Предприятия» из конфигуратора отображение показателей производительности включено по умолчанию. В текущем сеансе в режиме 1С:Предприятие также можно включить отображение окна показателей производительности. Установить данный режим можно также при запуске «1С:Предприятия» из командной строки с соответствующим параметром.

Режим имитации задержек при вызове сервера позволяет разработчику оценить реальную скорость работы его прикладного решения у клиента. При включении этого режима платформа (даже в файловом варианте на локальном компьютере) будет работать с теми временными задержками, с которыми работает реальный канал связи.

Установка режима имитации задержек при вызове сервера позволяет разработчику превратить быстрый канал связи в медленный. Причем сделать это можно как при запуске приложения в настройках конфигуратора, так и в процессе его работы в режиме 1С:Предприятие. При этом режим, в котором было запущено приложение, не изменится. Однако включить режим имитации задержек при вызове сервера можно только для тонкого клиента и толстого клиента, запущенного в режиме управляемого приложения.

Установить данный режим можно также при запуске «1С:Предприятия» из командной строки с соответствующим параметром.

Подробнее об использовании показателей производительности и режима имитации задержек при вызове сервера можно прочитать в книге «Разработка управляемого интерфейса» (ч. 4) из серии «Профессиональная разработка».

1С:Библиотека стандартных подсистем 8.2

«1С:Библиотека стандартных подсистем 8.2» (БСП) - это инструмент, облегчающий разработку прикладных решений. Она содержит набор универсальных функциональных подсистем и технологию для разработки прикладных решений на платформе «1С:Предприятие 8.2». С помощью библиотеки стандартных подсистем можно быстро создавать новые конфигурации с уже готовой базовой функциональностью, а также включать готовые функциональные блоки в существующие конфигурации.

Подсистемы, содержащиеся в библиотеке стандартных подсистем, включают различные области разработки: администрирование информационной базы, администрирование пользователей системы, настройка доступа к данным информационной базы, ведение различной нормативно-справочной информации (адресный классификатор, курсы валют, календарные графики и пр.). БСП предоставляет базовые пользовательские и программные интерфейсы для работы с задачами и бизнес-процессами, прикрепляемыми файлами и электронными цифровыми подписями, контактной информацией, дополнительными реквизитами и сведениями, почтовыми сообщениями и пр. Они могут быть использованы в разрабатываемой конфигурации как все вместе, так и по отдельности.

Обзор функциональности средств администрирования

Средства администрирования, входящие в состав интерфейса разработки и администрирования, позволяют управлять составом пользователей информационной базы, осуществлять регламентные операции по ее обслуживанию и выполнять другие административные действия (рис. 2.9).

Перечислим основные средства, используемые для администрирования информационных баз «1С:Предприятия».

Список пользователей

Платформа «1С:Предприятие» позволяет вести список пользователей, которым разрешена работа с данной информационной базой. Этот список не является частью прикладного решения, а создается отдельно, в каждой конкретной организации, в которой используется прикладное решение. Существует возможность добавлять, удалять пользователей системы, назначать им различные виды аутентификации, набор ролей, доступных в системе, режим запуска приложения, язык пользователя и пр.

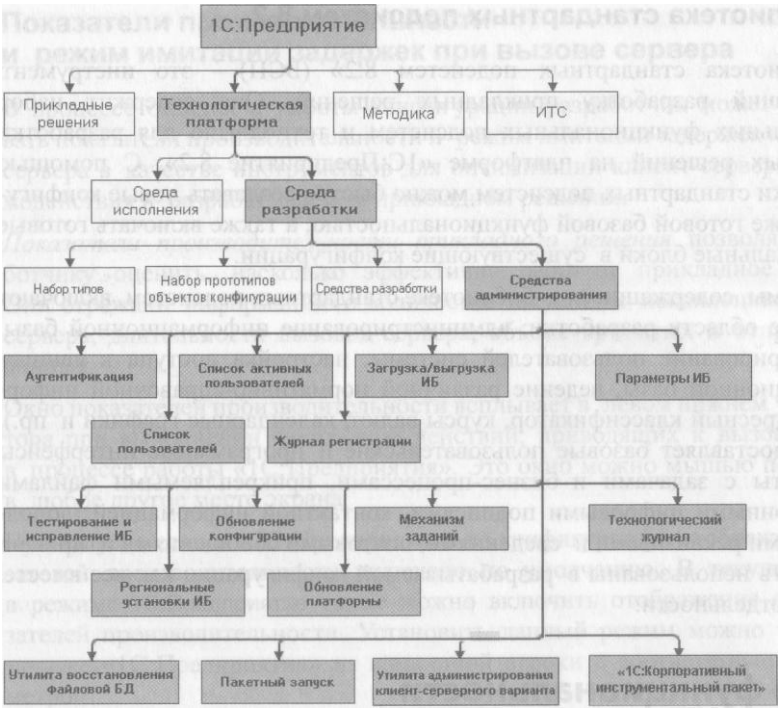


Рис. 2.9. Состав средств администрирования

Механизмы аутентификации

Механизм аутентификации позволяет определить, кто именно из пользователей, перечисленных в списке пользователей, подключается к прикладному решению. Эта информация может быть в дальнейшем использована для выбора соответствующего командного интерфейса пользователя, разрешения доступа к той или иной информации и пр. Система поддерживает два вида аутентификации: аутентификация «1С:Предприятия» и аутентификация операционной системы, которые могут использоваться в зависимости от конкретных задач, стоящих перед администратором информационной базы.

В случае аутентификации средствами «1С:Предприятия» в конфигураторе для каждого пользователя в списке задается пароль. Для выполнения аутентификации средствами «1С:Предприятия» пользователь при начале работы

с прикладным решением должен выбрать (или ввести) имя пользователя и соответствующий этому имени пароль.

В случае аутентификации средствами операционной системы в конфигураторе для пользователя выбирается один из пользователей операционной системы. Система анализирует, от имени какого пользователя операционной системы выполняется подключение к прикладному решению, и на основании этого определяет соответствующего пользователя «1С:Предприятия». При этом диалог аутентификации «1С:Предприятия» не отображается, если не указан специальный параметр командной строки.

Механизм заданий

Механизм заданий предназначен для выполнения административных действий по расписанию. Механизм заданий включает в себя фоновые и регламентные задания. Каждое задание представляет собой некоторую последовательность действий, описанных с помощью встроенного языка, содержащуюся в общих модулях.

Фоновые задания позволяют инициализировать выполнения процедур общих модулей асинхронно (без ожидания завершения). Регламентные задания позволяют организовать автоматический вызов процедур общих модулей по расписанию.

Механизм фоновых заданий реализуется средствами встроенного языка. Регламентные задания описываются на этапе конфигурирования и являются составной частью конкретного прикладного решения.

Система прав доступа

Для каждого конкретного прикладного решения система «1С:Предприятие» позволяет описать наборы прав, соответствующие должностям пользователей или виду деятельности. Структура прав определяется конкретным прикладным решением.

Все права, поддерживаемые системой «1С:Предприятие», можно разделить на две большие группы: основные и интерактивные. Основные права описывают действия, выполняемые над элементами данных системы или над всей системой в целом, и проверяются всегда, независимо от способа обращения к данным. Интерактивные права описывают действия, которые могут быть выполнены пользователем интерактивно. Соответственно, проверяются они только при выполнении интерактивных операций стандартными способами, причем в клиент-серверном варианте все проверки прав (кроме интерактивных) выполняются на сервере.

Ограничение прав на уровне записей и полей

Среди действий над объектами, хранящимися в базе данных, есть действия, отвечающие за чтение или изменение информации, хранящейся в базе данных. К таким действиям относятся:

- чтение - получение записей или их фрагментов из таблицы базы данных;
- добавление - добавление новых записей без изменения существующих;
- изменение - изменение существующих записей;
- удаление - удаление некоторых записей без внесения изменений в оставшиеся.

Для этих действий могут быть заданы дополнительные условия на данные (ограничение доступа к данным). В этом случае над конкретным объектом, хранимым в базе данных, может быть выполнено запрошенное действие только в том случае, если ограничение доступа к данным для данных этого объекта принимает значение Истина. Аналогичные условия могут быть заданы и для таблиц базы данных, не имеющих объектной природы (регистров).

Ограничение доступа к данным представляет собой условие, описанное на языке, который является подмножеством языка запросов. Это условие применяется для каждой записи таблицы базы данных, над которой выполняется операция. При просмотре списков и формировании отчетов существует возможность обеспечить отображение только тех данных, доступ к которым пользователю разрешен.

Журнал регистрации

Для хранения информации о событиях, происходящих в информационной базе, система «1С:Предприятие» ведет журнал регистрации. Администратор информационной базы имеет возможность настроить уровень событий, которые будут отображаться в журнале регистрации (ошибки, и/или предупреждения, и/или информация, и/или примечания). Просмотр журнала регистрации может быть выполнен с помощью специального фильтра, позволяющего отбирать информацию по большому количеству критериев. Например, можно отобрать только информацию о том, какие документы изменялись определенным пользователем в заданный промежуток времени. Также журнал регистрации может быть полностью или частично выгружен в формат XML для программного анализа.

Загрузка/выгрузка информационной базы

Команды загрузки/выгрузки информационной базы позволяют сохранить информационную базу в файл на диске или загрузить информационную базу из файла. Эти команды используются, например, для того, чтобы перенести базу из файлового варианта в клиент-серверный и обратно.

Утилита администрирования кластера серверов

Утилита администрирования кластера серверов используется при работе в клиент-серверном варианте и позволяет выполнять такие действия, как:

- создание, изменение и удаление кластеров серверов;
- модификация существующих кластеров: создание, изменение параметров и удаление рабочих серверов, рабочих процессов кластера, информационных баз, входящих в состав кластера;
- управление списками администраторов центральных серверов кластеров и списками администраторов кластеров;
- мониторинг соединений пользователей с информационными базами и служебных соединений;
- отключение пользователей от информационной базы;
- мониторинг объектных блокировок «1С:Предприятия» и блокировок клиентского соединения;
- оперативный анализ транзакционных блокировок системы управления базами данных;
- управление блокировкой установки соединений пользователей с информационной базой;
- управление блокировкой выполнения регламентных заданий.

Утилита представляет собой подключаемый модуль MMC (Microsoft Management Console) и может быть использована на компьютерах, на которых установлено соответствующее программное обеспечение (для операционных систем Windows 2000/XP/Server 2003 это программное обеспечение является стандартным).

Подробнее об утилите администрирования кластера серверов можно прочитать в разделе «Утилита администрирования кластера серверов» на стр. 237.

«1 С:Корпоративный инструментальный пакет»

Этот продукт предназначен для повышения производительности, масштабируемости и надежности информационных систем на платформе «1С:Предприятие». Он используется для нагрузочного тестирования и последующего сопровождения крупных корпоративных систем. Продукт «1С:КИП» включает в себя следующие инструменты:

- *1С:Центр управления производительностью* — инструмент мониторинга и анализа производительности информационных систем на платформе «1С:Предприятие». Этот инструмент предназначен для оценки производительности системы, сбора подробной технической информации об имеющихся проблемах производительности и анализа этой информации с целью дальнейшей оптимизации.
- *1С:Тест-центр* - инструмент автоматизации многопользовательских нагрузочных испытаний информационных систем на платформе «1С:Предприятие». С его помощью можно моделировать работу предприятия без участия реальных пользователей, что позволяет оценивать применимость, производительность и масштабируемость информационной системы в реальных условиях.
- *1С:Центр контроля качества.* Инструмент позволяет гарантировать эффективную и надежную работу системы на платформе «1С:Предприятие» за счет своевременного и правильного технического обслуживания системы. Это особенно важно для внедрений с высокими требованиями по производительности и надежности системы.

Глава 3. Использование встроенного языка

Введение

Как уже говорилось выше, прикладное решение «1С:Предприятия» представляет собой определенную структуру метаданных, которая исполняется технологической платформой. В то же время технологическая платформа содержит в себе интерфейс разработки и администрирования, позволяющий создавать новые и модифицировать существующие прикладные решения.

Разработка прикладного решения заключается по большому счету в двух основных действиях: визуальном конструировании объектов конфигурации и описании специфического поведения системы с использованием встроенного языка и языка запросов.

Создание нового прикладного решения начинается с добавления новой информационной базы в список баз «1С:Предприятия». При этом платформа создает некую «базовую» структуру метаданных, которая уже представляет собой работоспособное прикладное решение. В соответствии с этой структурой метаданных платформа также создает ряд информационных структур (таблиц), обеспечивающих работу этой конфигурации и хранение данных. Информационные структуры создаются в том варианте работы, который выбран пользователем (например, файловый или клиент-серверный).

Таким образом, разработчик начинает создание прикладного решения не с нуля, а с некоторой «базовой» структуры метаданных (пустой конфигурации), которую он может изменять в соответствии со своими потребностями.

Следующим шагом в разработке прикладного решения является, как правило, создание структур для хранения данных. Отличительной особенностью разработки в системе «1С:Предприятие» является то, что платформа изолирует разработчика от физических деталей хранения данных, предлагая более высокий уровень абстракции - уровень объектов метаданных. Другими словами, разработчик добавляет в структуру метаданных нужные ему объекты, определяет, например, какие реквизиты и табличные части будет содержать тот или иной объект. Платформа же анализирует изменения, выполняемые разработчиком, и создает в информационной базе соответствующие информационные структуры, позволяющие хранить и эффективно использовать данные. В результате с точки зрения разработчика процесс создания прикладного решения в файловом или в клиент-серверном варианте выглядит совершенно одинаково, так как он не оперирует физическими таблицами базы данных, а имеет дело с более абстрактными сущностями - объектами метаданных.

Таким образом, вся структура хранимых данных описывается средствами визуального конфигурирования и не может быть изменена в процессе работы прикладного решения (в реальном режиме времени). Такой подход позволяет платформе контролировать целостность информационной базы и данных, которые в ней хранятся. Платформе известно о назначении того или иного объекта конфигурации, она знает, в каких таблицах должны храниться данные этого объекта, и обладает информацией о том, как эти данные должны взаимодействовать с данными других объектов конфигурации. Таким образом, с одной стороны ограничивая разработчика в динамической модификации структуры информационной базы, она значительно облегчает задачу создания и модификации прикладного решения, избавляя разработчика от необходимости самостоятельно контролировать правильность таблиц информационной базы и их связей.

Другим положительным моментом такого подхода является то, что структура прикладного решения становится прозрачной и легко читаемой. Любому разработчику, не принимавшему непосредственного участия в разработке данного решения, достаточно взглянуть на структуру объектов метаданных, чтобы понять в общих чертах, как оно устроено. Открыв конфигуратор, он увидит привычный набор объектов метаданных, назначение каждого из которых ему хорошо известно.

Ограниченный набор объектов конфигурации, которые может использовать разработчик, является еще одной особенностью прикладных решений «1С:Предприятия». Платформа поддерживает фиксированный набор прототипов объектов метаданных (например, справочник, документ, регистр накопления, бизнес-процесс и т.д.). Разработчик не имеет возможности

каким-либо образом создать собственный объект конфигурации (например, просто таблицу, содержащую набор колонок). Он может только добавить в прикладное решение новый объект конфигурации, соответствующий одному из прототипов, поддерживаемых системой (например, регистр сведений, обладающий набором реквизитов). Такой подход позволяет создать среду описания прикладных решений, состоящую из логических сущностей, одинаковых для всех прикладных решений (рис. 3.1).

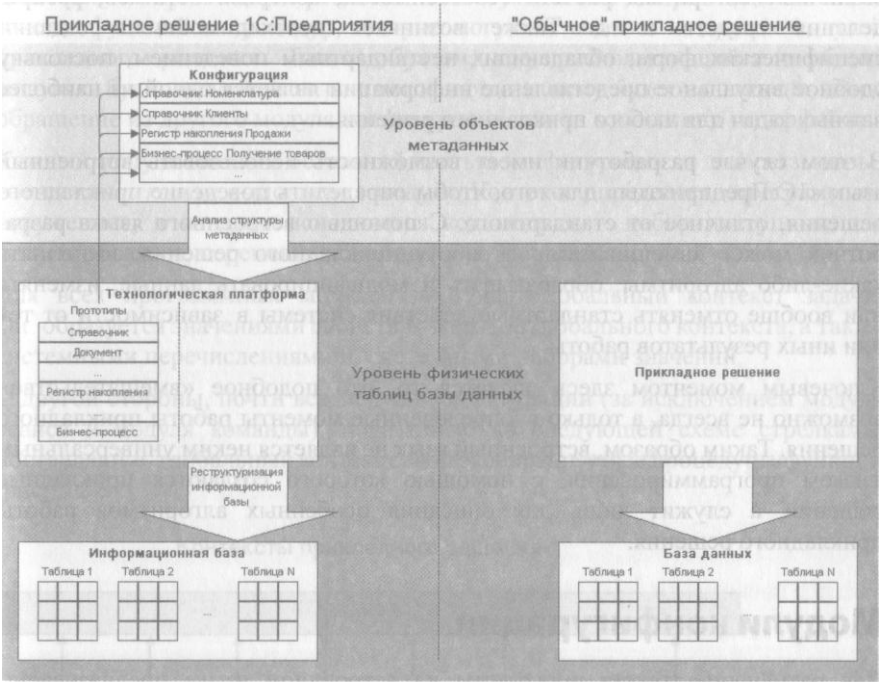


Рис. 3.1. Сравнение логических уровней приложения

Добавление новых объектов конфигурации в прикладное решение сразу же позволяет использовать их для ввода и модификации информации. Технологическая платформа «знает», как работать с данными того или иного объекта конфигурации, и в определенные моменты работы прикладного решения самостоятельно может сгенерировать нужные формы для работы с данными.

Таким образом, все объекты конфигурации, которые может использовать разработчик, обладают некоторым базовым поведением, которое реализуется платформой без участия разработчика. В простейшем варианте создание

функционального и работоспособного прикладного решения возможно исключительно средствами визуального конструирования, без написания какого-либо текста программы. Платформа проанализирует состав объектов конфигурации и обеспечит все необходимые функции для работы с ними.

Однако реальные задачи, решаемые с помощью системы «1С:Предприятие», всегда требуют наличия в прикладном решении некоторых алгоритмов, специфичных для автоматизируемой области. Это могут быть, например, различные алгоритмы расчета себестоимости, контроля остатков, распределения средств и т.д. Также возникает необходимость в создании специфических форм, обладающих нестандартным поведением, поскольку удобное визуальное представление информации является одной из наиболее важных задач для любого прикладного решения.

В этом случае разработчик имеет возможность использовать встроенный язык «1С:Предприятия» для того, чтобы определить поведение прикладного решения, отличное от стандартного. С помощью встроенного языка разработчик может «вмешиваться» в работу прикладного решения: выполнять какие-либо алгоритмы, обрабатывать и модифицировать данные, изменять или вообще отменять стандартные действия системы в зависимости от тех или иных результатов работы.

Ключевым моментом здесь является то, что подобное «вмешательство» возможно не всегда, а только в определенные моменты работы прикладного решения. Таким образом, встроенный язык не является неким универсальным языком программирования, с помощью которого создаются прикладные решения, а служит лишь для описания особенных алгоритмов работы прикладного решения.

Модули конфигурации

Для размещения текста программы на встроенном языке предназначены модули прикладного решения (например, модуль приложения, общие модули, модули объектов, модули форм, модули команд, модули менеджеров и т.д.). Эти модули располагаются в различных местах конфигурации и имеют различное назначение.

Большинство модулей привязано к определенным объектам конфигурации (например, к формам) или к самому прикладному решению (например, модуль управляемого приложения). Такие модули вызываются в определенные моменты работы прикладного решения (например, модуль управляемого приложения, который запускается при запуске системы «1С:Предприятие» в режиме управляемого приложения, или модуль объекта справочника Номенклатура, который запускается при создании объекта

(элемента) справочника). Таким образом, если модуль содержит некоторый код, то этот код будет выполнен и работа прикладного решения продолжена. В таких модулях могут располагаться процедуры обработки событий, определенных для различных объектов прикладного решения. Эти процедуры также будут выполняться при наступлении соответствующего события.

Наряду с модулями, вызываемыми в процессе работы прикладного решения, существуют общие модули, которые сами по себе не вызываются в процессе работы прикладного решения. Они служат лишь для размещения в них текстов функций и процедур, которые могут вызываться из других модулей прикладного решения. Таким образом, код, располагающийся в общих модулях, будет исполнен только тогда, когда к нему будет выполнено явное обращение из другого модуля конфигурации или из командного интерфейса.

Каждый модуль связан с остальной частью конфигурации, и эта связь называется *контекстом* выполнения модуля. Контекст определяет «программное окружение», в котором исполняется код модуля, - набор доступных для модуля объектов, переменных, процедур и функций.

Для всех программных модулей доступен глобальный контекст задачи. Он образуется значениями свойств и методов глобального контекста, а также системными перечислениями и системными наборами значений.

С другой стороны, почти все модули конфигурации (за исключением модуля сеанса и модуля команды), показанные на следующей схеме стрелками, поставляют в глобальный контекст свои экспортируемые процедуры/функции (рис. 3.2).

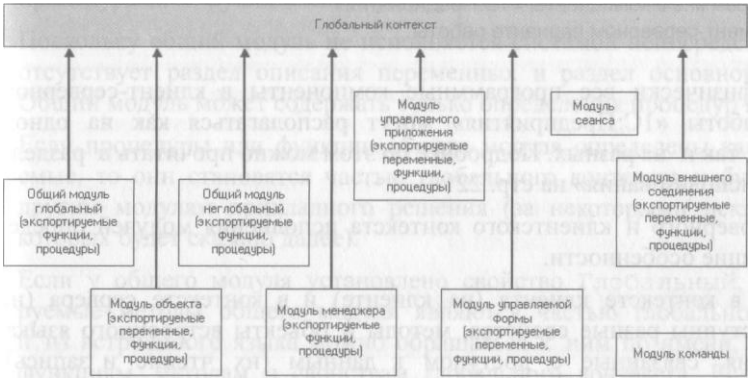


Рис. 3.2. Контексты прикладного решения

Локальный контекст модуля образуется тем конкретным местом конфигурации задачи, для которого использован программный модуль. Локальный контекст виден только конкретному программному модулю и определяет для модуля набор непосредственно доступных ему объектов, их свойств и методов.

Подробнее о модулях конфигурации можно прочитать в книге «^Предприятие 8.2. Руководство разработчика», ч. 1, раздел 4.2 «Формат исходных текстов программных модулей».

Контекст исполнения модулей

С точки зрения места исполнения модуля существует понятие «контекст исполнения модулей», определяющий программную среду, в которой выполняется модуль. Для клиент-серверного варианта работы «1С:Предприятия» таких контекстов два - контекст клиента и контекст сервера (рис. 3.3).



Рис. 3.3. Программные компоненты «1С:Предприятия» в клиент-серверном варианте работы

При этом физически все программные компоненты в клиент-серверном варианте работы «1С:Предприятия» могут располагаться как на одном компьютере, так и на разных. Подробнее об этом можно прочитать в разделе «Варианты использования» на стр. 221.

Наличие серверного и клиентского контекста исполнения модулей определяет следующие особенности.

Во-первых, в контексте клиента (на клиенте) и в контексте сервера (на сервере) доступны разные свойства, методы и объекты встроенного языка. Все действия, связанные с доступом к данным (их чтение и запись), возможны только на сервере, а отображение этих данных пользователю и другие интерактивные действия возможны только на клиенте. Поэтому

клиентские процедуры в модулях в явном виде отделяются от серверных, и в них используется ограниченный состав объектной модели встроенного языка.

Во-вторых, реальным вариантом работы клиент-серверного варианта будет ситуация, когда все программные компоненты «1С:Предприятия» расположены на отдельных компьютерах. Как правило, сервер «1С:Предприятия» и сервер СУБД находятся в пределах одной локальной сети, а вот клиентский компьютер с клиентским приложением может находиться как в этой же сети, так и подключаться к серверу через Интернет с использованием самых разных каналов связи с разной скоростью передачи информации.

В процессе работы «1С:Предприятия» клиентское приложение все время вызывает сервер для выполнения каких-либо действий, сервер выполняет их и возвращает управление и результат этих действий обратно, на клиента. Механизмы платформы оптимизируют клиент-серверное взаимодействие для того, чтобы обеспечить приемлемую скорость работы системы даже на низкоскоростных каналах связи. Поэтому разработчик, организующий клиент-серверное взаимодействие путем вызова серверных процедур из процедур, исполняющихся на клиенте, также должен стремиться к такой оптимизации. Этому вопросу посвящена четвертая глава книги «Разработка управляемого интерфейса» из серии «Профессиональная разработка».

Общий модуль

В конфигурации может быть определено произвольное количество общих модулей, в том числе и ни одного. Контекст общего модуля образуется глобальным контекстом и локальным контекстом самого общего модуля, т. е. процедурами и функциями, определенными внутри общего модуля.

Поскольку общий модуль не исполняется системой непосредственно, в нем отсутствует раздел описания переменных и раздел основной программы. Общий модуль может содержать только определения процедур и функций.

Если процедуры или функции общего модуля определены как экспортируемые, то они становятся частью глобального контекста и будут доступны другим модулям прикладного решения (за некоторыми исключениями, о которых будет сказано далее).

Если у общего модуля установлено свойство `Глобальный`, то экспортируемые методы общего модуля являются частью глобального контекста, и из встроенного языка можно обращаться к ним по имени, как к другим функциям, методам и свойствам глобального контекста. Если же модуль неглобальный, то при обращении к его экспортируемой процедуре или функции ее имя нужно указывать через точку от имени модуля (например,

Обмен. ПолучитьПрефиксНомера0). Последний способ предпочтительнее, так как неглобальные общие модули компилируются по мере обращения к ним, а не при запуске системы, как глобальные.

Поскольку общий модуль не привязан к какому-либо объекту конфигурации, а относится ко всему прикладному решению, имена экспортируемых процедур и функций должны быть различными в разных общих модулях. В противном случае будет выдана синтаксическая ошибка, т.к. глобальный контекст будет содержать повторяющиеся имена.

С помощью свойств Клиент (управляемое приложение) и Сервер, а также указаний препроцессору можно организовывать выполнение различных процедур и функций общих модулей на сервере приложения или на клиентском месте.

По умолчанию в свойствах общих модулей устанавливается флажок Сервер. Это означает, что все процедуры и функции общего модуля будут доступны только на сервере. Если при этом установить свойство Вызов сервера, то процедуры и функции этого общего модуля станут доступны на клиенте. Соответственно, если у общего модуля установлены свойства Клиент (управляемое приложение) и/или Внешнее соединение, то экспортируемые (определенные с ключевым словом Экспорт) процедуры и функции этого общего модуля будут доступны на клиенте (тонком клиенте, веб- клиенте, толстом клиенте в режиме управляемого приложения) и/или во внешнем соединении.

Если у неглобального общего модуля установлено свойство Клиент (управляемое приложение), то в контексте этого модуля будут доступны экспортируемые переменные, процедуры и функции модуля управляемого приложения. Если у неглобального общего модуля установлено свойство Внешнее соединение, то в контексте этого модуля будут доступны экспортируемые переменные, процедуры и функции модуля внешнего соединения.

Также у неглобальных общих модулей доступно свойство Повторное использование возвращаемых значений. Установка этого свойства ускоряет выполнение экспортируемых функций (только функций, а не процедур) неглобальных общих модулей. Это происходит за счет использования механизма кеширования, так как значения параметров, переданных в функцию, и возвращенный ей при этом результат запоминаются и сохраняются для дальнейшего использования.

Модуль управляемого приложения

Модуль управляемого приложения выполняется при запуске системы «1С:Предприятие» в управляемом режиме (свойство конфигурации
Основной режим запуска установлено в значение Управляемое приложение, т.е. в режимах тонкого клиента, веб-клиента и толстого клиента

в режиме управляемого приложения) или при обращении к приложению как к Automation-серверу (объект V82c.Application).

Этот модуль предназначен для отработки действий, связанных с сеансом работы конечного пользователя. Помимо описания переменных и основной программы, модуль приложения может содержать описание процедур- обработчиков событий, связанных с началом и окончанием сеанса работы пользователя.

ПРИМЕЧАНИЕ

> В конфигурации существует также модуль обычного приложения, который выполняется при запуске «1 С: Предприятия» в обычном режиме. Он остался от прежних версий платформы и в рамках этой книги рассматриваться не будет. Далее везде речь пойдет об управляемом клиенте (тонком клиенте, веб-клиенте, толстом клиенте в режиме управляемого приложения), управляемом приложении и об управляемых формах, даже если слово «управляемое» опущено.

В конфигурации всегда существует единственный модуль управляемого приложения, который располагается в корневом разделе конфигурации. Контекст модуля управляемого приложения (рис. 3.4) образуется:

- свойствами и методами глобального контекста, которые доступны в управляемых клиентах (тонкий клиент, веб-клиент, толстый клиент в режиме управляемого приложения);
- экспортируемыми функциями и процедурами клиентских общих модулей (у которых установлено свойство `Клиент (управляемое приложение)`) и неглобальных серверных общих модулей, для которых установлено свойство `Вызов сервера);`
- локальным контекстом самого модуля управляемого приложения. Экспортируемые

переменные, процедуры или функции модуля управляемого приложения (у которых в заголовке указано ключевое слово `Экспорт`) будут доступны (рис. 3.5):

- в неглобальных клиентских общих модулях с установленным свойством `Клиент (управляемое приложение);`
- клиентских процедурах и функциях модуля управляемой формы;
- клиентских процедурах и функциях модуля команды.

Модуль управляемого приложения недоступен для процедур, работающих на сервере. Также экспортируемые переменные, процедуры или функции модуля управляемого приложения недоступны для модуля внешнего соединения и глобальных общих модулей.

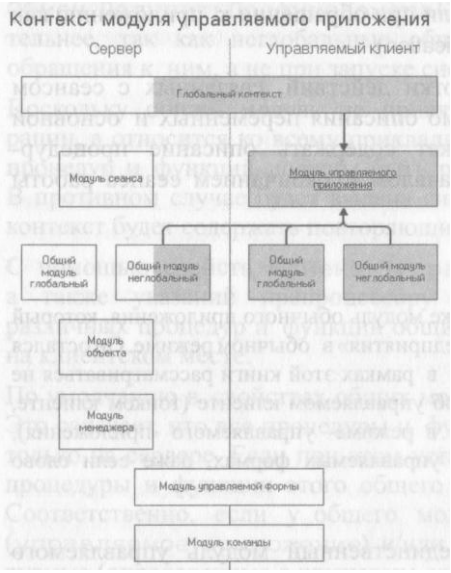


Рис. 3.4. Контекст модуля управляемого приложения

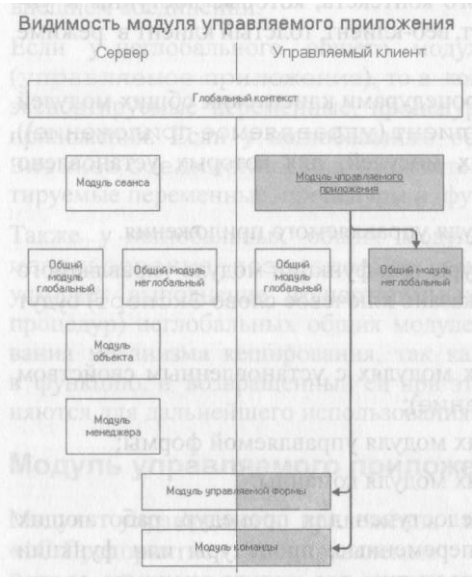


Рис. 3.5. Видимость модуля управляемого приложения

Основными событиями, которые могут обрабатываться в модуле приложения, являются события начала и окончания работы приложения. Последовательность их вызова представлена на рис. 3.6.

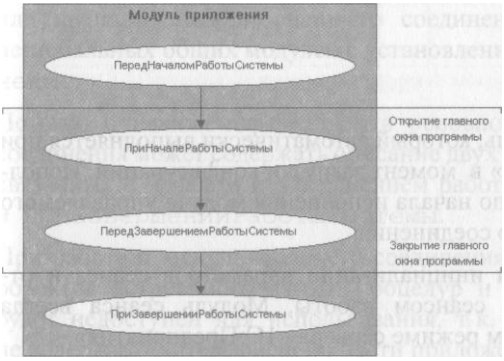


Рис. 3.6. Последовательность вызова событий модуля приложения

Событие `ПередНачаломРаботыСистемы` возникает при запуске системы в управляемом режиме «1С:Предприятие» до открытия главного окна. Обрабатывая это событие, разработчик имеет возможность отказаться от запуска системы в случае, если какие-либо условия не выполнены. Следует учитывать, что, поскольку это событие вызывается в тот момент, когда главное окно программы еще не открыто, будет недоступен ряд действий, требующих наличия главного окна (например, выдача сообщений, открытие форм и т.д.).

Событие `ПриНачалеРаботыСистемы` возникает при запуске системы в управляемом режиме 1С: Предприятие после открытая главного окна. В обработчике этого события разработчик уже не может отказаться от запуска системы, зато может выполнить действия, которые обязательно должны быть выполнены при начале работы пользователя системы (например, открыть форму, содержащую справочную информацию, и т. д.).

Событие `ПередЗавершениемРаботыСистемы` возникает при завершении работы системы в управляемом режиме 1С: Предприятие до закрытия главного окна. Обрабатывая это событие, разработчик имеет возможность отказаться от завершения работы, если какие-либо условия не выполнены.

Событие `ПриЗавершенииРаботыСистемы` возникает при завершении работы системы в управляемом режиме 1С: Предприятие после закрытия главного окна. В обработчике этого события разработчик уже не может отказаться от закрытия приложения, но может выполнить действия, которые

обязательно должны быть выполнены при окончании работы пользователя. Следует учитывать, что, поскольку это событие вызывается в тот момент, когда главное окно программы уже закрыто, будет недоступен ряд действий, требующих наличия главного окна (например, выдача сообщений, открытие форм и т.д.).

Модуль сеанса

Модулем сеанса называется модуль, который автоматически выполняется при старте системы «1С:Предприятие» в момент загрузки конфигурации. Исполнение модуля сеанса происходит до начала исполнения модуля управляемого приложения или модуля внешнего соединения.

Модуль сеанса предназначен для инициализации параметров сеанса и отработки действий, связанных с сеансом работы. Модуль сеанса всегда исполняется в привилегированном режиме сервера «1С:Предприятия».

Установка параметров сеанса выполняется в обработчике события `УстановкаПараметровСеанса()`.

Модуль сеанса может содержать только определения неэкспортируемых процедур и функций и может использовать процедуры из общих модулей конфигурации.

Модуль внешнего соединения

В конфигурации всегда существует единственный модуль внешнего соединения, который располагается в корневом разделе конфигурации. Контекст модуля внешнего соединения образуется:

- глобальным контекстом, в том числе экспортируемыми функциями и процедурами общих модулей (если для этих модулей установлено свойство `Внешнее соединение`);
- локальным контекстом самого модуля внешнего соединения.

Модуль внешнего соединения выполняется при обращении к приложению как к COM-серверу (в режиме внешнего соединения). В режиме внешнего соединения запускается не полноценное приложение «1С:Предприятия», а «облегченный» вариант приложения, в котором недоступны все функции, так или иначе связанные с организацией пользовательского интерфейса. Поэтому в режиме внешнего соединения вместо модуля приложения исполняется модуль внешнего соединения. Этот модуль предназначен для отработки действий, связанных с сеансом работы с приложением «1С:Предприятия».

Если переменные, процедуры или функции модуля внешнего соединения определены как экспортируемые, то они будут доступны другим модулям внешнего соединения, за исключением глобальных общих модулей, в которых они доступны не будут. Экспортируемые переменные, процедуры или функции модуля внешнего соединения также будут доступны для неглобальных общих модулей с установленным свойством `Внешнее соединение`.

Помимо описания переменных и основной программы, модуль внешнего соединения может содержать описание двух процедур-обработчиков событий, связанных с началом и завершением работы: `ПриНачалеРаботыСистемы` и `ПриЗавершенииРаботыСистемы`.

При работе в модуле внешнего соединения следует помнить о том, что ряд объектов встроенного языка, процедур и функций глобального контекста будет недоступен для использования, т.к. COM-сервер «1С:Предприятия» не содержит «интерфейсной» части приложения.

Например, в режиме внешнего соединения нельзя использовать такие объекты, как диаграмма, табличный документ, недоступны функции для вызова диалога ввода данных и т.д. Точная информация о возможности использования объектов, процедур и функций в модуле внешнего соединения находится в документации, в описании конкретных объектов, процедур и функций.

Модуль объекта

Каждый прикладной объект конфигурации, данные которого могут быть модифицированы в режиме 1С:Предприятие, имеет свой модуль. Этот модуль исполняется при создании объекта встроенного языка, который позволяет модифицировать данные объекта конфигурации. Соответствующий объект встроенного языка создается, например, при вводе нового объекта, при копировании, при получении данных существующего объекта и т. д. Для различных объектов конфигурации этот модуль имеет разное название (табл. 3.1).

Контекст модуля объекта (рис. 3.7) образуется:

- глобальным контекстом, в том числе экспортируемыми функциями и процедурами общих модулей, скомпилированных на сервере (для этих модулей установлено свойство `Сервер`);
- свойствами и методами объекта встроенного языка, контекст которого расширяется модулем;
- реквизитами объекта конфигурации, которому принадлежит модуль;
- локальным контекстом самого модуля объекта.

Таблица 3.1. Название модуля объекта для различных объектов конфигурации

Объект конфигурации	Название модуля	Объект встроенного языка, который расширяется модулем
Константа	Модуль менеджера значения	КонстантаМенеджерЗначения.<имя>
Справочник Документ Отчет Обработка План видов характеристик План счетов План видов расчета План обмена Бизнес-процесс Задача	Модуль объекта	СправочникОбъект.<имя> ДокументОбъект.<имя> ОтчетОбъект.<имя> ОбработкаОбъект. <имя> ПланВидовХарактеристикОбъект.<имя> ПланСчетовОбъект.<имя> ПланВидовРасчетаОбъект.<имя> ПланОбменаОбъект.<имя> БизнесП роцессОбъект.<имя> ЗадачаОбъект.<имя>
Последовательность Регистр сведений Регистр накопления Регистр бухгалтерии Регистр расчета Перерасчет	Модуль набора записей	ПоследовательностьНаборЗаписей.<имя> Регистре веденийНаборЗаписей.<имя> РегистрНакопленияНаборЗаписей.<имя> РегистрБухгалтерииНаборЗаписей.<имя> РегистрРасчетаНаборЗаписей.<имя> ПерерасчетНаборЗаписей.<имя>

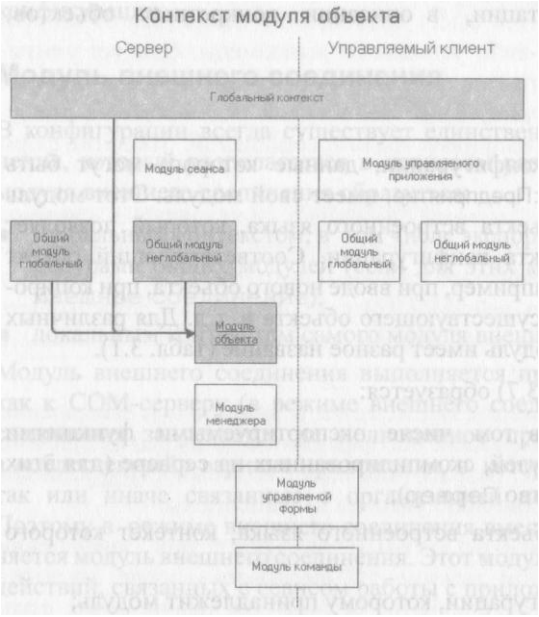


Рис. 3.7. Контекст модуля объекта

Если переменные, процедуры или функции модуля объекта определены как экспортируемые, то они будут доступны в качестве свойств и методов соответствующих объектов встроенного языка.

Например, пусть в модуле справочника ТиповыеАнкеты определена экспортируемая процедура (листинг 3.1).

Листинг 3.1. Экспортируемая процедура в модуле справочника

```
Процедура ПроверитьКоррекгностьЗаполненияАнкетыО Экспорт

    II Алгоритм вывода на экран печатной формы анкеты
    //...

КйнецПроцедуры //
```

Тогда возможен следующий вызов этой процедуры, например, из модуля внешней обработки (листинг 3.2).

Листинг 3.2. Пример вызова экспортируемой процедуры объекта справочника

```
Анкета = Справочники.ТиповыеАнкеты.НайтиПоКоду("000000001").ПолучитьОбъект();
Анкета. ПроверитьКоррекгностьЗаполненияАнкеты();
```

Помимо описания переменных и основной программы, модуль объекта может содержать описание процедур-обработчиков событий, связанных с данным объектом конфигурации. Состав этих событий различен для разных объектов, однако есть три события, которые вызываются для всех объектов, - `ОбработкаПроверкиЗаполнения`, `ПередЗаписью` и `ПриЗаписи`. Последовательность их вызова представлена на рис. 3.8.

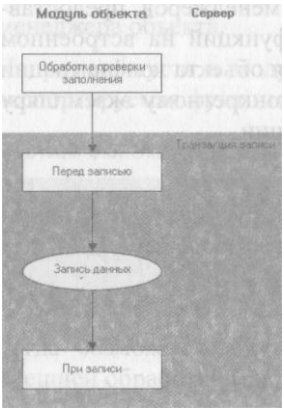


Рис. 3.8. Последовательность вызова событий модуля объекта

Событие ОбработкаПроверкиЗаполнения вызывается перед записью данных объекта, до начала транзакции записи. В обработчике этого события разработчик может реализовать собственные алгоритмы проверки заполнения реквизитов объекта, добавить в массив проверяемых реквизитов дополнительные реквизиты, чтобы платформа выполнила их проверку, или очистить массив проверяемых реквизитов, отказавшись от их стандартной проверки. Установив параметр Отказ в значение Истина, разработчик может отказаться от записи объекта, если, например, какие-то условия проверки не были выполнены.

Событие ПередЗаписью вызывается перед записью данных, после начала транзакции записи, но до начала непосредственной записи данных в базу данных. В обработчике этого события разработчик имеет возможность отказаться от записи данных, если, например, не выполнены требуемые условия.

Событие ПриЗаписи вызывается после того, как была выполнена запись данных в базу данных, но до окончания транзакции записи. В обработчике этого события выполняются действия над данными, неразрывно связанными с данными объекта, которые не могут быть изменены отдельно от основных данных. Здесь также разработчик может отказаться от записи данных, если, например, в результате записи этих данных в базу нарушаются какие-либо условия.

Модуль менеджера объекта

Для каждого прикладного объекта существует менеджер, предназначенный для управления этим объектом как объектом конфигурации. С помощью менеджера можно создавать объекты, работать с формами и макетами. Модуль менеджера позволяет расширить функциональность менеджеров, предоставляемых системой, за счет написания процедур и функций на встроенном языке. Фактически это позволяет описать методы для объекта конфигурации (например, справочника), которые относятся не к конкретному экземпляру объекта базы данных, а к самому объекту конфигурации.

Контекст модуля менеджера (рис. 3.9) образуется:

- глобальным контекстом, в том числе экспортируемыми функциями и процедурами общих модулей, скомпилированных на сервере (для этих модулей установлено свойство Сервер);
- локальным контекстом самого модуля менеджера.

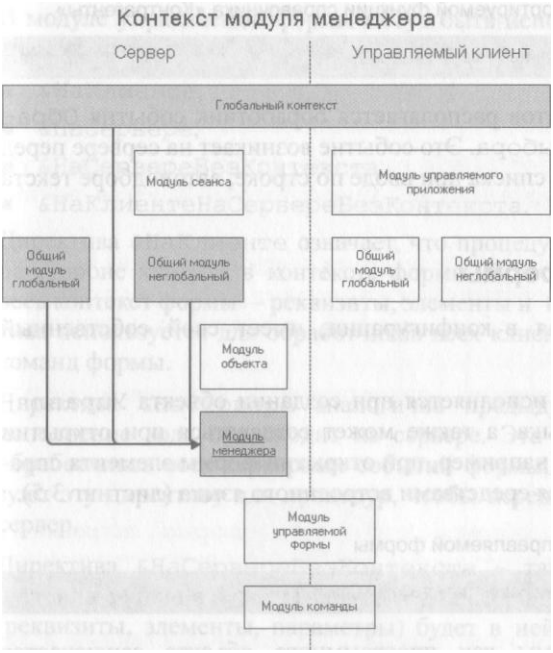


Рис. 3.9. Контекст модуля менеджера

Модуль менеджера исполняется только на сервере и не может иметь переменных и тела модуля. Если функции или процедуры модуля менеджера объявлены как экспортируемые, к ним можно будет получить доступ через менеджера объекта.

Например, пусть в модуле менеджера справочника Контрагенты определена экспортируемая функция (листинг 3.3).

Листинг 3.3. Экспортируемая функция в модуле менеджера справочника «Контрагенты» Функция ПолучитьСгшсокДебиторов() Экспорт

```
    //...
КонецФункц
ии
```

Тогда возможен следующий вызов этой функции, например, из модуля внешней обработки (листинг 3.4).

Листинг 3.4. Пример вызова экспортируемой функции справочника «Контрагенты»

```
Дебиторы = Справочники.Контрагенты.ПолучитьСписокДебиторов();
```

В модулях менеджеров объектов располагается обработчик события `ОбработкаПолученияДанныхВыбора`. Это событие возникает на сервере перед стандартным формированием списка при вводе по строке, автоподборе текста и быстром выборе.

Модуль управляемой формы

Каждая форма, определенная в конфигурации, имеет свой собственный модуль.

Модуль управляемой формы исполняется при создании объекта `УправляемаяФорма` встроенного языка, а также может создаваться при открытии формы прикладного объекта (например, при открытии формы элемента справочника) либо явно получаться средствами встроенного языка (листинг 3.5).

Листинг 3.5. Пример получения управляемой формы ФормаСписка =

```
ПолучитьФорму("Справочник.Номенклатура.ФормаСписка");
```

Основная особенность формы как программного объекта заключается в том, что она существует одновременно и на клиенте, и на сервере. Таким образом, процедуры, содержащиеся в этом модуле, исполняются или в контексте клиента, или в контексте сервера.

Поэтому все процедуры и функции, создаваемые разработчиком в модуле формы, должны иметь явное указание на то, в каком контексте они будут исполняться. Для этого используются *директивы компиляции,* которыми должно предваряться описание процедур (листинг 3.6).

Листинг 3.6. Процедура «ПриСозданииНаСервере()» &НаСервере

```
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
    Если Объект.Ссылка = ПланыОбмена.УдаленныеСклады.ЭтотУзел() Тогда
Элементы.Главный-Достулность = Ложь; Элементы. Склад. Доступность = Ложь;
Элементы.АдресОбмена.Доступность = Ложь; КонецЕсли; КонецПроцедуры
```

В приведенном примере стандартная процедура, выполняемая платформой при создании формы на сервере, предваряется соответствующей директивой компиляции `&НаСервере`.

В модуле управляемой формы могут быть использованы следующие директивы компиляции:

- &НаКлиенте,
- &НаСервере,
- &НаСервереБезКонтекста,
- &НаКлиентеНаСервереБезКонтекста.

Директива &НаКлиенте означает, что процедура или функция выполняется на стороне клиента в контексте формы. В этой процедуре будет доступен весь контекст формы - реквизиты, элементы и параметры формы. Эта директива используется для обработчиков всех клиентских событий и локальных команд формы.

Директива &НаСервере аналогична предыдущей с той разницей, что исполнение кода происходит на сервере. Эта директива используется для обработчиков всех серверных событий формы. Разработчики часто используют эту директиву для процедур, чтобы передать исполнение с клиента на сервер.

Директива &НаСервереБезКонтекста - также определяет, что процедура или функция будет выполняться на стороне сервера, но контекст формы (реквизиты, элементы, параметры) будет в ней недоступен. Эту директиву также используют разработчики для процедур, в которых нужно выполнить некоторые действия на сервере, но контекст формы при этом не нужен. Поскольку внеконтекстный вызов проще и эффективнее, то везде, где можно обойтись без затратного и сложного контекстного серверного вызова, рекомендуется использовать именно внеконтекстный вызов сервера.

Директива &НаКлиентеНаСервереБезКонтекста определяет, что процедура или функция будет выполняться как на стороне сервера, так и на стороне клиента, смотря откуда она вызвана. Контекст формы будет в ней недоступен. Эту директиву разработчики обычно используют тогда, когда нужно выполнить одинаковые действия при создании формы на сервере и в процессе ее функционирования на клиенте. В этом случае использование этой директивы позволяет избежать ненужного дублирования кода и лишних вызовов сервера.

Если по ошибке никакая директива компиляции перед процедурой не указана, то платформа будет использовать директиву &НаСервере.

В целом исполнение модуля формы на клиенте и на сервере характеризуется тем, что из клиентских процедур модуля формы можно вызывать серверные, тем самым передавая выполнение с клиента на сервер. После выполнения серверных процедур исполнение кода возвращается на клиента. Принудительно передать исполнение кода в обратную сторону, с сервера на клиента,

невозможно, то есть нельзя из серверных процедур модуля формы вызывать клиентские.

Контекст клиентских процедур и функций модуля управляемой формы (рис. 3.10) образуется:

- свойствами и методами глобального контекста, которые доступны в управляемых клиентах (тонкий клиент, веб-клиент, толстый клиент в режиме управляемого приложения);
- экспортируемыми функциями и процедурами клиентских общих модулей, у которых установлено свойство `Клиент (управляемое приложение)` и неглобальных серверных общих модулей, для которых установлено свойство `Вызов сервера;`
- экспортируемыми переменными, процедурами и функциями модуля управляемого приложения;
- свойствами и методами расширения формы, определяемого основным реквизитом формы;
- свойствами и методами объекта `УправляемаяФорма` встроенного языка;
- реквизитами формы, которой «принадлежит» модуль;
- локальным контекстом самого модуля формы, скомпилированным в управляемых клиентах.

Контекст серверных процедур и функций модуля управляемой формы (рис. 3.10) образуется:

- глобальным контекстом, в том числе экспортируемыми функциями и процедурами общих модулей, скомпилированных на сервере (для этих модулей установлено свойство `Сервер);`
- свойствами и методами расширения формы, определяемого основным реквизитом формы;
- свойствами и методами объекта `УправляемаяФорма` встроенного языка;
- реквизитами формы, которой «принадлежит» модуль;
- серверными процедурами и функциями самого модуля управляемой формы, а также переменными и телом модуля, скомпилированными на сервере.



Рис. 3.10. Контекст модуля управляемой формы

В контексте управляемой формы недоступны свойства и методы объекта, который назначен основным реквизитом формы. Получить к ним доступ можно только из серверной процедуры модуля формы, предварительно преобразовав данные формы в объект.

Например, пусть в модуле объекта справочника определена экспортируемая процедура ПересчитатьЦеныО (листинг 3.7).

Листинг 3.7. Экспортируемая процедура в модуле объекта справочника

```
Процедура ПересчитатьЦеныО Экспорт

    //...

КонецПроцедура
```

Тогда в серверной контекстной процедуре модуля формы элемента этого справочника можно преобразовать данные формы в объект (РеквизитФор- мыВЗначениеО) и выполнить метод этого объекта ПересчитатьЦеныО.

Затем, чтобы пересчитанные данные отразились в форме, нужно выполнить обратное преобразование (ЗначениеВРеквизитФормыО), конвертировав объект справочника в данные формы (листинг 3.8).

Листинг 3.8. Пример вызова экспортируемой процедуры справочника из модуля формы &НаСервере

```
Процедура ОбновитьДанные()

    СправочникОбъект =
    РеквизитФормыВЗначение("Объект");
    СправочникОбъект.ПересчитатьЦены();
    ЗначениеВРеквмзитФормы(СправочникОбъект, "Объект");

КонецПроцедуры
```

Если процедуры или функции модуля формы определены как экспортируемые, то они будут доступны в качестве методов соответствующих объектов УправляемаяФорма встроенного языка. При этом для этих процедур обязательно должна быть указана директива компиляции. Поскольку форма функционирует на клиенте, то практический смысл для описания контекста исполнения процедур имеет директива &НаКлиенте. Описание переменных также должно предваряться директивами компиляции.

Помимо описания переменных и основной программы, модуль формы может содержать описание процедур-обработчиков событий, связанных с формой. В первую очередь это событие, возникающее при создании формы на сервере. Последовательность событий, вызывающихся при создании формы уже существующего объекта информационной базы, а также открытии и закрытии окна формы, представлена на рис. 3.11.

Форма полностью подготавливается к открытию на сервере и затем открывается на клиенте. При открытии формы на стороне сервера вызываются два события - ПриЧтенииНаСервере и ПриСозданииНаСервере.

Первое событие формы ПриЧтенииНаСервере вызывается только для существующих в информационной базе объектов. Причем поставляется это событие не самой формой, а ее расширением, определяемым типом основного реквизита формы. В обработчике этого события, в параметре ТекущийОбъект разработчику доступен прикладной объект, содержащийся в форме, со всей его функциональностью. То есть здесь можно вызвать экспортируемые методы этого объекта или получить значения его реквизитов, не конвертируя данные формы в объект. Таким образом, в обработчике этого события разработчик может подготовить данные формы, зависящие от данных объекта, к открытию.

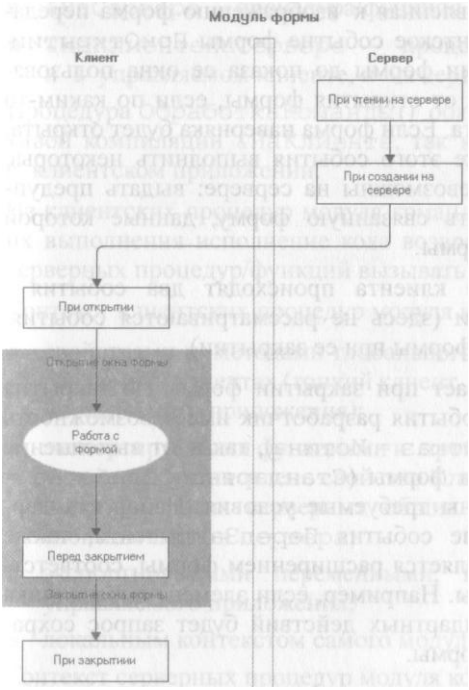


Рис. 3.11. Последовательность вызова событий модуля формы

Второе событие формы `ПриСозданииНаСервере` вызывается всегда при открытии форм и новых, и существующих объектов. Здесь уже недоступен прикладной объект, отображаемый в форме. В обработчике этого события разработчик может полностью подготовить форму, ее внешнее представление, к открытию, в зависимости от различных условий настроить ее интерфейсные свойства.

В обработчике этого события разработчик имеет возможность отказаться как от открытия формы `(Отказ = Истина)`, так и от выполнения стандартных действий при открытии формы `(СтандартнаяОбработка = Ложь)`, если, например, не выполнены требуемые условия. Набор стандартных действий различен для разных форм и определяется расширением формы, соответствующим основному реквизиту формы. Например, для формы списка стандартная обработка заключается в передаче параметров, указанных при открытии формы, в динамический список, а при открытии формы отчета при выполнении стандартной обработки в форму загружается вариант и пользовательские настройки отчета.

После этого уже полностью подготовленная к изображению форма передается на клиента, и вызывается клиентское событие формы ПриОткрытии. Это событие возникает при открытии формы до показа ее окна пользователю. Здесь также можно отказаться от открытия формы, если по каким-то причинам она не должна быть открыта. Если форма наверняка будет открыта, то разработчик может в обработчике этого события выполнить некоторые интерактивные действия, которые невозможны на сервере: выдать предупреждение пользователю или открыть связанную форму, данные которой зависят от основной открываемой формы.

При закрытии формы на стороне клиента происходят два события - ПередЗакрытием и ПриЗакрытии (здесь не рассматриваются события, возникающие при записи объекта из формы при ее закрытии).

Событие ПередЗакрытием возникает при закрытии формы до закрытия окна формы. В обработчике этого события разработчик имеет возможность отказаться как от закрытия формы (Отказ = Истина), так и от выполнения стандартных действий при закрытии формы (СтандартнаяОбработка = Ложь), если, например, не выполнены требуемые условия. Набор стандартных действий, выполняемых после события ПередЗакрытием, также различен для разных форм и определяется расширением формы, соответствующим основному реквизиту формы. Например, если элемент справочника был модифицирован, одним из стандартных действий будет запрос сохранения изменений перед закрытием формы.

Событие ПриЗакрытии возникает при закрытии формы после закрытия окна формы. В обработчике этого события можно описывать алгоритмы, которые должны быть выполнены только в случае, когда форма будет наверняка закрыта. Например, если нужно закрыть вспомогательную форму, существующую только в случае, если открыта основная форма.

Модуль команды

Для прикладных объектов конфигурации существуют подчиненные объекты Команды. Существуют также общие объекты конфигурации — объекты Общая команда. У каждой команды существует модуль команды, в котором можно написать предопределенную процедуру ОбработкаКомандыО, в которой на встроенном языке описываются действия, которые должна выполнить система при вызове этой команды.

Модуль команды, так же как и модуль управляемой формы, существует и на сервере, и на клиенте. В модуле команды могут быть использованы следующие директивы компиляции:

- &НаКлиенте - процедура/функция исполняется в управляемом клиенте;

- &НаСервере - процедура/функция исполняется на сервере;
- &НаКлиентеНаСервере - процедура/функция может исполняться и в управляемом клиенте, и на сервере.

Процедура ОбработкаКомандыО обязательно должна предваряться директивой компиляции &НаКлиенте, так как выполнение команды происходит в клиентском приложении.

Из клиентских процедур модуля команды можно вызывать серверные, после их выполнения исполнение кода возвращается на клиента. Но при этом из серверных процедур/функций вызывать клиентские процедуры нельзя.

Контекст клиентских процедур модуля команды (рис. 3.12) образуется:

- свойствами и методами глобального контекста, которые доступны в управляемых клиентах (тонкий клиент, веб-клиент, толстый клиент в режиме управляемого приложения);
- экспортируемыми функциями и процедурами клиентских общих модулей, у которых установлено свойство Клиент (управляемое приложение), и неглобальных серверных общих модулей, для которых установлено свойство Вызов сервера;
- экспортируемыми переменными, процедурами и функциями модуля управляемого приложения;
- локальным контекстом самого модуля команды.

Контекст серверных процедур модуля команды (рис. 3.12) образуется:

- свойствами и методами глобального контекста;
- экспортируемыми функциями и процедурами общих модулей, скомпилированных на сервере (для этих модулей установлено свойство Сервер);
- серверными методами модуля команды.

Модуль команды должен содержать только описание процедур и функций и не может иметь переменных и тела модуля.

Хотя в модуле команды можно описывать методы с ключевым словом Экспорт, из встроенного языка невозможно получить доступ к командам и, следовательно, к их контексту тоже.

Работа с метаданными

Встроенный язык «1С:Предприятия» позволяет работать с метаданными. Одним из концептуальных моментов этой работы является то, что доступ к метаданным предоставляется только на чтение. Таким образом, средствами встроенного языка разработчик не может модифицировать метаданные. Изменение структуры метаданных возможно только средствами визуального конструирования в режиме Конфигуратор.

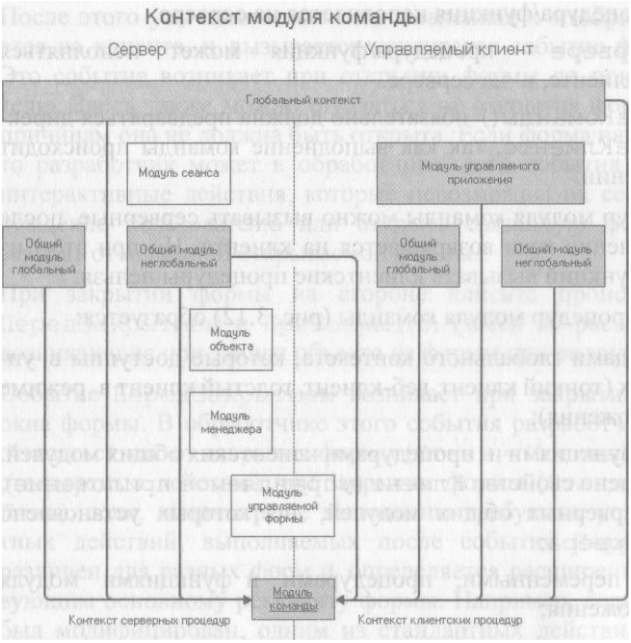


Рис. 3.12. Контекст модуля команды

Для доступа к метаданным из встроенного языка используется объект `ОбъектМетаданныхКонфигурация`, который доступен через свойство глобального контекста `Метаданные`. Этот объект предоставляет доступ как к свойствам самой конфигурации, так и к отдельным коллекциям объектов метаданных, которые описывают различные виды прикладных объектов, определенных в прикладном решении. Например, для доступа к коллекции объектов метаданных, описывающих документы, можно использовать следующий код (листинг 3.9).

Листинг 3.9. Доступ к коллекции метаданных

```
МетаданныеДокументов = Метаданные.Документы;
```

А для того, чтобы получить доступ к объектам конфигурации, описывающим справочники, можно использовать такой вызов (листинг 3.10).

Листинг 3.10. Доступ к коллекции метаданных

```
МетаданныеСправочников = Метаданные.Справочники;
```

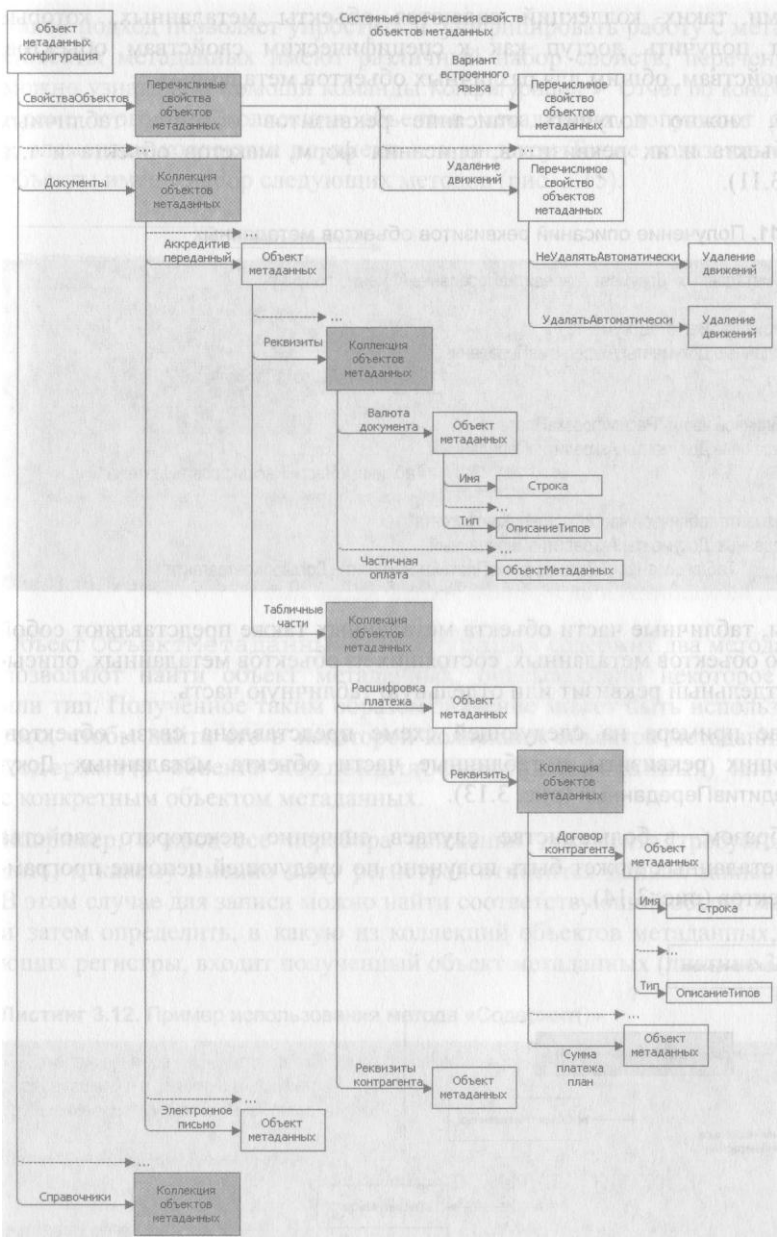


Рис. 3.13. Описание реквизитов и табличных частей документа

Элементами таких коллекций являются объекты метаданных, которые позволяют получить доступ как к специфическим свойствам объектов, так и к свойствам, общим для различных объектов метаданных.

Например, можно получить описание реквизитов объекта, табличных частей объекта и их реквизитов, описания форм, макетов объекта и т.д. (листинг 3.11).

Листинг 3.11. Получение описаний реквизитов объектов метаданных

```
// Для объекта метаданных "Документ.АккредитивПереданный" можно получить:

// описание реквизита "ВалютаДокумента"
Объект = Метаданные.Документы.АккредитивПереданный.
                                                    Реквизиты.ВалютаДокумента;

// описание табличной части "РасшифровкаПлатежа"
Объект =
Метаданные.Документы.АккредитивПереданный.
                                                    ТабличныеЧасти.РасшифровкаПлатежа;

// описание реквизита табличной части
"ДоговорКонтрагента" Объект =
МетаданныеДокументы.АккредитивПереданный.
                    ТабличныеЧасти.РасшифровкаПлатежа.Реквизиты.ДоговорКонтрагента;
```

Реквизиты, табличные части объекта метаданных также представляют собой коллекцию объектов метаданных, состоящих из объектов метаданных, описывающих отдельный реквизит или отдельную табличную часть.

В качестве примера на следующей схеме представлена связь объектов, описывающих реквизиты и табличные части объекта метаданных Документ.АккредитивПереданный (рис. 3.13).

Таким образом, в большинстве случаев значение некоторого свойства объекта метаданных может быть получено по следующей цепочке программных объектов (рис. 3.14).

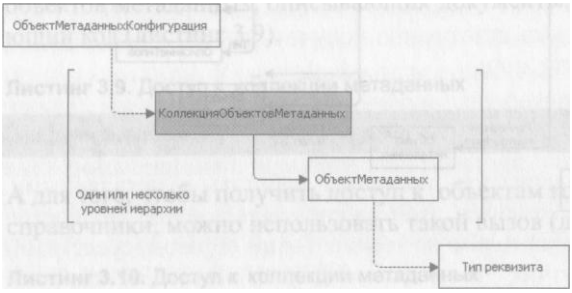


Рис. 3.14. Цепочка объектов описания метаданных

Такой подход позволяет упростить и унифицировать работу с метаданными. Объекты метаданных имеют различный набор свойств, перечень которых можно узнать при помощи команды Конфигурация ► Отчет по конфигурации... Кроме этого, все коллекции объектов метаданных допускают обращение к элементам коллекции по имени или индексу. Также показанные на схеме объекты имеют набор следующих методов (рис. 3.15).

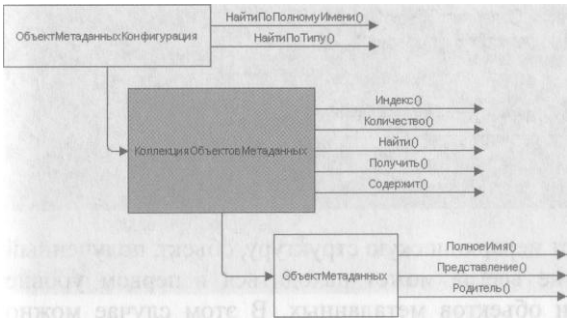


Рис. 3.15. Методы объектов описания метаданных

Объект `ОбъектМетаданныхКонфигурация` содержит два метода, которые позволяют найти объект метаданных, описывающий некоторое значение или тип. Полученное таким образом значение может быть использовано для того, чтобы найти его в некоторой коллекции объектов метаданных (метод `Содержит ()` объекта `КоллекцияОбъектовМетаданных`) или сравнить с конкретным объектом метаданных. Например, в процессе перебора движений документа требуется определить, к какому именно виду регистров относится данная запись регистра. В этом случае для записи можно найти соответствующий объект метаданных и затем определить, в какую из коллекций объектов метаданных, описывающих регистры, входит полученный объект метаданных (листинг 3.12).

Листинг 3.12. Пример использования метода «Содержит()»

```
//В качестве примера выбирается первая запись из первого набора записей,
// содержащегося в движениях документа Сообщение = Новый
СообщениеПользователю;

Движения = Документы .АвансовыйОтчет.
                                        НайтиПоНомеруГТК000009", '20110101000000').
                                        ПолучитьОбъект().Движения[0];
Движения.Прочитать();



ОбъектМетаданных = Метаданные.НайтиПоТипу(ТипЗнч{Движения[0]));
```

```
Если Метаданные.РегистрыНакопления.Содержит{0бъектМетаданных) Тогда
        Сообщение.Текст = " Это движения по регистру накопления";
        Сообщение,Сообщить();

ИначеЕсли Метаданные.РегистрыСведений.Содержит(ОбъектМетаданных) Тогда
        Сообщение.Текст =" Это движения по регистру сведений";
        Сообщение.Сообщить();

ИначеЕсли Метаданные.РегистрыБухгалтерии.Содержит(ОбъектМетаданных) Тогда
        Сообщение.Текст = 0 Это движения по регистру бухгалтерии";
        Сообщение.Сообщить();

ИначеЕсли Метаданные.РегистрыРасчета.Содержит(ОбъектМетаданных) Тогда
Сообщение.Текст =" Это движения по регистру расчета"; Сообщение.Сообщить();
КонецЕсли;
```

Поскольку метаданные имеют иерархическую структуру, объект, полученный методом `НайтиПоТипуО`, не всегда может находиться в первом уровне иерархии нужной коллекции объектов метаданных. В этом случае можно использовать метод `Родитель ()` объекта метаданных для того, чтобы получить объект, находящийся на нужном уровне иерархии.

Например, если некоторая универсальная процедура обрабатывает табличные части справочников по одному алгоритму, а табличные части документов — по другому, то проанализировать принадлежность табличной части определенному виду метаданных можно следующим образом (листинг 3.13).

Листинг 3.13. Пример использования метода «Родитель()»

```
IIВ качестве примера получается табличная часть "Товары"
II документа "АвансовыйОтчет"
Сообщение = Новый СообщениеПользователю;

ТабличнаяЧасть = Документы. АвансовыйОтчет,
                                        НайтиПоНомеру("ТК000009", '20110101000000').
                                        ПолучитьОбъект().Товары;
ОбъектМетаданных = Метаданные.
                                        НайтиЛоТипу(ТилЗнч(ТабличнаяЧасть)).Родитель{};

Если Метаданные.Справочники.Содержит(ОбъектМетаданных) Тогда
        Сообщение.Текст = "Это табличная часть справочника";
        Сообщение.Сообщить();

ИначеЕсли Метаданные.Документы.Содержит(ОбъектМетаданных) Тогда
Сообщение.Текст = "Это табличная часть документа";
Сообщение.Сообщить();
КонецЕсли;
```

Как уже говорилось выше, коллекция объектов метаданных позволяет обращаться к объектам по имени или индексу. Кроме этого, коллекция поддерживает итератор `Для Каждого...`, с помощью которого, например, может быть организован перебор всех элементов справочников, содержащихся в конфигурации (листинг 3.14).

Листинг 3.14. Использование итератора «Для Каждого»

```
Сообщение = Новый СообщениеПользователю;
Для Каждого Справочник из Метаданные.Справочники Цикл
        Сообщение.Текст = Симвояы.ПС + Справочник.ПолноеИмя():
        Сообщение.Сообщить();
        МенеджерСправочника = Справочники[СПравочник.Имя];
        Выборка = МенеджерСпраеочника.Выбрать(); Пока
        Выборка.СледующийО Цикл
                Сообщение.Текст =
Выборка.Наименование; Сообщение Сообщить!):
КонецЦикла; КонецЦикла;
```

Глава 4. Работа сданными

Объектные и необъектные данные

Все данные, которые хранятся в базе данных «1С:Предприятия», можно разделить на две категории: объектные и необъектные. Поскольку природа этих данных различна, различаются и способы работы с ними.

К объектным данным относятся данные справочников, документов, планов видов характеристик, планов счетов, планов видов расчета, бизнес-процессов, задач, планов обмена.

К необъектным данным относятся данные регистров сведений, регистров накопления, регистров расчета, перерасчетов, регистров бухгалтерии и последовательностей. Также к необъектным данным относятся константы.

С точки зрения системы объектные данные состоят из отдельных объектов. Каждый из этих объектов обладает внутренним уникальным идентификатором, благодаря наличию которого к некоторой совокупности значений, хранящихся в базе данных, можно обращаться как к единому целому - объекту. Например, объектом является элемент справочника или документ. Каждый объект, помимо того что он является совокупностью значений некоторых полей, имеет также определенную значимость сам по себе. Например, элемент справочника ФизическиеЛица - это некое физическое лицо, которое имеет набор характеризующих его значений: имя, фамилия, отчество,

паспортные данные и т.д. У него может поменяться, например, фамилия или паспортные данные, но при этом с точки зрения системы он останется тем же самым физическим лицом - объектом. Удаление какого-либо объекта из системы приводит к тому, что состояние базы данных с точки зрения прикладного решения изменяется. Даже создав новый элемент справочника ФизическиеЛица с теми же самыми значениями реквизитов, мы получим уже другое состояние базы данных, поскольку это будет уже другой объект с другим уникальным идентификатором.

В отличие от объектных данных, необъектные данные не имеют собственной ценности и полностью описываются значениями своих полей. Необъектные данные представляют собой записи, которые хранятся в базе данных. Для записей не поддерживаются внутренние уникальные идентификаторы. Удалив некоторую запись и создав новую с точно такими же значениями полей, мы получим то же самое состояние базы данных, которое было до удаления записи.

Рассмотрим работу с объектными и необъектными данными более подробно.

Объектные данные

Модель хранения

данных

К объектным данным в «1С:11редприятии» относятся данные следующих объектов конфигурации:

- Справочник,
- Документ,
- План видов характеристик,
- План счетов,
- План видов расчета,
- План обмена,
- Бизнес-процесс,
- Задача.

Для каждой объектной сущности конфигурации система создает набор связанных между собой таблиц, в которых и будут храниться данные этого объекта. Количество и состав таблиц различны для каждого объекта метаданных.

Структура хранения объектных данных всегда состоит из основной таблицы и, возможно, нескольких других таблиц (по одной таблице на каждую табличную часть объекта). Например, для документа Доверенность, который

имеет некоторый набор реквизитов и табличную часть Товары (рис. 4.1), в базе данных будут созданы две таблицы.

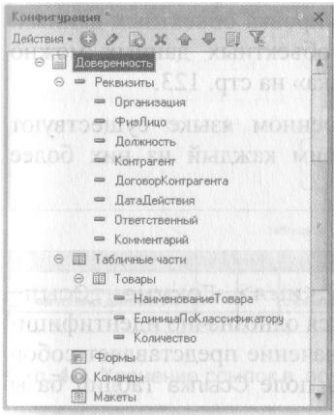


Рис. 4.1. Документ «Доверенность»

Таблица документа будет содержать поля для каждого реквизита документа, а для табличной части будет создана отдельная таблица, содержащая поля для всех реквизитов табличной части документа (рис. 4.2).

Таблица документа "Доверенность"

Ссылка	Дата	Номер	Организация	ФизЛицо	...
...	...	...	...	...	...
0x8D3650505450303011DA147DCE8DB6E9	10.04.2011 12:00:00,000	ТК000003	Торговый дом "Комплексный"	Бахшиев П. И.	...
...	...	...	...	...	...

Табличная часть "Товары" документа "Доверенность"

Ссылка	Номер строки	Наименование товара	Единица по классификатору	Количество
...	...	...	...	...
0x8D3650505450303011DA147DCE8DB6E9	1	Ассорти (конфеты)	упак	10
0x8D3650505450303011DA147DCE8DB6E9	2	Барбарис (конфеты)	кг	10
0x8D3650505450303011DA147DCE8DB6E9	3	Белочка (конфеты)	кг	10
0x8D3650505450303011DA147DCE8DB6E9	4	Грильяж (конфеты)	кг	10
...	...	...	...	...

Рис. 4.2. Таблицы документа «Доверенность»

Отличительной особенностью этих таблиц является то, что каждая из них содержит поле Ссылка, в котором хранится внутренний идентификатор, соответствующий каждому из документов. Таким образом, объект документа представляет собой совокупность записи основной таблицы и строк табличных частей, относящихся к этому документу.

Основная таблица объектных данных также содержит обязательное поле, в котором хранится текущая версия объекта. Значение этого поля изменяется при каждой записи данных объекта в базу данных. Благодаря использованию этого поля обеспечивается оптимистическая блокировка объектных данных. Подробнее про оптимистическую блокировку объектных данных можно прочитать в разделе «Оптимистическая блокировка» на стр. 123.

Для работы с объектными данными во встроенном языке существуют два основных типа: ссылка и объект. Рассмотрим каждый из них более подробно.

Ссылка

Значение ссылочного типа (`СправочникСсылка.<имя>`, `ДокументСсылка.<имя>` и т.д.) используется везде, где требуется однозначно идентифицировать объект базы данных. Фактически такое значение представляет собой внутренний идентификатор, который хранится в поле Ссылка таблиц базы данных.

Например, справочник Валюты хранится в базе данных в таблице, которая, помимо служебных колонок (стандартных реквизитов), содержит отдельные колонки для каждого реквизита справочника, заданного в конфигураторе. Поле Ссылка - это одно из служебных полей. Значение этого поля позволяет однозначно отличить один элемент справочника от другого (проще говоря, одну валюту от другой), рис. 4.3.

Ссылка	Пометка удаления	Предопределенный	Код	Наименован...	Наименование полное
0x848A00112F43529A11D955BCBD72D8F9	Ложь	Ложь	810	руб.	Российский рубль
0x848A00112F43529A11D955BCBD72D8FA	Ложь	Ложь	840	USD	Доллар США
0x848A00112F43529A11D955BCCBCF4923	Ложь	Ложь	978	EUR	ЕВРО
0x8486.00112F43529A11D955BD0E617G14	Ложь	Ложь	792	TRL	Турецкая лира
0x848A00112F43529A11D955BD0E617G15	Ложь	Ложь	001	У-е.	Условная единица

Рис. 4.3. Поле «Ссылка» справочника «Валюты»

Значение ссылки может, например, выбираться в полях ввода, храниться в полях других таблиц базы данных и т.д. Например, поля Организация и ФизЛицо документа Доверенность будут хранить ссылки на элементы справочников Организации и ФизическиеЛица (рис. 4.4).

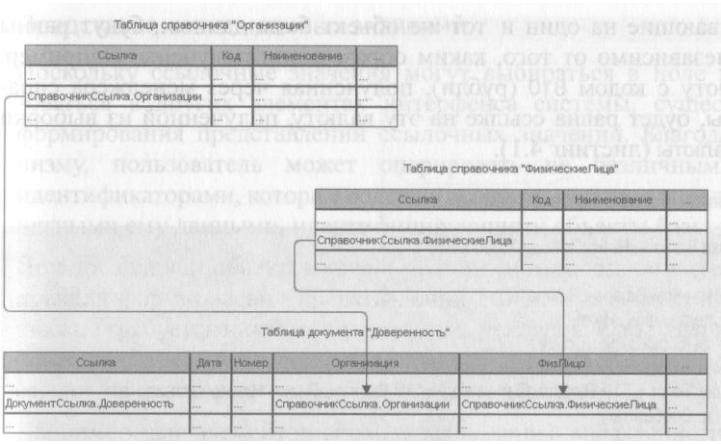


Рис. 4.4. Хранение ссылок в полях базы данных

Значения ссылочного типа можно сравнивать между собой. Важным моментом является то, что для каждого объекта конфигурации во встроенном языке создается свой тип ссылки. Таким образом, например, ссылка на справочник Организации никогда не будет равна ссылке на справочник Физи- ческиеЛица, поскольку это значения разных типов. Однако две ссылки на справочник Организации могут быть равны между собой, и это будет выполняться только в том случае, если это ссылки на один и тот же объект базы данных (рис. 4.5).

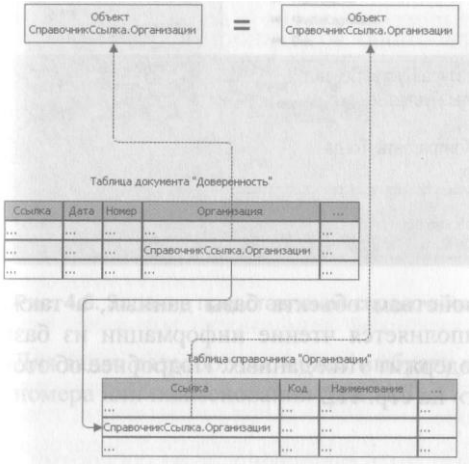


Рис. 4.5. Сравнение объектов «Ссылка»

Ссылки, указывающие на один и тот же объект базы данных, будут равны между собой независимо от того, каким образом они получены. Например, ссылка на валюту с кодом 810 (рубли), полученная через менеджера справочника Валюты, будет равна ссылке на эту валюту, полученной из выборки справочника Валюты (листинг 4.1).

Листинг 4.1. Сравнение объектов «Ссылка»

```
Ссылка1 = Справочники.Валюты.НайтиПоКоду("000000810");

ВыборкаВалют = Справочники.Валюты.Выбрать();
Пока ВыборкаВалют.Следующий() Цикл
        Если Ссылка1 = ВыборкаВалют.Ссылка Тогда
                Сообщение = Новый СообщениеПользователю;
                Сообщение.Текст ="" + ВыборкаВалют.Код +"" + ВыборкаВалют.Наименование;
Сообщение.Сообщить(); КонецЕсли; КонецЦикла;
```

Типы ссылок имеют значение по умолчанию - так называемую пустую ссылку. Пустая ссылка - это значение ссылки, которому не соответствует ни один объект в базе данных. Фактически такой внутренний идентификатор выглядит следующим образом:
00000000-0000-0000-0000-000000000000

Так как тип ссылки создается для каждого объекта конфигурации, то, например, пустые ссылки на разные справочники никогда не будут равны между собой (листинг 4.2).

Листинг 4.2. Сравнение пустых ссылок

```
ПустаяСсылкаНоменклатура -- Спразочники.Номенклатура.ПустаяСсылка();
ПустаяСсылкаКонтрагенты = Справочники.Контрагенты.ПустаяСсылка();

Если ПустаяСсылкаНоменклатура <> ПустаяСсылкаКонтрагенты Тогда
Сообщение = Новый СообщениеПользователю; Сообщение.Текст =
"Ссылки не равны"; Сообщение.Сообщить(); КонецЕсли;
```

Ссылка позволяет обращаться к свойствам объекта базы данных, а также получать сам объект. При этом выполняется чтение информации из базы данных, поскольку сама ссылка не содержит этих данных. Подробнее об этом можно прочитать в разделе «Объект» на стр. 112.

Представление ссылочных значений

Поскольку ссылочные значения могут выбираться в поле ввода и использоваться в других элементах интерфейса системы, существует механизм формирования представлений ссылочных значений. Благодаря этому механизму, пользователь может оперировать не безличными внутренними идентификаторами, которые содержит ссылка, а вполне определенными и понятными ему данными, идентифицирующими объекты базы данных.

При добавлении объектов конфигурации система самостоятельно определяет правила формирования представлений ссылочных значений, и от разработчика не требуется никаких специальных действий. Однако при необходимости он может внести изменения в правила формирования представлений для ссылок на некоторые типы объектов базы данных.

Например, для элементов справочников, видов характеристик, счетов, видов расчетов и узлов планов обмена представление ссылок может быть задано либо в виде кода, либо в виде наименования (рис. 4.6).

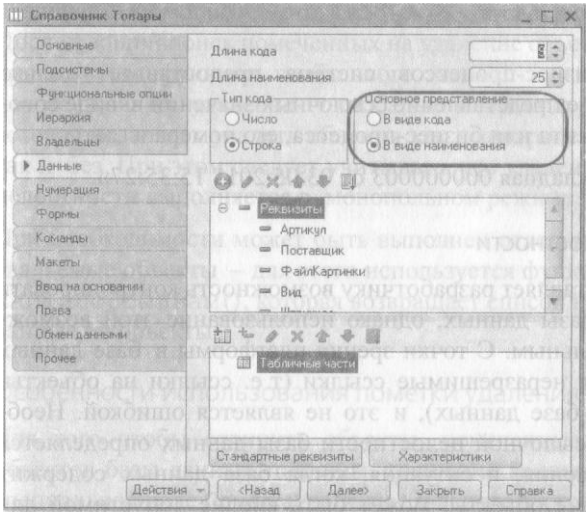


Рис. 4.6. Задание представления справочника

Для задач разработчик может выбрать между представлением ссылок в виде номера или наименования (рис. 4.7).

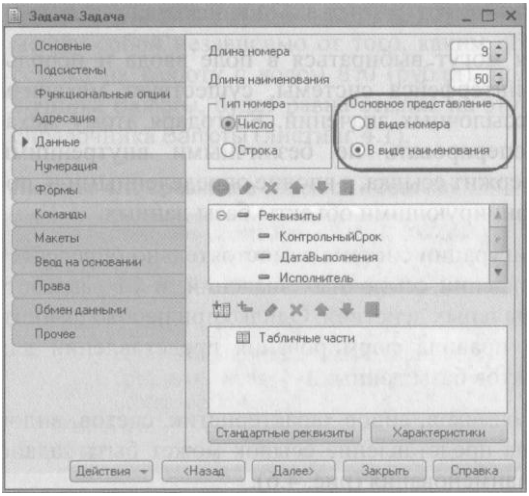


Рис. 4.7. Задание представления задачи

А для документов и бизнес-процессов система предоставляет только единственное неизменяемое представление ссылочных значений в виде совокупности синонима документа или бизнес-процесса, его номера и даты.

Например: «Приходная накладная 000000003 от 03.05.2011 15:35:27». Контроль

ссылочной целостности

«1С:Предприятие» предоставляет разработчику возможность контролировать ссылочную целостность базы данных, однако использование этой возможности не является обязательным. С точки зрения платформы в базе данных вполне могут содержаться неразрешимые ссылки (т.е. ссылки на объекты, которых не существует в базе данных), и это не является ошибкой. Необходимость поддержания ссылочной целостности базы данных определяется логикой прикладного решения, и ситуация, когда база данных содержит ссылки на несуществующие объекты, может быть вполне допустимой для данного прикладного решения.

Вопрос о необходимости контроля ссылочной целостности возникает при удалении данных из базы данных. Система позволяет гибко использовать возможности контроля ссылочной целостности как при интерактивном, так и при программном удалении объектов.

Интерактивное удаление объектов может быть выполнено двумя способами.

Во-первых, объект может быть удален непосредственно из базы данных. В этом случае не будет выполняться никаких проверок ссылочной целостности и последующее восстановление объекта будет невозможно. Возможность непосредственного интерактивного удаления регулируется правом `Интерактивное удаление`.

Во-вторых, интерактивное удаление может быть выполнено с использованием механизма контроля ссылочной целостности. В этом случае удаление выполняется в два этапа: сначала пользователь устанавливает пометку удаления для тех объектов, которые он собирается удалить, а затем выполняет процедуру удаления помеченных объектов. При этом сначала будет выполнен поиск ссылок на удаляемые объекты в других данных, а затем для объектов, ссылки на которые отсутствуют, будет выполнено удаление.

При удалении объектов средствами встроенного языка также существует возможность использовать или не использовать контроль ссылочной целостности. Непосредственное удаление объектов может быть выполнено методом объекта `Удалить`(), в то время как пометка на удаление может быть установлена с помощью метода объекта `УстановитьПометкуУдаления()`. Последующий поиск помеченных на удаление объектов может быть выполнен при помощи функции глобального контекста `НайтиПомеченныеНаУда- ление()`, а их удаление - глобальной процедурой `УдалитьОбъектыО`, которая позволяет удалять объекты как с контролем ссылочной целостности, так и без. При этом следует учитывать, что удаление с контролем ссылочной целостности выполняется в монопольном режиме.

При необходимости может быть выполнен отдельно только поиск ссылок на удаляемые объекты - для этого используется функция глобального контекста `НайтиПоСсылкам` (), которая возвращает список найденных ссылок на переданные ей объекты.

Особенности использования пометки удаления

Для того чтобы отличить объекты, помеченные на удаление, от других объектов базы данных, таблицы, хранящие данные этих объектов, содержат специальное поле ПометкаУдаления. Значение этого поля может быть установлено двумя способами.

Во-первых, может использоваться интерактивная установка пометки удаления или программное выполнение метода объекта `УстановитьПометкуУда- ления()`. В этом случае кроме собственно установки значения этого поля будет выполнен ряд дополнительных действий, состав которых зависит от типа помечаемого объекта. Например, для справочника будет установлена пометка удаления для всех подчиненных элементов этого справочника и подчиненных справочников, для документа будет выполнена отмена проведения и т.д.

Также будет вызвано событие объекта ПередЗаписьюО, поскольку будет выполняться сохранение измененного свойства ПометкаУдаления этого объекта.

Во-вторых, значение поля ПометкаУдаления может быть установлено путем непосредственной установки свойства ПометкаУдаления объекта и последующей его записи. В этом случае никаких дополнительных действий выполняться не будет, однако следует учитывать особенности работы некоторых объектов. Например, документ не может быть одновременно проведен и помечен на удаление, поэтому, если требуется пометить на удаление проведенный документ, предварительно следует установить его свойство Проведен в значение Ложь или выполнить отмену проведения документа (Записать (РежимЗаписиДокумента.ОтменаПроведения)).

Объект

Тип объекта служит, прежде всего, для модификации (чтения и изменения) данных, содержащихся в объекте базы данных. Остальные объекты встроенного языка позволяют только читать информацию базы данных.

Как уже говорилось ранее, объект представляет собой совокупность записи основной таблицы и строк табличных частей, относящихся к этому объекту. Например, рассмотрим документ ЗаказПоставщику, имеющий три табличные части (рис. 4.8).

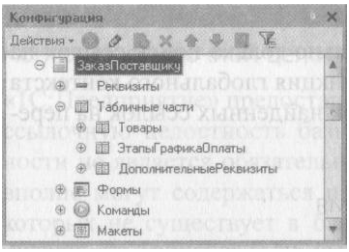


Рис. 4.8. Документ «ЗаказПоставщику»

Объект документа (ДокументОбъект.ЗаказПоставщику) будет представлять собой совокупность значений полей основной таблицы документа и полей каждой из таблиц, в которых хранятся данные его табличных частей (рис. 4.9).

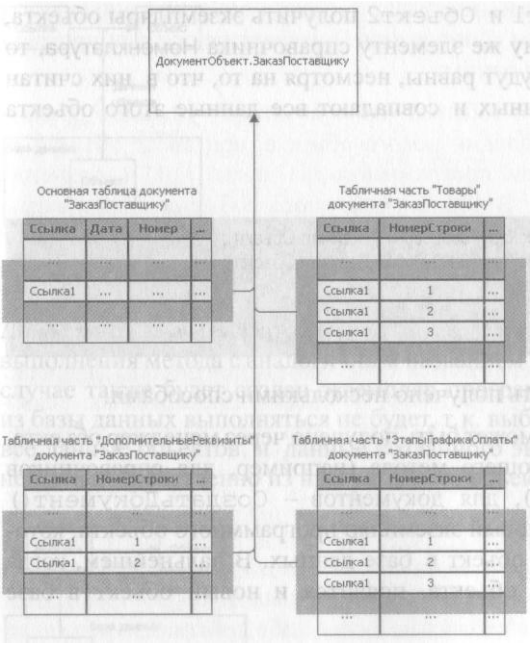


Рис. 4.9. Данные объекта документа

Тип объекта используется при создании новых объектов для редактирования и удаления существующих объектов. Кроме этого, тип объекта используется для отображения и редактирования всех данных объекта в форме объекта.

Значения этого типа, так же как и значения ссылок, можно сравнивать между собой. Однако, в отличие от ссылок, значения этого типа будут равны между собой только тогда, когда они являются одним и тем же экземпляром программного объекта.

Например, если в следующем примере (листинг 4.3) в переменную `Объект1` получить экземпляр программного объекта, соответствующий элементу справочника Номенклатура и затем значение этой переменной присвоить переменной `Объект2`, то значения этих переменных будут равны.

Листинг 4.3. Сравнение объектов

```
0бъект1 =справочники.номенкпатура.найтапокоду("000000001").получить0бъект();
0бъект2 = 0бъект1;
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст =" 0бъект1 = Объект2";
Сообщение.Сообщить();
```

Если же в переменные `Объект1` и `Объект2` получить экземпляры объекта, соответствующие одному и тому же элементу справочника Номенклатура, то значения этих переменных не будут равны, несмотря на то, что в них считан один и тот же объект базы данных и совпадают все данные этого объекта (листинг 4.4).

Листинг 4.4. Сравнение объектов

```
Объект1 = Справочники.Номенклатура.НайтиПоКоду("000000001
").ПолучитьОбъект(); Объект2 =
Справочники.Номенклатура.НайтиПоКоду("000000001 ").ПолучитьОбъект();
Сообщение = Новый СообщениеПользователю; Сообщение.Текст =" Объект1 <>
Объект2"; Сообщение.Сообщить();
```

Значение типа объект может быть получено несколькими способами.

Во-первых, значение этого типа может быть получено через менеджер объекта, с использованием соответствующего метода (например, для справочников это метод `СоздатьЭлемент()`, для документов — `СоздатьДокумент()` и т.д.). При этом будет создан новый экземпляр программного объекта, которому не соответствует ни один объект в базе данных. В дальнейшем, когда будет выполнена запись этого объекта, появится и новый объект в базе данных (рис. 4.10).

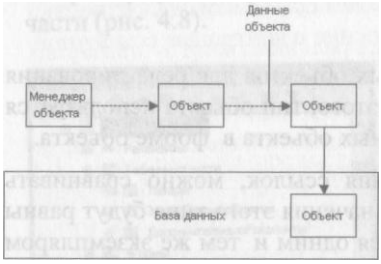


Рис. 4.10. Создание нового объекта

Во-вторых, значение этого типа может быть получено из ссылки путем выполнения метода `ПолучитьОбъект()`. В этом случае будет создан экземпляр программного объекта и выполнено чтение данных из базы данных (или из кеша). При этом считываются значения всех реквизитов объекта и значения всех реквизитов всех его табличных частей (рис. 4.11).



Рис. 4.11. Получение объекта

В-третьих, значение этого типа может быть получено из выборки путем выполнения метода с аналогичным названием - `ПолучитьОбъект` (). В этом случае также будет создан экземпляр программного объекта, однако чтение из базы данных выполняться не будет, т. к. выборка считывает из базы данных все данные объектов, и данные созданного экземпляра объекта будут заполнены непосредственно из программного объекта выборки (рис. 4.12).



Рис. 4.12. Получение объекта

В связи с тем, что тип объекта позволяет модифицировать данные, он имеет (помимо свойств и методов) целый ряд событий, позволяющих разработчику определять собственный алгоритм обработки выполняемых действий. Среди основных событий, поддерживаемых этим типом, можно перечислить следующие.

`ПередЗаписью` - это событие возникает перед записью объекта в базу данных, после начала транзакции записи, но до начала непосредственной записи данных. В обработчике этого события можно проанализировать необходимость (или возможность) выполнения записи данных и отказаться от нее, если какие-либо условия не выполняются.

`ПриЗаписи` - это событие возникает после того, как данные записаны в базу данных, но до того, как окончится транзакция записи. В обработчике этого события можно выполнять те действия, которые обязательно должны быть выполнены только в том случае, если объект записан. Здесь также разработчик может отказаться от записи данных, если, например, в результате записи этих данных в базу нарушаются какие-либо условия.

ПриКопировании - это событие возникает у нового объекта данных в том случае, если он создается путем копирования (интерактивного или программного) существующего объекта данных.

ОбработкаЗаполнения - это событие также возникает у нового объекта данных при интерактивном создании нового объекта, при вводе объекта на основании (интерактивного или программного), а также при выполнении метода объекта Заполнить(). В обработчике этого события предусматривается некоторый алгоритм, позволяющий выполнять начальное заполнение реквизитов нового объекта в зависимости от типа объекта-основания.

ОбработкаПроверкиЗаполнения - это событие вызывается перед записью данных объекта, до начала транзакции записи. В обработчике этого события разработчик может реализовать собственные алгоритмы проверки заполнения реквизитов объекта, добавить в массив проверяемых реквизитов дополнительные реквизиты, чтобы платформа выполнила их проверку, или очистить массив проверяемых реквизитов, отказавшись от их стандартной проверки. Разработчик также может отказаться от записи объекта, если, например, какие-то условия проверки не были выполнены.

ПередУдалением - это событие возникает в транзакции удаления перед непосредственным удалением объекта из базы данных. В обработчике этого события можно предусмотреть выполнение каких-либо действий перед удалением объекта, а также при необходимости отменить удаление объекта, если не выполняются какие-либо условия.

Кеширование представлений и объектов

При работе с объектными данными (как программно, так и интерактивно), система выполняет кеширование считываемых данных в оперативной памяти. Для этого используется кеш объектов. При любых интерактивных действиях и при программном доступе к объектным данным с использованием объектной модели прежде всего система будет выполнять обращение к кешу объектов для того, чтобы получить запрашиваемые данные.

Кеш объектов состоит из двух частей: транзакционного кеша и обычного кеша. В зависимости от того, происходит ли чтение в рамках транзакции или нет, будет выполняться обращение к той или иной части кеша (рис. 4.13).

В кеш объектов считываются два «вида» данных: либо все данные объекта целиком, либо значения полей, необходимые для формирования представления ссылки на данный объект.

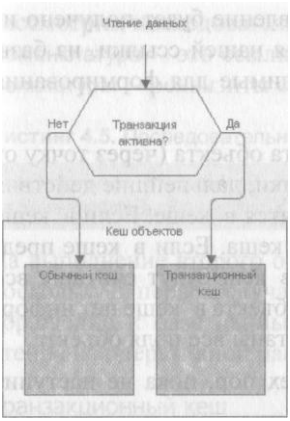


Рис. 4.13. Обращение к кешу объектов

Значения полей, необходимые для получения представления ссылки, считы- ваются тогда, когда возникает необходимость отобразить ссылку на объект в каком-либо элементе интерфейса, а также при явном или неявном преобразовании ссылочной переменной к типу Строка. В остальных случаях выполняется полное чтение всех данных объекта, в том числе и тогда, когда выполняется обращение к какому-либо реквизиту объекта через точку от ссылки.

Эту особенность нужно учитывать при проектировании структуры объектов конфигурации. Так, например, если предполагается хранение в информационной базе картинок, образов файлов или больших текстовых данных, то рекомендуется создавать для этого отдельные структуры хранения (например, справочники или регистры сведений), а не включать эту информацию в состав реквизитов или табличных частей объектов, которым эта информация соответствует. Это позволит избежать считывания и записи больших объемов данных при работе с этими объектами.

Обычный кеш

Если при обращении к обычному кешу требуемых данных в нем нет, то выполняется чтение данных объекта из базы данных и сохранение их в кеше. Уникальным идентификатором для кеша в данном случае будет являться ссылка на объект базы данных. Поэтому данные каждого считанного объекта могут существовать в кеше в одном из двух видов: либо все данные объекта, либо представление объекта.

Таким образом, если мы обратимся к кешу для получения представления объекта и в кеше есть информация для нашей ссылки, данные будут взяты

из кеша (если в кеше весь объект, нужное представление будет получено из данных объекта). Если в кеше нет информации для нашей ссылки, из базы данных в кеш будут считаны только поля, необходимые для формирования представления объекта.

Если мы обратимся к кешу для получения реквизита объекта (через точку от ссылки) и в кеше есть информация для нашей ссылки, дальнейшие действия будут зависеть от того, какие именно данные находятся в кеше. Если в кеше весь объект, значение реквизита будет получено из кеша. Если в кеше представление объекта, оно будет удалено из кеша и в кеш будут считаны все данные объекта. Если же при получении реквизита объекта в кеше нет информации для нашей ссылки, из базы данных будут считаны все поля объекта.

Считанные данные будут находиться в кеше до тех пор, пока не наступит одно из четырех событий:

- считанные данные будут вытеснены из кеша другими считанными данными других объектов (переполнение кеша);

ш при очередном обращении к кешу окажется, что считанные данные были изменены в базе данных;

- закончится интервал времени в 20 минут;
- данные будут изменены в базе данных.

Все считанные данные помещаются в последовательную очередь, и, поскольку объем кеша ограничен, наиболее старые данные будут вытесняться из кеша последними считанными данными.

При повторном обращении к кешу за данными уже считанного объекта будет анализироваться интервал времени, прошедший с момента появления данных в кеше.

Если обращение происходит в пределах 20 секунд после поступления данных в кеш, данные считаются верными (валидными). Если интервал превысил 20 секунд, будет выполняться проверка на то, что версия данных, хранящихся в кеше, соответствует версии данных, находящихся в базе данных. Если окажется, что версии данных не совпадают (т. е. произошло изменение данных в базе данных), данные, находящиеся в кеше, будут удалены из него и выполнено повторное считывание данных из базы данных. Начиная с этого момента, начнется отсчет следующего 20-секундного интервала валидности этих данных.

Кроме всех вышеперечисленных событий, считанные данные будут удалены из кеша по истечении 20 минут после их последнего считывания из базы данных.

Рассмотрим последовательное выполнение двух операторов (листинг 4.5), где Номенклатура - это ссылка на объект справочника, а Наименование и ВидНо- менкпатуры - реквизиты справочника Номенклатура.

Листинг 4.5. Последовательное обращение к реквизитам объекта

```
А = Номенклатура.Наименование; В
= Номенклатура. ВидНоменклатуры;
```

На выполнение второго оператора будет тратиться гораздо меньше времени, поскольку в первом случае (с большой долей вероятности) будет выполняться обращение к базе данных и чтение всех данных объекта, а во втором - чтение из оперативной памяти (кеша объектов).

Транзакционный кеш

Если обращение к данным происходит в рамках транзакции, то оно переадресуется транзакционному кешу. Транзакционный кеш по сути представляет собой ту же последовательную очередь, что и обычный кеш, за исключением того, что все данные, находящиеся в транзакционном кеше, являются валидными (гарантированно актуальными). При считывании данных в транзакционный кеш устанавливается блокировка на данные в базе данных, поэтому они гарантированно не могут быть изменены до окончания транзакции.

Транзакционный кеш хранит считанные данные до тех пор, пока они не будут вытеснены более поздними считанными данными или пока не закончится транзакция.

По окончании транзакции транзакционный кеш очищается, однако действия, выполняемые при этом, зависят от состояния завершения транзакции.

Если транзакция завершена успешно (Commit), данные всех объектов, содержащиеся в транзакционном кеше, переносятся в обычный кеш, а транзакционный кеш очищается (рис. 4.14).

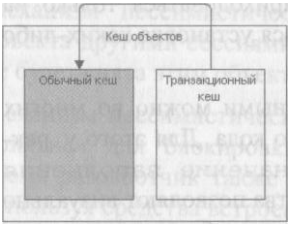


Рис. 4.14. Успешное завершение транзакции

Если был выполнен отказ от изменений (Rollback), то просто очищается транзакционный кеш (рис. 4.15).

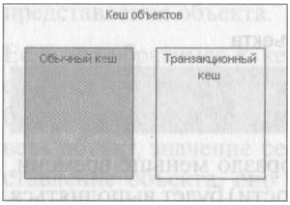


Рис. 4.15. Откат транзакции

Оптимизированная запись объектов

При записи объектов базы данных (программной или интерактивной) выполняется оптимизация записи изменений в базу данных.

Так, например, если не менялись значения реквизитов самого объекта, то будет записана только минимальная информация об изменении (версия объекта).

Если не менялись строки табличной части, то табличная часть записываться не будет.

Если менялись только отдельные строки табличной части или добавлялись новые строки, то будут записаны только измененные и добавленные строки. Однако если менялся порядок строк или строки удалялись, то будут записаны все строки табличной части.

Создание новых объектов

Зачастую при разработке или модификации прикладных решений требуется выполнять некоторые действия, сопровождающие создание тех или иных новых объектов базы данных. Например, при создании новой приходной накладной может потребоваться автоматически заполнять поле Склад, если известно, что все поступающие товары должны приходоваться только на один определенный склад. Также может потребоваться установка каких-либо других реквизитов документа по умолчанию.

Прежде всего, при заполнении нового объекта данными можно во многих случаях обойтись вообще без написания какого-либо кода. Для этого у реквизитов объектов конфигурации есть свойства Значение заполнения и Заполнять из данных заполнения. Эти свойства позволяют визуально (в конфигураторе) задать правила, по которым реквизит будет заполняться данными при создании нового объекта.

Если этих возможностей недостаточно, то нужно использовать возможности встроенного языка.

Действия с данными объекта нужно выполнять в модуле объекта, в обработчике события ОбработкаЗаполнения. Это событие возникает при интерактивном создании нового объекта, при вводе объекта на основании (интерактивном или программном), а также при выполнении метода объекта Заполнить (). В обработчике этого события предусматривается некоторый алгоритм, позволяющий выполнять начальное заполнение реквизитов нового объекта в зависимости от типа объекта-основания.

При этом в параметр ДанныеЗаполнения этого обработчика могут быть переданы различные значения в зависимости от того, каким образом создается новый элемент. Например, он может иметь тип ссылки, если новый объект вводится на основании. Или он может иметь тип Структура, если новый объект создается интерактивной командой из списка, в котором установлен отбор. В этом случае структура будет содержать значения элементов отбора этого списка. Также этот параметр может иметь тип Неопределено, если новый элемент создается интерактивной командой из панели действий. Поэтому в алгоритме начального заполнения нового объекта полезно проанализировать этот параметр и в зависимости от его значения выполнить определенные действия.

Следует учитывать, что при копировании (как интерактивном, так и средствами встроенного языка) событие ОбработкаЗаполнения не вызывается.

Блокировки

При работе с объектными данными система обеспечивает два вида блокировок - пессимистическую и оптимистическую, которые позволяют выполнять целостные изменения объектов при одновременной работе нескольких пользователей.

Пессимистическая блокировка

Механизм пессимистической блокировки запрещает изменение данных объекта другими сессиями или данной сессией до тех пор, пока блокировка не будет снята этим объектом встроенного языка (рис. 4.16).

Механизм пессимистической блокировки используется системой «1С:Предприятие» для блокировки объектов, редактируемых в форме. В то же время разработчик также имеет возможность задействовать этот механизм, используя средства встроенного языка.

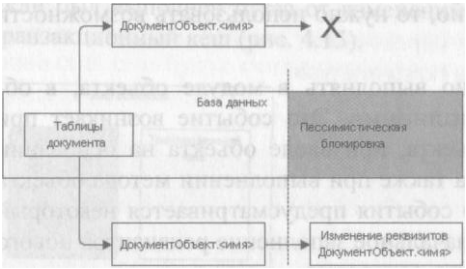


Рис. 4.16. Пессимистическая блокировка

Если говорить о системе, то механизмом пессимистической блокировки управляют расширения форм прикладных объектов. В тот момент, когда пользователь начинает модификацию объекта в форме, расширение формы устанавливает пессимистическую блокировку. Если после этого другой пользователь, например, попытается выполнить редактирование того же объекта, ему будет выдано сообщение о том, что не удалось заблокировать объект. Когда пользователь, редактировавший объект, закроет форму объекта, расширение формы снимет пессимистическую блокировку.

Поэтому пессимистическую блокировку можно рассматривать как блокировку данных для редактирования, связанную с идентификатором конкретной формы, в которой редактируется объект.

Расширения стандартных форм объектов устанавливают пессимистическую блокировку автоматически, но разработчик, находясь в модуле формы, может с помощью метода формы `ЗаблокироватьДанныеФормыДляРедак- тирования()` обеспечить поведение любой другой формы, аналогичное стандартному. Для снятия блокировки используется метод формы `Разбло- кироватьДанныеФормыДляРедактирования ()`. При использовании этих методов блокировка будет привязана к идентификатору той формы, из которой она установлена.

Кроме того, чтобы задействовать пессимистическую блокировку, разработчик может использовать методы глобального контекста `ЗаблокироватьДан- ныеДляРедактирования()` и `РазблокироватьДанныеДляРедактиро- вания()`. В эти методы обязательно нужно передать ссылку на объект информационной базы или ключ записи регистра сведений, данные которого требуется заблокировать.

Если блокировка устанавливается с привязкой к конкретной форме, то в метод `ЗаблокироватьДанныеДляРедактированияO` нужно передать также идентификатор этой формы. В этом случае для снятия блокировки методом `РазблокироватьДанныеДляРедактирования ()` указывается этот же идентификатор.

Можно не использовать в методах глобального контекста идентификатор формы. В этом случае устанавливаемая пессимистическая блокировка не будет привязана к конкретной форме. Например, это может понадобиться в обработке, выполняющей программную модификацию данных параллельно работе других пользователей.

Перед модификацией данных нужно попытаться их заблокировать. Если попытка удалась, данные можно модифицировать. Если объект уже заблокирован, то будет вызвано исключение, которое может быть обработано конструкцией `Попытка ... Исключение ... КонецПопытки` (листинг 4.6).

Листинг 4.6. Пример проверки пессимистической блокировки

```
&НаСервере

Функция ПримерМодификации()

ТоварСсылка = Справочники.Товары.НайтиПоКоду("00000000Г");

Попытка

ЗаблокироватьДанныеДляРедактирования(ТоварСсылка);
II Можно выполнять модификацию данных объекта II...
ТоварОбъект = ТоварСсылка.ПолучитьОбъект();
ТоварОбъект.Наименование = "Новое наименование";
ТоварОбъект.Записать();

Возврат Истина;

Исключение

II Нельзя модифицировать данные объекта Сообщение = Новый
СообщениеПользователю; Сообщение.Текст = "Данные объекта
уже заблокированы"; Сообщение.Сообщить();

Возврат  Ложь;
КонецПопытки;
КонецФункции
```

Следует помнить, что попытки установить блокировку одного и того же объекта с указанием идентификатора формы и без указания идентификатора несовместимы друг с другом.

Оптимистическая блокировка

Оптимистическая блокировка запрещает запись объекта в базу данных, если после считывания объекта он был изменен в базе данных (рис. 4.17).

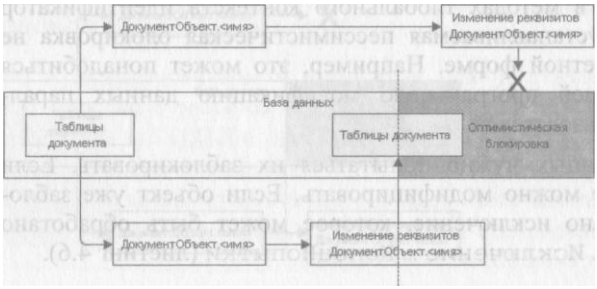


Рис. 4.17. Оптимистическая блокировка

Строго говоря, оптимистическая блокировка представляет собой проверку, которая выполняется перед записью объекта в базу данных. Когда программный объект считывает данные из базы данных, в числе прочего считывается и версия объекта, хранящегося в базе данных. Предположим, что до начала редактирования данных (до установки пессимистической блокировки) данные объекта в базе данных были изменены (например, другим пользователем). В этом случае номер версии объекта, хранящийся в базе данных, также изменится. При попытке первого пользователя записать или заблокировать этот объект будет выполнена проверка соответствия версии объекта, находящегося в памяти, и версии объекта, хранящейся в базе данных. Так как версии отличаются, будет выдано сообщение об ошибке, т. е. сработает оптимистическая блокировка.

Оптимистическая блокировка гарантирует, что если пользователь изменяет объект, то его изменения не «затрут» изменения, сделанные другими сессиями или другими программными объектами в этой же сессии.

Пессимистические блокировки и транзакции

Важным моментом является то, что пессимистическая и оптимистическая блокировки обеспечиваются не средствами базы данных, а средствами собственного менеджера блокировок, который работает «над уровнем» базы данных.

По этой причине объектные блокировки абсолютно прозрачны для транзакций. Единственная особенность заключается в том, что если пессимистическая блокировка была установлена в транзакции и в результате был выполнен откат этой транзакции (rollback), то блокировка будет автоматически снята.

Подробнее о транзакциях можно прочитать в разделе «Транзакции» на стр. 188.

Необъектные данные

Модель хранения данных

К необъектным данным в «1С:Предприятии» относятся данные следующих объектов конфигурации:

- регистр сведений,
- регистр накопления,
- регистр бухгалтерии,
- регистр расчета,
- перерасчет,

и последовательность,

- константа.

Необъектные данные с точки зрения «1С:Предприятия» представляют собой некоторый набор записей, которые хранятся в таблице. Каждая из этих записей полностью описывается значениями своих полей. Для системы эти записи не обладают какой-либо значимостью, кроме того, что в их полях содержатся некоторые значения. Запись можно удалить, а затем создать новую, с такими же значениями полей. Состояние базы данных при этом не изменится (с точки зрения логики прикладного решения). Это принципиально отличает необъектные данные от объектных: объект нельзя создать дважды, он ценен сам по себе, самим фактом своего существования.

Второе важное отличие заключается в том, что, изменив значения полей записи, мы получаем другую запись, в то время как изменение значений полей объекта не влечет за собой появление нового объекта.

Большинство необъектных сущностей конфигурации имеют общий порядок работы с данными. Из всей совокупности необъектных сущностей выделяются только константы: для каждой константы в базе данных хранится одно значение. Поэтому в дальнейшем мы их рассматривать не будем; работа с ними не вызывает сложностей.

Подчинение регистратору

Важным свойством всех записей является их подчинение регистратору. Регистратор - это некоторый документ, с которым связаны записи необъектных данных. Не может существовать записей, не относящихся к какому-либо регистратору. Исключение составляет лишь регистр сведений, когда для него выбран независимый режим записи (без подчинения регистратору). Поэтому любая запись всегда содержит обязательное поле Регистратор (рис. 4.18).

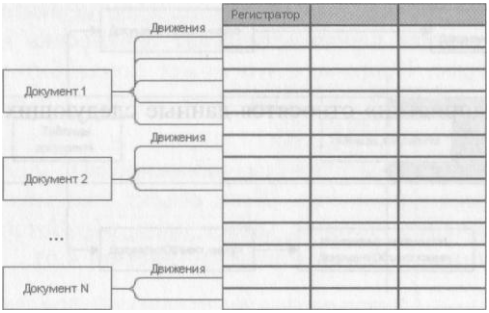


Рис. 4.18. Поле «Регистратор»

Записи, относящиеся к одному регистратору, называются движениями этого регистратора. Движения не являются частью владеющего ими документа, при записи и чтении документа они не записываются и не считываются. Однако они тесно связаны с ним.

Движения могут создаваться при проведении документов, при записи документов, могут формироваться интерактивно или в виде набора записей регистра (с отбором по регистратору).

Движения не могут существовать отдельно от своих регистраторов. Поэтому при удалении документов или отмене проведения (если свойство `Удаление движений` установлено в значение, отличное от `Не удалять`) система производит поиск движений этого документа в таблицах всех регистров, для которых данный документ может быть регистратором, и удаляет их.

Кроме этого, при удалении документа его движения всегда удаляются. Таким образом, время жизни записей определяется их регистратором: если регистратора нет, то и нет смысла в существовании его движений. Даже в том случае, когда записи создаются без непосредственного участия регистратора, они все равно обязательно должны быть подчинены какому-либо документу; запись нельзя записать, если в ней не указана ссылка на регистратор.

Наличие или отсутствие движений не связано с проведенностью документа- регистратора или с пометкой его на удаление: непроведенный документ может иметь движения, проведенный документ может не иметь движений, помеченный на удаление документ также может иметь движения. Такой подход позволяет реализовывать в системе «1С:Предприятие» различные способы регистрации изменений в учетных механизмах (регистрах).

Например, движения документа могут редактироваться непосредственно в самом документе, и в этом случае понятие проведения документа просто

не имеет смысла. Такой прием используется при автоматизации бухгалтерских задач, для ручного ввода проводок.

Также дата движений документа не связана жестко с датой регистратора. Например, один и тот же документ может иметь движения разными датами. Эта возможность используется, например, при автоматизации задач планирования, для ввода тех или иных планируемых в будущем значений.

Наиболее простой моделью использования регистров является создание движений в обработчике проведения регистратора. В этом случае вся остальная логика работы будет поддерживаться системой автоматически. Другие варианты взаимодействия движений и регистратора следует разрабатывать и описывать самостоятельно, в зависимости от конкретной прикладной задачи.

Уникальность записей

Для каждого объекта метаданных в системе определен ключ записи. Данные объекта конфигурации не могут содержать записи с одинаковыми значениями ключа записи.

Ключ записи формируется, как правило, из значений нескольких полей объекта конфигурации. Для разных объектов конфигурации состав ключа записи отличается, кроме этого он может быть различным для одного и того же объекта конфигурации, в зависимости от его свойств.

Для всех необъектных сущностей, подчиненных регистратору, ключ записи включает ссылку на регистратор и номер строки. Номер строки, собственно, и используется для обеспечения уникальности записей, а также для упорядочивания записей в пределах регистратора. Кроме этого, в состав ключа записи могут входить и другие поля, например, период, измерения регистра - для необъектных сущностей, не подчиненных регистратору.

Наличие уникального ключа требуется системе для решения различных задач. Например, это позволяет позиционироваться в табличном поле на некоторую запись.

Набор записей

По аналогии с объектными данными, изменение которых возможно только при помощи типов объектов, наборы записей используются для модификации необъектных данных: они позволяют читать, модифицировать и удалять необъектные данные. По своей сути набор записей представляет собой коллекцию отдельных записей, принадлежащих некоторому объекту конфигурации.

В зависимости от объекта конфигурации набор записей может содержать только некоторые или же все записи, принадлежащие этому объекту конфигурации. Для указания того, какие именно записи должны входить в набор записей, используется его свойство Отбор. Элементы отбора в наборе записей для того или иного объекта конфигурации создаются платформой; разработчик не имеет возможности добавлять собственные элементы отбора, но может использовать существующие, устанавливая для них условия равенства нужному значению. В таблице 4.1 представлен состав отбора в наборе записей для различных объектов конфигурации.

Таблица 4.1. Поля, по которым устанавливается отбор

Объект конфигурации	**Поля, по которым устанавливается отбор**
Регистр сведений (подчиненный регистратору)	Регистратор
Регистр сведений (непериодический, независимый)	Набор измерений
Регистр сведений (периодический, независимый)	Период и набор измерений
Регистр накопления	Регистратор
Регистр бухгалтерии	Регистратор
Регистр расчета	Регистратор и набор измерений
Перерасчет	Объект перерасчета (регистратор регистра расчета) и набор измерений перерасчета
Последовательность	Регистратор

Существует следующая особенность работы системы с наборами записей: если среди элементов отбора существует отбор по регистратору, то он должен обязательно быть установлен, иначе при записи такого набора будет выдано сообщение об ошибке. Отсюда следует, что для большинства объектов конфигурации, которые хранят необъектные данные, допускается модификация этих данных только частями, «гранулами» - наборами записей, относящихся к определенному регистратору (или более мелкими, если есть возможность установки других отборов). Наряду с этим для независимых регистров сведений допускается модификация сразу всех данных, хранящихся в регистре (если ни одно из условий отбора не задано).

Например, для независимого регистра сведений Комплектующие номенклатуры набор записей может включать в себя как все записи регистра (если отбор не установлен), так и лишь некоторые его записи (рис. 4.19).

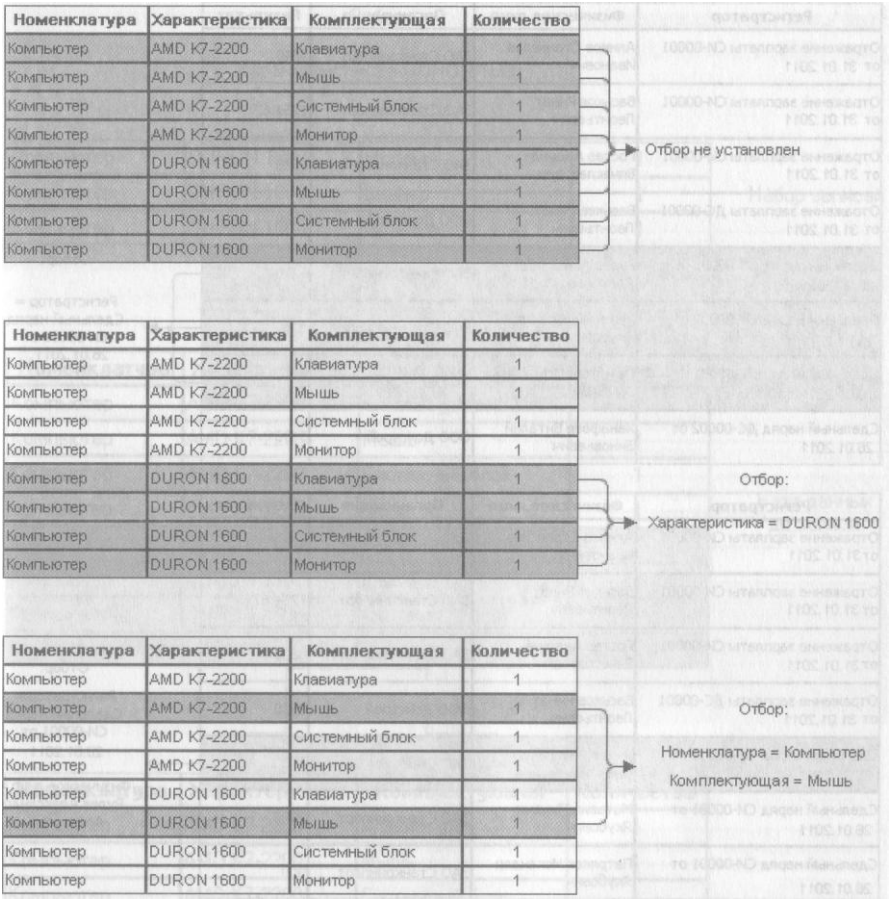


Рис. 4.19. Примеры наборов записей

В то же время для регистра расчета Основные начисления, у которого обязательно должен быть установлен отбор по регистратору, набор записей может включать в себя максимум все записи, принадлежащие одному регистратору, или некоторое их подмножество (рис. 4.20).

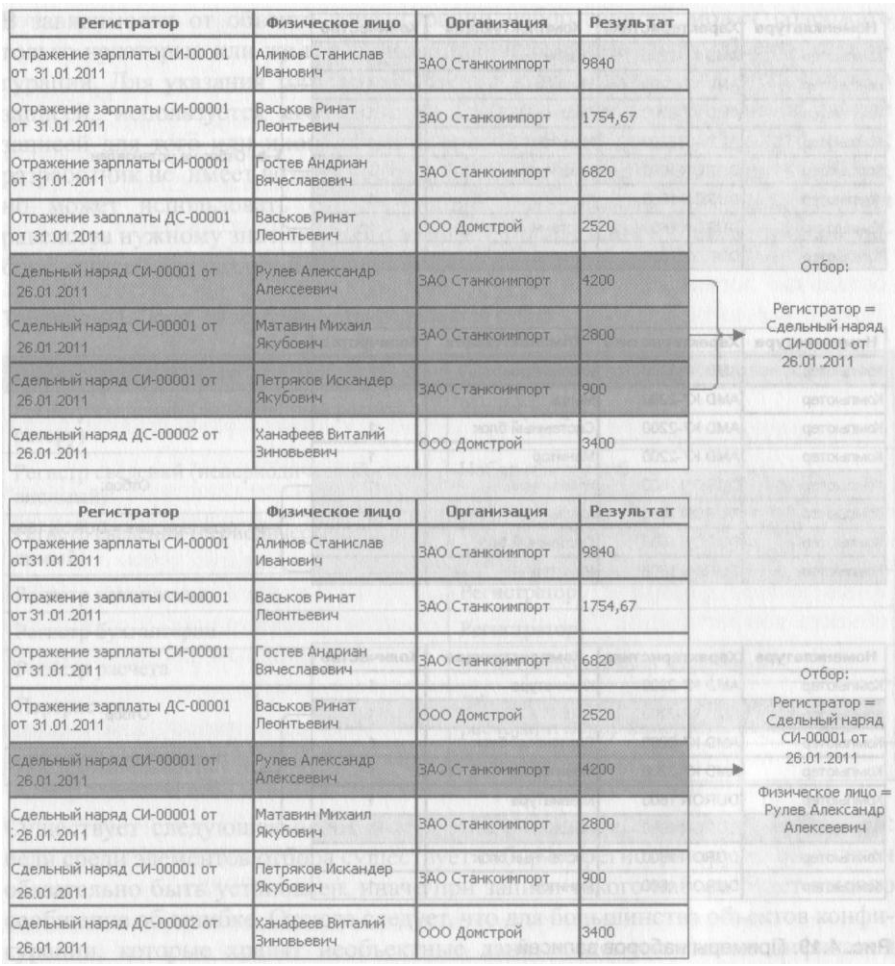


Рис. 4.20. Примеры наборов записей

Отличительной особенностью наборов записей является то, что для них не существует понятия удаления. Набор записей можно только записать, причем запись может быть выполнена либо с замещением существующих записей, удовлетворяющих отбору, либо с добавлением новых записей к существующим.

При записи с добавлением новые записи будут добавлены к существующим (рис. 4.21).

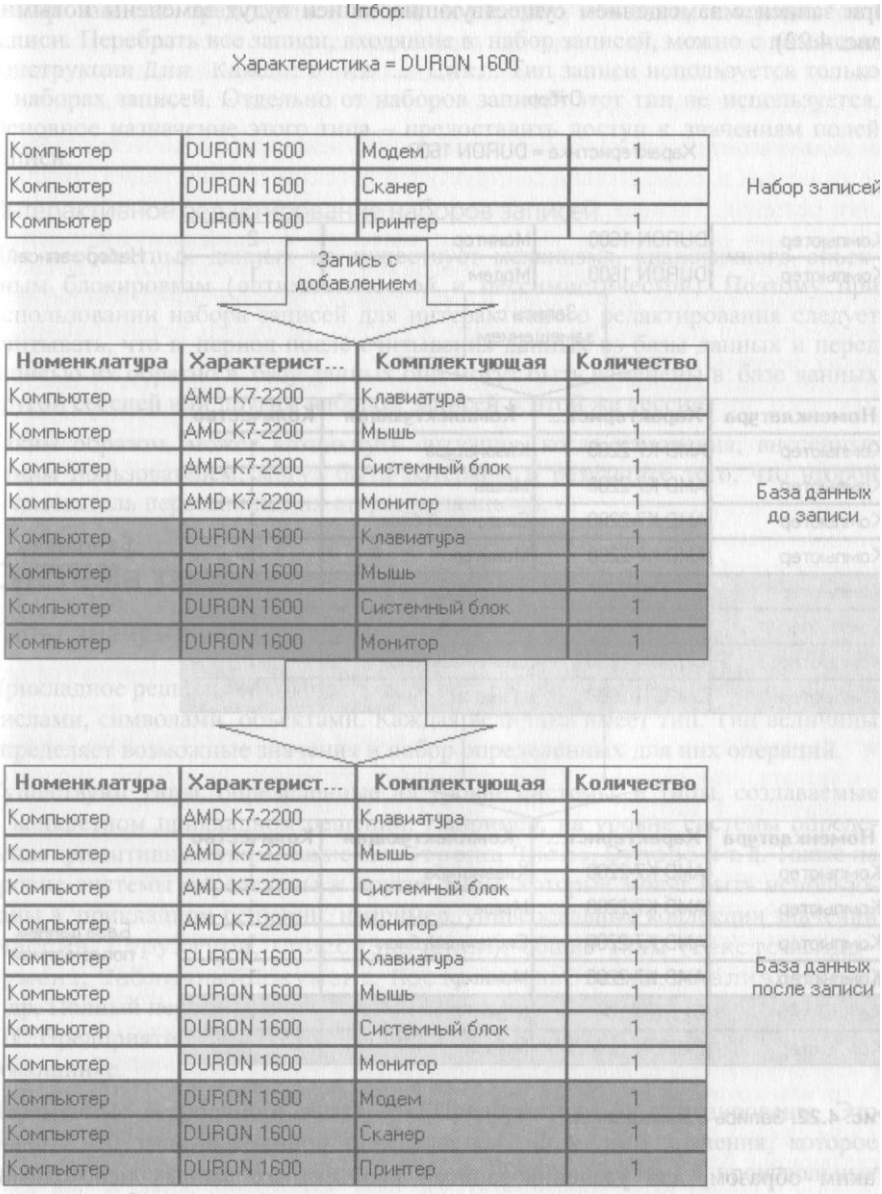


Рис. 4.21. Запись с добавлением

При записи с замещением существующие записи будут заменены новыми (рис. 4.22).

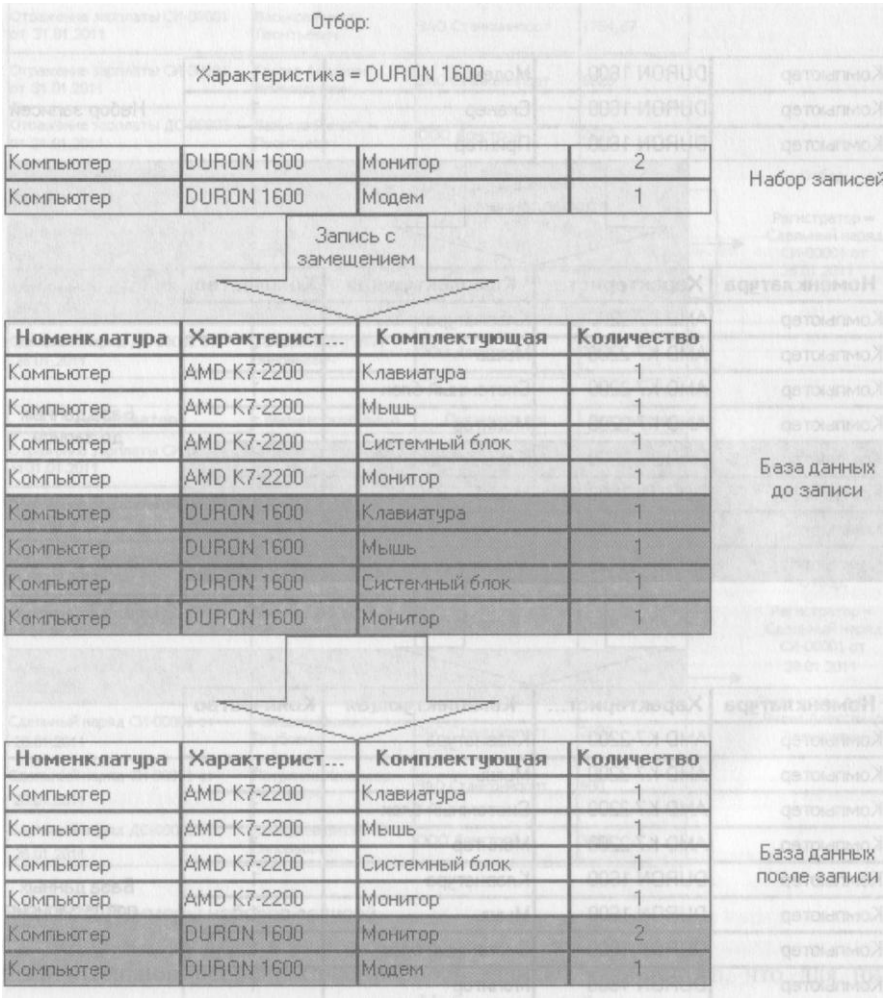


Рис. 4.22. Запись с замещением

Таким образом, для удаления записей необъектных данных необходимо просто записать с замещением пустой набор записей (т.е. набор записей, не содержащий ни одной записи).

Набор записей представляет собой коллекцию объектов, имеющих тип записи. Перебрать все записи, входящие в набор записей, можно с помощью конструкции `Для Каждого Из ... Цикл`. Тип записи используется только в наборах записей. Отдельно от наборов записей этот тип не используется. Основное назначение этого типа - предоставить доступ к значениям полей записи.

Интерактивное редактирование наборов записей

Для необъектных данных не существует механизма, аналогичного объектным блокировкам (оптимистической и пессимистической). Поэтому при использовании набора записей для интерактивного редактирования следует учитывать, что в период после считывания данных из базы данных и перед записью их обратно в базу данных они могут быть изменены в базе данных другой сессией или другим набором записей в этой же сессии.

Таким образом, может возникнуть ситуация, когда изменения, внесенные одним пользователем, могут быть потеряны в результате того, что второй пользователь перезапишет их старыми данными.

Система типов

Типы значений и типы данных

Прикладное решение «1С:Предприятия» оперирует различными величинами: числами, символами, объектами. Каждая величина имеет тип. Тип величины определяет возможные значения и набор определенных для них операций.

Существуют типы, определенные на уровне системы, и типы, создаваемые в конкретном прикладном решении. Например, на уровне системы определены примитивные типы, такие как `Строка`, `Число`, `Булево` и т.д. Также на уровне системы определены и другие типы, которые могут быть использованы в прикладном решении, например, универсальные коллекции значений (`Массив`, `Структура`, `СписокЗначений`), общие типы (`ТекстовыйДо- кумент`, `ТабличныйДокумент`, `ПостроительОтчета`, `АнализДанных`) и др. Полный перечень типов значений, которые может использовать система «1С:Предприятие», приведен в описании встроенного языка и в синтакс-помощнике.

Переменные встроенного языка «1С:Предприятия» не типизированы. Это значит, что тип переменной определяется типом того значения, которое хранится в переменной в данный момент. Таким образом, в произвольные моменты работы прикладного решения переменная может иметь различные типы значения. Подробнее о типах значений можно прочитать в разделе «Типы значений» на стр. 135.

В то же время данные «1С:Предприятия» существуют не только в оперативной памяти компьютера, где они содержатся в объектах встроенного языка, но и в базе данных, где осуществляется долговременное хранение этих данных.

База данных представляет собой совокупность некоторого количества таблиц, создаваемых в соответствии со структурой объектов конфигурации прикладного решения. Таблицы базы данных состоят из полей, и для каждого поля обязательно должен быть указан тип значений, которые могут храниться в этом поле. По этой причине все объекты конфигурации, которые «отвечают» за создание тех или иных полей в базе данных, должны иметь совершенно определенный тип. Такими объектами являются, например, реквизиты, измерения, ресурсы и т.д.

Для того чтобы указать тип объекта конфигурации, «отвечающего» за создание того или иного поля базы данных, в системе «1С:Предприятие» используются не типы значений, а более сложное понятие - *тип данных*.

Такой подход позволяет, с одной стороны, изолировать разработчика от конкретного хранилища данных, а с другой стороны - сделать работу с данными значительно более гибкой.

Отличие типа данных от типа значения заключается в том, что тип данных является характеристикой, которая может содержать описание как одного, так и нескольких типов значений. В результате у разработчика появляются две важные возможности:

- Уточнение некоторых примитивных типов значений. Например, можно указать, что реквизит Количество будет содержать не просто числовые значения, а неотрицательные числовые значения с количеством разрядов не более 15 и дробной частью из 3 разрядов.
- Указание одновременно нескольких возможных типов значений. Например, можно указать, что реквизит Сделка может содержать ссылку как на счет, выставленный покупателю, так и на счет, выставленный поставщику. В результате в реквизите Сделка смогут одновременно храниться как значения одного, так и другого типа. При этом тип данных этого реквизита будет всегда одним и тем же, но тип значения этого реквизита в каждый конкретный момент времени будет определяться типом значения, которое в нем хранится.

Типы данных используются в системе «1С:Предприятие» не только в части, связанной с базой данных, но и в интерфейсной части. Так, например, типы данных должны быть определены для всех реквизитов форм и для всех элементов управления, расположенных в форме.

Кроме этого, некоторые объекты встроенного языка в силу специфики своего использования также могут потребовать указания типа данных, которые в них хранятся. Например, указание типа данных может использоваться для списка значений, для колонок таблицы значений и дерева значений.

Подробнее о типах данных можно прочитать в разделе «Типы данных» на стр. 177.

Типы значений

Примитивные типы

Число

Числовой тип предназначен для представления десятичных чисел. Максимально допустимая разрядность числа, которая может быть сохранена в базе данных, составляет 32 знака, включая десятичную точку. Для числовых значений, хранящихся в памяти (являющихся, например, значением некоторой переменной), разрядность неограниченна.

Литерал значения типа «Число»

Конкретные значения числового типа во встроенном языке и языке запросов могут быть заданы литералом, который представляет собой набор цифр, написанных непосредственно в тексте модуля. Этот набор цифр может начинаться с символов «+» или «-», обозначающих знак числа. В качестве разделителя целой и дробной части числа используется символ «.» (точка), табл. 4.2.

Таблица 4.2. Примеры литералов типа «Число»

Число	Литерал
3874	3874
-1475,25	-1475.25

Значение типа «Число» по умолчанию

Для числового типа значением по умолчанию является значение 0. По умолчанию, если не задан специальный формат, это значение будет представляться как 0.

Однако некоторые элементы управления, используемые в интерфейсе системы, имеют другое представление нулевого значения, что сделано для более удобного представления информации пользователю.

Например, в ячейке таблицы значение по умолчанию числового типа отображается как пустое место (рис. 4.23).

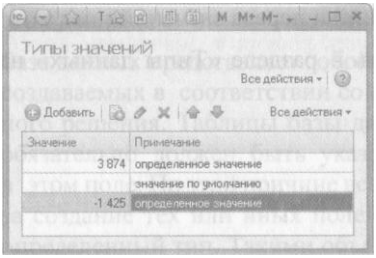


Рис. 4.23. Представление значения типа «Число» по умолчанию в колонке таблицы

Такое поведение таблицы позволяет создать список, который легко читается и не содержит лишней информации. Однако если есть необходимость указывать для нулевых значений определенное представление, то это можно выполнить, используя свойство `Формат` и `ФорматРедактирования` соответствующей колонки таблицы (в приведенном примере - колонки ТЗЗна- чение), рис. 4.24.

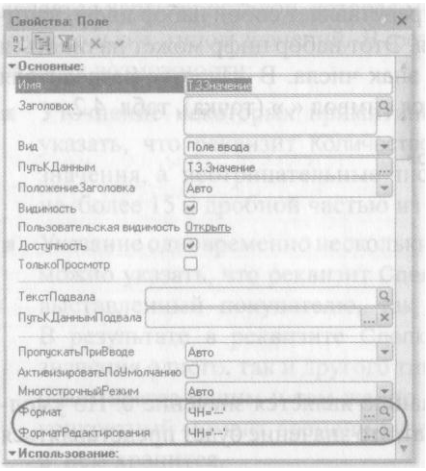


Рис. 4.24. Установка формата колонки таблицы

Для колонки таблицы форматная строка поддерживает параметр `ЧН`, который задает представление нулевого значения. В результате приведенная выше таблица будет выглядеть, например, следующим образом (рис. 4.25).

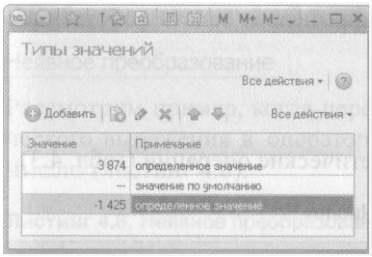


Рис. 4.25. Представление нулевого значения в колонке таблицы

Аналогичного результата можно добиться средствами встроенного языка (листинг 4.7).

Листинг 4.7. Установка формата колонки таблицы

```
Элементы.ТЗЗначение, Формат g "ЧН='—'";
Элементы.ТЗЗначение.ФорматРедактирования = "ЧН='—
```

При задании представления нулевого значения для колонки таблицы следует иметь в виду, что концевые пробелы при выводе заданного представления нулевого значения в ячейке таблицы будут обрезаны.

В поле ввода значение по умолчанию числового типа отображается в зависимости от разрядности дробной части, заданной при указании числового типа данных этого поля ввода (подробнее о числовых типах данных можно прочитать в разделе «Числовые типы данных» на стр. 178).

Если количество разрядов дробной части равно нулю, то числовое значение будет отображаться как 0. Если количество разрядов дробной части отлично от нуля, то числовое значение будет отображено как 0,000 (количество нулей после запятой будет соответствовать количеству разрядов дробной части), рис. 4.26.

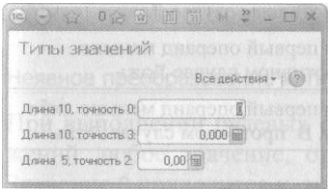


Рис. 4.26. Представление значения типа «Число» по умолчанию в поле ввода

В отличие от колонки таблицы, для поля ввода нельзя задать собственное представление нулевого значения.

Операции со значениями типа «Число»

Для значений типа `Число` определены арифметические операции (табл. 4.3).

Таблица 4.3. Арифметические операции для типа «Число»

Операция	Операнды	Результат
Сложение	Число + Число	Сумма операндов
Вычитание	Число - Число	Разность операндов
Умножение	Число * Число	Произведение операндов
Деление	Число / Число	Частное операндов. Если второй операнд имеет значение 0, будет вызвана ошибка исполнения
Остаток от деления	Число % Число	Остаток от деления первого операнда на второй. Если второй операнд имеет значение 0, будет вызвана ошибка исполнения
Унарный минус	- Число	Изменение знака числа

Для значений типа `Число` определены логические операции (табл. 4.4).

Таблица 4.4. Логические операции для типа «Число»

Операция	Операнды	Результат
Больше	Число > Число	`Истина`, если первый операнд больше второго. В противном случае `Ложь`
Больше или равно	Число >= Число	`Истина`, если первый операнд больше либо равен второму. В противном случае Ложь
Равно	Число = Число	`Истина`, если первый операнд равен второму. В противном случае `Ложь`
Не равно	Число ○ Число	`Истина`, если первый операнд не равен второму. В противном случае Ложь
Меньше	Число < Число	`Истина`, если первый операнд меньше второго. В противном случае Ложь
Меньше или равно	Число <- Число	`Истина`, если первый операнд меньше либо равен второму. В противном случае Ложь

Преобразование значений типа «Число» Неявное преобразование

Рассмотрим пример, когда переменная, указанная в качестве значения логического выражения в операторе цикла, по каким-либо причинам имеет тип Число (листинг 4.8).

Листинг 4.8. Неявное преобразование значения типа «Число»

```
Условие = 22; Если

Условие Тогда

КонецЕсли;
```

В этом случае будет выполняться неявное преобразование значения типа Число к нужному типу. Такое преобразование в ряде случаев может быть выполнено системой. Если же такое преобразование невозможно, будет вызвана ошибка исполнения.

Неявное преобразование к типу «Строка»

Любое значение типа Число может быть преобразовано к типу Строка. Это позволяет всегда иметь возможность получить представление такого значения (листинг 4.9).

Листинг 4.9. Неявное преобразование значения типа «Число» к значению типа «Строка»

```
Переменная =
156935.785;
Сообщить(Переменная);
```

Результатом выполнения этих операторов будет представление числа: 156935,785. Преобразование будет выполняться в соответствии с текущими региональными настройками информационной базы, например, как в данном случае, группы разрядов будут отделены друг от друга символом «пробел».

Неявное преобразование к типу «Булево»

При выполнении булевых операций или при вычислении логических выражений любое значение, отличное от 0, приводится к значению Истина. Значение 0 приводится к значению Ложь (листинг 4.10).

Листинг 4.10. Неявное преобразование значения типа «Число» к значению типа «Булево»

```
Условие = 22;
Сообщение = Новый СообщениеПользователю; Если Условие Тогда
        Сообщение.Текст = "Условие истинно.";
иначе
        Сообщение.Текст = "Условие ложно.";
КонецЕсли;
Сообщение.Сообщить{);
```

В приведенном примере будет получено сообщение об истинности условия, поскольку при вычислении логического выражения значение переменной Условие будет преобразовано к типу Булево, результатом чего будет значение Истина.

Также, например, при выполнении булевой операции Не в следующем примере сначала значение переменной Условие будет преобразовано к типу Булево (значение Истина), а затем уже выполнена операция НЕ. В результате в окно сообщений будет выведено Нет (листинг 4.11).

Листинг 4.11. Неявное преобразование значения типа «Число» к значению типа «Булево»

```
Условие = 22;
Сообщить(НЕ Условие);
```

Значения логического типа Истина/Ложь будут представлены в соответствии с языком интерфейса платформы, выбранным в параметрах конфигуратора или заданным в параметре /L командной строки при запуске «1С:Предпри- ятия».

Если «1С:Предприятие» запущено с русским языком интерфейса, логические значения будут представлены как «Да»/«Нет», с английским - «Yes»/«No», с турецким - «Evet»/«Hayir» и т. д.

Например, в приведенном примере для русского языка интерфейса будет выведено Нет, а если платформа будет запущена с литовским интерфейсом, то - Ne.

Неявное преобразование в операциях сравнения

Порядок неявного преобразования типов в операциях сравнения несколько отличается от описанного выше. Если в операции сравнения один из операндов имеет тип Число, а другой - Булево, то значение типа Булево будет приводиться к типу Число, а затем будет выполнено сравнение.

Например, в результате выполнения следующего кода (листинг 4.12) будет получено сообщение о ложности условия, поскольку булев операнд будет приведен к типу Число (значение 1), а затем уже будет выполнено сравнение.

Листинг 4.12. Неявное преобразование в операции сравнения Условие = 22;

```
Сообщение = Новый СообщениеПользователю;
Если Истина = Условие Тогда
        Сообщение.Текст = "Условие истинно.";
иначе
        Сообщение.Текст = "Условие ложно.";
КонецЕсли;
Сообщение.Сообщить();
```

Явное преобразование

Преобразование к типу «Булево»

Встроенный язык позволяет выполнить явное преобразование значения типа `Число` к типу `Булево`. Для этого используется встроенная функция `Булево`(), в качестве параметра которой передается преобразуемое число. Значение 0 преобразуется в `Ложь`, все остальные значения преобразуются в значение `Истина`. Например, в результате выполнения оператора (листинг 4.13) будет получено сообщение Да.

Листинг 4.13. Явное преобразование к типу «Булево»

```
Сообщить(Булево(34.456));
```

А в результате выполнения следующего оператора (листинг 4.14) будет получено сообщение Нет.

Листинг 4.14. Явное преобразование к типу «Булево»

```
Сообщить(Булево(0));
```

Преобразование к типу «Строка»

Любое числовое значение может быть преобразовано к типу `Строка` при помощи встроенной функции `Строка`(). Результатом такого преобразования будет строковое представление числа, полученное в соответствии с текущими региональными установками информационной базы.

Например, результатом выполнения оператора (листинг 4.15) будет сообщение 26475 834,456, в котором используются в данном случае разделители групп разрядов (символ «пробел») и десятичный разделитель (символ «запятая»).

Листинг 4.15. Явное преобразование к типу «Строка»

```
Сообщение = Новый СообщениеПользователю;
Сообщение .Текст = Строка(26475834.456);
Сообщение.Сообщить();
```

Строка

Строковый тип предназначен для представления строк в формате Unicode произвольной длины.

Литерал значения типа «Строка»

Для того чтобы задать конкретные значения строкового типа, во встроенном языке и языке запросов используются литералы строкового типа. Они представляют собой набор символов, заключенных в кавычки (табл. 4.5).

Таблица 4.5. Примеры литералов типа «Строка»

Строка	Литерал
книга	"книга"
проверка работы	"проверка работы"

Если в строке необходимо задать символ кавычка ("), то записываются две кавычки подряд (табл. 4.6).

Таблица 4.6. Пример литерала типа «Строка»

Строка	Литерал
фирма "Ваш сад"	"фирма ""Ваш сад"""

Наряду с однострочными литералами во встроенном языке могут быть использованы строковые литералы, состоящие из нескольких строк. Для обозначения таких литералов используются два различных способа записи.

Во-первых, отдельные строки могут быть заключены в кавычки. В этом случае между ними не должно находиться никаких символов, кроме пробелов, переводов строки и комментариев. Например, строка Внимание! В документе не могут присутствовать строки с нулевым количеством! может быть записана следующим образом (листинг 4.16).

Листинг 4.16. Пример литерала типа «Строка»

```
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "Внимание!"
                  "В документе не могут присутствовать строки"
"с нулевым количеством!"; Сообщение.Сообщить();
```

Вторым способом указания многострочных литералов является использование символа «|». В этом случае кавычки используются только в начале и конце литерала, а символ «|» размещается в начале каждой новой строки (листинг 4.17).

Листинг 4.17. Пример литерала типа «Строка»

```
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = "Внимание!
                  |В документе не могут присутствовать строки
|с нулевым количеством!"; Сообщение.Сообщить();
```

Результатом как первого, так и второго оператора будет следующее сообщение: Внимание! В документе не могут присутствовать строки с нулевым количеством!

Значение типа «Строка» по умолчанию

Для строкового типа значением по умолчанию является пустая строка (литерал ""). Это значение не имеет представления, иначе говоря, оно не отображается. В некоторых случаях, например, при вычислении значений в отладчике, для наглядности в качестве представления этого значения может использоваться его литерал (").

Операции со значениями типа «Строка»

Для значений типа `Строка` определена операция конкатенации (табл. 4.7).

Таблица 4.7. Операция конкатенации

Операция	Операнды	Результат
Конкатенация	Строка + Строка	Строка, являющаяся соединением первой и второй строки. Длина результирующей строки равна сумме длин соединяемых строк

Для значений типа `Строка` определены логические операции (табл. 4.8).

Таблица 4.8. Логические операции

Операция	Операнды	Результат
Больше	Строка > Строка	`Истина`, если первая строка больше второй. В противном случае Ложь
Больше или равно	Строка >= Строка	`Истина`, если первая строка больше либо равна второй. В противном случае Ложь
Меньше	Строка < Строка	`Истина`, если первая строка меньше второй. В противном случае Ложь
Меньше или равно	Строка <= Строка	`Истина`, если первая строка меньше либо равна второй. В противном случае Ложь
Равно	Строка = Строка	`Истина`, если первая строка равна второй. В противном случае Ложь
Не равно	Строка о Строка	`Истина`, если первая строка не равна второй. В противном случае Ложь

При выполнении логических операций больше, больше или равно, меньше, меньше или равно существует особенность, связанная с тем, что система «1С:Предприятие» позволяет создавать прикладные решения на различных языках.

Внутри платформы «1С:Предприятие» работа со строками ведется только в UNICODE, и строковые данные в базах данных также сохраняются в UNICODE. Стандартом определен некоторый генеральный порядок сортировки для UNICODE. Но он подходит не для всех языков. Поэтому для некоторых языков в этот генеральный порядок внесены минимальные изменения для обеспечения соответствия конкретному языку.

Порядок сортировки строковых значений, принятый в конкретном регионе, определяет параметр Язык/Страна, выбранный в диалоге Региональные установки информационной базы.

В результате этого, например, для русских и латышских региональных установок будет различный порядок следования строк при сортировке их по возрастанию (табл. 4.9).

Таблица 4.9. Порядок следования строк для различных региональных установок

Русский(Россия)	Латышский (Латвия)
Aca	Acb
Acb	Aca

Такой порядок соответствует принятому в России и Латвии алфавиту.

Поскольку сортировка строковых значений выполняется на основе сравнения их друг с другом, то один и тот же оператор (листинг 4.18) при установке параметра Язык/Страна в диалоге Региональные установки информационной базы в значение Россия будет давать результат Истина, представленный в соответствии с русским языком интерфейса как Да, а если Язык/Страна установлен в значение Латвия, то - Ложь, представленный как Нет. Если запустить «1С:Предприятие» из командной строки с латышским языком интерфейса ("C:\Program Files\1Cv82\8.2.13.199\bin\1cv8.exe" /L lv), то будет выведено Ne, что значит Нет.

Листинг 4.18. Сравнение строк

```
Результат = "acb">
"aca";
Сообщить(Результат);
```

Таким образом, результат выполнения логических операций над строковыми значениями будет зависеть не только от самих операндов, участвующих в выражении, но и от региональных установок информационной базы, в которой выполняется эта операция.

Преобразование значений типа «Строка» Неявное преобразование

При выполнении различных операторов могут возникать ситуации, когда значения типа Строка не подходят для выполнения данных операций. Например, если при выполнении следующего оператора переменная Строка будет иметь значение 25 (листинг 4.19).

Листинг 4.19. Неявное преобразование значения типа «Строка»

```
Строка = "25";
Результат = 35 +
Строка;
Сообщить(Результат);
```

В этом случае будет выполняться неявное преобразование значения типа Строка к нужному типу.
Такое преобразование в ряде случаев может быть выполнено системой. Если же такое преобразование невозможно, будет вызвана ошибка исполнения.

Неявное преобразование к типу «Число»

Если выполнить приведенный выше пример, то строка 25 будет преобразована к типу Число и в окно сообщений будет выведено 60.

Однако следует заметить, что преобразование значения типа Строка к типу Число будет выполняться только тогда, когда подобное преобразование осмысленно.

Например, если значение переменной Строка будет равно а25, то преобразование уже выполнено не будет и будет вызвано исключение.

Неявное преобразование к типу «Булево»

Неявное преобразование значений строкового типа к типу Булево также выполняется лишь тогда, когда такое преобразование имеет смысл.

К значениям типа Булево будут преобразованы строки, соответствующие литералам булевых значений, т.е. Истина, Ложь, True и False.

Например, рассмотрим выполнение следующего фрагмента кода (листинг 4.20).

Листинг 4.20. Неявное преобразование к типу «Булево»

```
Сообщение = Новый
СообщениеПользователю; Если "истина" тогда
        Сообщение.Текст = "Истина";
Иначе
        Сообщение.Текст = "Ложь";
КонецЕсли;
Сообщение.Сообщить();

Если "false" тогда
        Сообщение.Текст = "Истина";
Иначе
        Сообщение.Текст = "Ложь";
КонецЕсли;
Сообщение.Сообщить();
```

В результате в окно сообщений будет выведен текст:
Истина

Ложь

Однако кроме этого к булевым значениям будут преобразовываться также строковые значения, соответствующие представлению булевых значений для языка, выбранного в языке интерфейса платформы. Например, если запустить «1С:Предприятие» из командной строки с литовским языком интерфейса ("C:\Program Files\1Cv82\8.2.13.199\bin\1cv8.exe" /L It), то тогда следующий код (листинг 4.21) будет выполнен без ошибок и в окно состояния будет выведено Ложь.

Листинг 4.21. Неявное преобразование к типу «Булево»

```
Сообщение = Новый СообщениеПользователю;
Если "Ne" тогда
        Сообщение.Текст = "Истина";
Иначе
        Сообщение.Текст = "Ложь";
КонецЕсли;
Сообщение.Сообщить();
```

Если этот же код попытаться выполнить с языком интерфейса `de (Немецкий)`, то будет вызвано исключение, в результате того что значение пе не может быть преобразовано к типу `Булево`. Если же в этой ситуации строку пе заменить на Nein, что значит Нет по-немецки, то пример (листинг 4.22) отработает без ошибок и в окно сообщений будет выведено Ложь.

Листинг 4.22. Неявное преобразование к типу «Булево»

```
Сообщение = Новый
СообщениеПользователю; Если "Nein" тогда
        Сообщение.Текст = "Истина";
Иначе
        Сообщение.Текст = "Ложь";
КонецЕсли;
Сообщение.Сообщить();
```

Явное преобразование

Преобразование к типу «Число»

Встроенный язык позволяет выполнить явное преобразование значения типа `Строка` к типу `Число` при помощи встроенной функции `Число()`. При этом будут преобразованы только те строки, которые представляют собой правильное строковое представление литерала численного типа. Например, может быть выполнено преобразование строки 1125.78 к числу 1125.78 (листинг 4.23).

Листинг 4.23. Явное преобразование к типу «Число»

```
Сообщить(Число{"112578"));
```

Следует заметить, что если в строке в качестве разделителя целой и дробной части будет использоваться не точка, а запятая, преобразование также будет выполнено (листинг 4.24).

Листинг 4.24. Явное преобразование к типу «Число»

```
Сообщить(Число{"1125,78"));
```

Преобразование к типу «Дата»

Строковое значение может быть преобразовано к типу Дата. Для этого используется встроенная функция Дата(). Она позволяет преобразовать строку, которая представляет собой части даты, записанные в определенном порядке: год, месяц, день, час, минута и секунда.

Например, для того, чтобы получить дату 15 апреля 2011 года 17 часов 45 минут 34 секунды, преобразуемая строка должна иметь вид 20110415174534 (листинг 4.25).

Листинг 4.25. Преобразование к типу «Дата»

```
Сообщить(Дата("20110415174534"));
```

В результате выполнения приведенного кода будет получена нужная дата:
15.04.2011 17:45:34.

Дата

Тип Дата предназначен для представления значения даты и времени с точностью до секунды. Минимальным значением типа Дата является дата
01 января 0001 года 00 часов 00 минут 00 секунд.

Литерал значения типа «Дата»

Конкретные значения типа Дата могут быть заданы литералом, который представляет собой последовательность цифр, заключенных в одинарные кавычки. Последовательность цифр представляет собой части даты в следующем порядке: год, месяц, день, час, минута и секунда. Части даты могут быть отделены друг от друга различными разделителями; час, минута и секунда могут быть не указаны - в этом случае предполагается, что они равны нулю (табл. 4.10).

Таблица 4.10. Примеры литералов типа «Дата»

Дата	Литерал
15.04.2011 22:45:33	'20110415224533'
15.04.2011 0:00:00	'20110415'
15.04.2011 22:45:33	'2011-04-15 22:45:33'
15.04.2011 22:45:33	'2011/04/15-22/45/33'

В языке запросов для указания литералов типа Дата используется ключевое слово ДАТАВРЕМЯ, после которого в скобках последовательно указываются год, месяц, день, час, минута и секунда. Последние три указывать не обязательно (табл. 4.11).

Таблица 4.11. Примеры литералов типа «Дата»

Дата	Литерал
15.04.2011 22:45:33	ДАТАВРЕМЯ(2011, 04, 15, 22, 45, 33)
15.04.2011 0:00:00	ДАТАВРЕМЯ(2011, 04, 15)

Значение типа «Дата» по умолчанию

Для типа `Дата` значением по умолчанию является 01 января 0001 года 00 часов 00 минут 00 секунд.

При выводе значения даты по умолчанию она отображается в соответствии с используемыми региональными установками сеанса (например, 01.01.0001 0:00:00).

Однако при выводе значения даты по умолчанию в некоторые элементы управления она будет представляться иначе, что обусловлено удобством ввода или отображения.

Например, значение даты по умолчанию в поле ввода будет отображено в виде символов-разделителей даты, используемых при заданных региональных установках. Состав разделителей будет определяться составом даты (`Дата и время`, `Дата`, `Время`), указанным при описании типа данных, с которыми работает данное поле ввода (рис. 4.27). Подробнее о типах данных, описывающих значения типа `Дата`, можно прочитать в разделе «Типы данных, описывающие значения типа «Дата»» на стр. 181.

Задать собственное представление даты по умолчанию можно с помощью свойства `ФорматРедактирования` поля ввода (рис. 4.28).

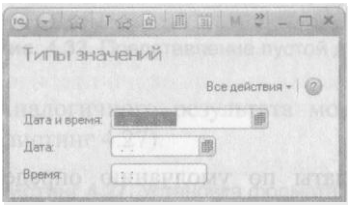


Рис. 4.27. Пустая дата в поле ввода

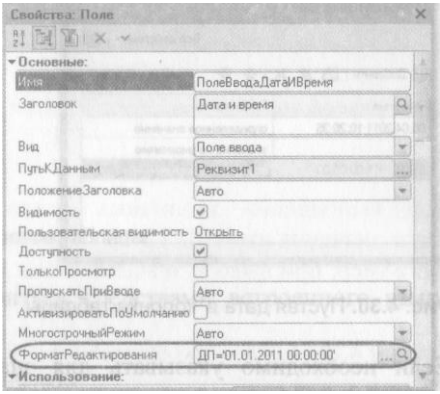


Рис. 4.28. Установка формата в поле ввода

В поле ввода форматная строка поддерживает параметр ДП, который задает представление значения даты по умолчанию. В результате приведенный выше пример будет выглядеть следующим образом (рис. 4.29).

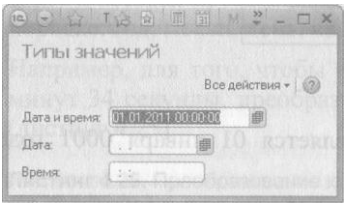


Рис. 4.29. Отображение пустой даты в поле ввода

Аналогичного результата можно добиться, используя средства встроенного языка (листинг 4.26).

Листинг 4.26. Установка формата в поле ввода

```
Элементы.ПолеВводаДатаИВремя.ФорматРедактирования = "ДП=*01.01.2011
00:00:00"'; Элементы.ПолеВводаДата.ФорматРедактирования = "ДП-01.01.2011"';
Элементы.ПолеВводаВремя.ФорматРедашрования = "ДП-00:00:00"';
```

При выводе значения даты по умолчанию в колонку таблицы она будет отображаться как пустое место, вне зависимости от того, какой состав даты установлен при описании типа данных (рис. 4.30).

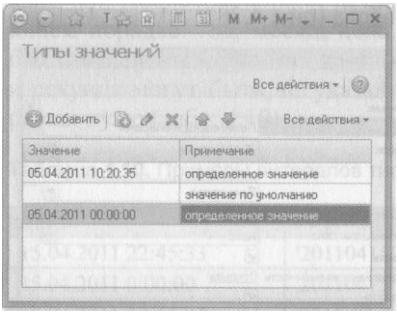


Рис. 4.30. Пустая дата в колонке таблицы

Если необходимо указывать для значений даты по умолчанию определенное представление, то это можно выполнить с помощью свойств Формат и ФорматРедактирования колонки таблицы (в приведенном примере - колонки ТЗЗначение), рис. 4.31.

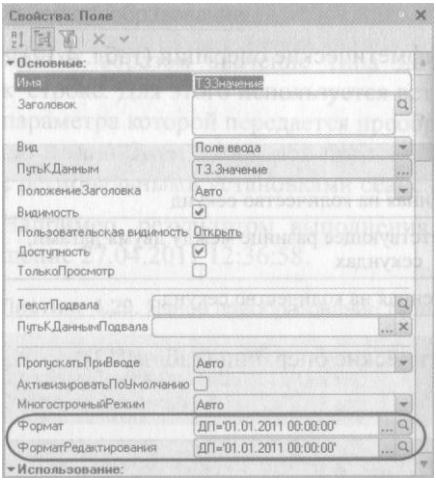


Рис. 4.31. Установка формата для колонки таблицы

Для колонки таблицы форматная строка поддерживает параметр ДП, который задает представление пустой даты. В результате приведенная выше таблица будет выглядеть, например, следующим образом (рис. 4.32).

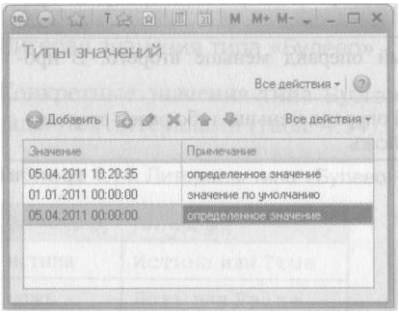


Рис. 4.32. Представление пустой даты в колонке таблицы

Аналогичного результата можно добиться средствами встроенного языка (листинг 4.27).

Листинг 4.27. Установка формата колонки таблицы

```
Элементы.ТЗЗначение.Формат = "ДП='01.01.2011 00:00:00'";
Элементы.ТЗЗначение.ФорматРедактирования = "ДП='01.01.2011
```

Операции со значениями типа «Дата»

Для значений типа Дата определены арифметические операции (табл. 4.12).

Таблица 4.12. Арифметические операции

Операция	Операнды	Результат
Сложение	Дата + Число	Дата, увеличенная на количество секунд
Вычитание	Дата - Дата	Число, соответствующее разнице между двумя датами, измеренной в секундах
Вычитание	Дата - Число	Дата, уменьшенная на количество секунд

Для значений типа Дата определены логические операции (табл. 4.13).

Таблица 4.13. Логические операции

Операция	Операнды	Результат
Больше	Дата > Дата	Истина, если первый операнд больше второго. В противном случае Ложь
Больше или равно	Дата >= Дата	Истина, если первый операнд больше либо равен второму. В противном случае Ложь
Равно	Дата = Дата	Истина, если первый операнд равен второму. В противном случае Ложь
Не равно	Дата ○ Дата	Истина, если первый операнд не равен второму. В противном случае Ложь
Меньше	Дата < Дата	Истина, если первый операнд меньше второго. В противном случае Ложь
Меньше или равно	Дата <= Дата	Истина, если первый операнд меньше либо равен второму. В противном случае Ложь

Преобразование значений типа «Дата»

Неявное преобразование

При выполнении различных операторов могут возникнуть ситуации, когда значения типа Дата не подходят для выполнения данных операций. Например, при выводе значений типа Дата может быть выполнено преобразование к строке (листинг 4.28).

Листинг 4.28. Неявное преобразование значения типа «Дата»

```
ПеременнаяДата =
'20110514234512';
Сообщить(ПеременнаяДата);
```

Явное преобразование

Явным образом значение типа Дата может быть преобразовано только к строке. Для этого используется встроенная функция Строка(), в качестве параметра которой передается преобразуемая дата. Результатом такого преобразования будет строковое представление даты, полученное в соответствии с региональными установками сеанса.

Например, результатом выполнения оператора (листинг 4.29) будет сообщение 27.04.2011 12:36:58.

Листинг 4.29. Явное преобразование типа «Дата»

```
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = Строка('20110427123658*};
Сообщение.Сообщить();
```

А для сеанса, запущенного с параметром командной строки /VL, равным el _ GR (Греция), результатом выполнения этого же оператора будет другое сообщение-27/4/2011 12:36:58 щц.

Булево

Тип Булево предназначен для представления логических величин. Он имеет только два значения: Истина и Ложь.

Литерал значения типа «Булево»

Конкретные значения типа Булево во встроенном языке и языке запросов задаются литералами (табл. 4.14).

Таблица 4.14. Литералы типа «Булево»

Значение	Литерал
истина	Истина или True
ложь	Ложь или False

Значение типа «Булево» по умолчанию

Значением типа Булево по умолчанию является значение Ложь.

Операции со значениями типа «Булево»

Для значений типа Булево определены операции сравнения. При сравнении булевых значений считается, что значение Истина больше, чем значение Ложь (табл. 4.15).

Таблица 4.15. Операции сравнения

Операция	Операнды
Больше	Булево > Булево
Больше или равно	Булево >= Булево
Равно	Булево = Булево
Не равно	Булево о Булево
Меньше	Булево < Булево
Меньше или равно	Булево <= Булево

Для значений типа Булево определены булевы операции (табл. 4.16). **Таблица**

4.16. Булевы операции

Операция	Операнды	Результат
И	Булево И Булево	Истина, если оба операнда имеют значение Истина. В остальных случаях Ложь
ИЛИ	Булево ИЛИ Булево	Ложь, если оба операнда имеют значение Ложь. В остальных случаях Истина
НЕ	НЕ Булево	Истина, если значение операнда Ложь. В противном случае Ложь

Преобразование значений типа «Булево» Неявное преобразование

При выполнении различных операторов могут возникать ситуации, когда значения типа Булево не подходят для выполнения данных операций (листинг 4.30).

Листинг 4.30. Неявное преобразование значений типа «Булево»

```
Число = 10; Булево =
Истина; Сообщить(Число +
Булево);
```

В этом случае может быть выполнено неявное преобразование значения типа Булево к нужному типу. Если такое преобразование не может быть выполнено, будет вызвана ошибка исполнения.

Неявное преобразование к типу «Число»

Если логика исполнения требует преобразования значения типа Булево к типу Число, то это преобразование выполняется по следующим правилам: значение Истина преобразуется к значению 1, а значение Ложь - к зна

чению 0. Таким образом, результатом выполнения предыдущего примера (листинг 4.30) будет строка «11».

Неявное преобразование к типу «Строка»

Неявное преобразование к типу Строка будет выполняться, например, в следующем примере (листинг 4.31).

Листинг 4.31. Неявное преобразование к типу «Строка»

```
Булево I Истина;
Сообщить(Булево);
```

В результате будет получено представление значения в соответствии с языком интерфейса платформы. Например, для русского интерфейса это будет значение Да, а для литовского - Taip.

Явное преобразование Преобразование

к типу «Число»

Встроенный язык позволяет выполнить явное преобразование значения типа Булево к типу Число. Для этого используется встроенная функция Число (), которой в качестве параметра передается булево значение. При этом значение Истина преобразуется в 1, а значение Ложь - в 0 (листинг 4.32).

Листинг 4.32. Преобразование к типу «Число»

```
Результат =
Чисяо(Истана);
```

Преобразование к типу «Строка»

Встроенный язык позволяет также преобразовывать значение типа Булево к типу Строка. Для этого используется встроенная функция Строка, которая возвращает представление булева значения в соответствии с языком интерфейса платформы (листинг 4.33).

Листинг 4.33. Преобразование к типу «Строка»

```
Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = Строка(Истина);
Сообщение,Сообщить();
```

В результате выполнения этого кода в окно сообщений будет выведено Да (для русского языка интерфейса).

Значение «Неопределено»

Значение типа Неопределено применяется тогда, когда необходимо использовать «пустое значение», не принадлежащее ни к одному другому типу данных, определенных в прикладном решении. Тип Неопределено имеет одно-единственное значение - Неопределено. Это значение используется, например, как значение составного типа по умолчанию. По этой причине составной тип всегда содержит тип Неопределено. О составном типе данных подробнее можно прочитать в разделе «Составной тип данных», см. стр. 182.

Представлением значения Неопределено является пустая строка. Литерал

значения типа «Неопределено»

Значение типа Неопределено во встроенном языке и языке запросов задается литералом Неопределено.

Операции со значением «Неопределено»

Для значения Неопределено определены операции сравнения на равенство и неравенство со значениями других типов.

Таким образом, для любой переменной всегда может быть выполнено сравнение на равенство (или неравенство) ее значения значению Неопределено (листинг 4.34).

Листинг 4.34. Сравнение со значением «Неопределено»

```
Сообщение = Новый СообщениеПользователю; Если
РеквизитФормыСоставной = Неопределено Тогда
Сообщение.Текст «"Истина";
Иначе
    Сообщение.Текст = "Ложь";
КонецЕсли;
Сообщение.Сообщить();
```

Преобразование значений типа «Неопределено»

Значение Неопределено может быть преобразовано только в значение типа Строка. Такое преобразование может выполняться системой неявно, например, при выводе значения Неопределено, или явно, с использованием встроенной функции Строка(). Результатом преобразования является пустая строка.

Null

Тип Null, так же как и тип Неопределено, имеет единственное значение - Null. Это значение используется системой для обозначения отсутствующего значения при работе с базой данных.

Такая ситуация может возникнуть, например, при соединении нескольких таблиц в запросах, когда для одной из соединяемых таблиц нет записей в другой таблице, удовлетворяющих указанным условиям.

Кроме приведенного примера, значение Null может быть получено и в других случаях. Например, когда выполняется обращение к реквизитам объекта, которые не используются. Допустим, у иерархического справочника Номенклатура существует реквизит Артикул, для которого свойство Использование установлено в значение ДляЭлемента. В этом случае поле Артикул будет доступно для редактирования только у элементов справочника. У групп справочника это поле не будет содержать никакой информации.

Поэтому при обращении к реквизиту Артикул у группы справочника Номенклатура система будет возвращать значение Null (листинг 4.35).

Листинг 4.35. Значение типа Null

```
Сообщение = Новый СообщениеПользователю;
Выборка =
Справочники.Номенклатура.ВыбратЦ);

Пока Выборка.Следующий) Цикл

                                    Если Выборка.ЭтоГрулпа Тогда Префикс = "Группа:";
      Иначе
            Префикс =
      "Элемент:": КонецЕсли;

      Сообщение.Текст? Префикс + СокрЛП(Выборка.Наименование) +

                                    ", тип артикула:" +ТипЗнч(Выборка.Артикул);

      Сообщение.Сообщить(); КонецЦикла;
```

В приведенном примере выбираются все элементы справочника Номенклатура, и затем выводится информация о типе значения реквизита Артикул для каждого элемента выборки.

Литерал значения типа Null

Значение Null во встроенном языке и языке запросов задается литералом Null, а представлением значения Null является пустая строка.

Операции со значениями типа Null

Для значения Null определены операции сравнения на равенство и неравенство со значениями других типов. В результате этого для любой переменной всегда может быть выполнено сравнение на равенство (или неравенство) ее значения значению Null (листинг 4.36).

Листинг 4.36. Сравнение со значением типа Null

```
Сообщение = Новый СообщениеПользователю;
Выборка =
Справочники,Номенклатура.Выбрать();

Пока Выборка.Следующий() Цикл

    Суффикс = ?(Выборка.Артикул = Null, "не используется", Выборка .Артикул);
    Сообщение.Текст = СокрЛП(Выборка.Наименование) +". артикул:" + Суффикс;
    Сообщение.Сообщить();

КонецЦикла;
```

В приведенном примере выбираются все элементы справочника Номенклатура. Затем для каждого элемента проверяется равенство его артикула значению `Null`, и в зависимости от этого формируется строка для вывода сообщения.

Преобразование значений типа Null

Значение `Null` может быть преобразовано только в значение типа `Строка`. Такое преобразование может выполняться системой неявно, например, при выводе значения `Null`, или явно, с использованием встроенной функции `Строка`(). Результатом преобразования является пустая строка.

Тип

Значения типа `Тип` используются для идентификации различных типов значений и их сравнения.

Значения этого типа возвращаются двумя системными функциями: `Тип()` и `ТипЗнч()`. Также значения этого типа могут быть получены из объекта `ОписаниеТипов` в виде массива значений (метод `Типы()`).

Функция `Тип` () позволяет получить значение этого типа, соответствующее переданному в качестве параметра имени некоторого типа (листинг 4.37).

Листинг 4.37. Использование функции «Тип()»

```
ТипСтрока =
Тип("Строка"}; ТипЧисло
= Тип("Число");
ТипССылкаНаСправочникНоменклатура = Тип("СправочникСсылка.Номенклатура");
```

Функция ТипЗнч() также позволяет получить значение типа Тип, однако в качестве параметра ей передается не имя типа, а само значение какого- либо типа (листинг 4.38).

Листинг 4.38. Использование функции «ТипЗнч()»

```
ТипСтрока = ТипЗнч{"произвольная
строка"); ТипЧисло = ТипЗнч(387.67);
ТипСсылкаНаСправочникНоменклатура =
                                    ТипЗнч(Справочники.Номенштура.НайтиПоКоду("00000000Г"));
```

Операции со значениями типа «Тип»

Для значений типа Тип определены операции сравнения на равенство и неравенство. Это позволяет выполнять проверку принадлежности произвольного значения некоторому типу (листинг 4.39).

Листинг 4.39. Проверка принадлежности значения типу

```
Сообщение = Новый СообщениеПользователю; Если
ТипЗнч(ПроверяемаяПеременная) = Тип("Строка") Тогда
Сообщение.Текст = "Тип проверяемой переменной - строка";
Сообщение.Сообшиты) КонецЕсли;
```

Преобразование значений типа «Тип»

Значения этого типа могут быть преобразованы только к значениям типа Строка. Такое преобразование может быть выполнено системой неявно, при выводе значения, например, или явно, с использованием встроенной функции Строка(). Результатом такого преобразования будет представление описываемого типа, полученное в соответствии с используемым языком интерфейса платформы (табл. 4.17).

Таблица 4.17. Преобразование к типу «Строка»

Значение	Язык интерфейса платформы	Представление
Тип("Строка")	Русский	Строка
Тип("Строка")	Украинский	Рядок
Тип("Строка")	Латышский	Rinda
Тип("СправочникСсылка.Номенклатура")	Русский	Справочник ссылка: Номенклатура
Тип("СправочникСсылка.Номенклатура")	Казахский	Аныктама сштеме: Номенклатура
Тип("СправочникСсылка.Номенклатура")	Грузинский	(Збго&ьАо ЭобоЭбд&б: Номенклатура

Типы, образуемые в прикладном решении

В отличие от примитивных типов, которые определены на уровне технологической платформы и поддерживаются в любом прикладном решении, прикладные типы создаются в конкретных прикладных решениях в результате добавления в конфигурацию какого-либо объекта метаданных. Прикладные типы создаются платформой автоматически и позволяют работать с данными, хранящимися в тех структурах, которые описываются данным объектом конфигурации.

В зависимости от объекта конфигурации будут добавляться различные типы данных. Например, при добавлении справочника Номенклатура будут созданы такие типы данных, как: СправочникМенеджер.Номенклатура; СправочникСсылка.Номенклатура; СправочникОбъект.Номенклатура; СправочникВыборка. Номенклатура.

Если же в конфигурацию добавить, например, регистр сведений КурсыВалют, то станут доступны следующие типы: РегистрСведенийМенеджер.КурсыВалют; РегистрСведенийВыборка.КурсыВалют; РегистрСведенийМенеджерЗаписи.КурсыВалют; РегистрСведенийНаборЗа- писей. КурсыВалют; РегистрСведенийЗапись .КурсыВалют; РегистрСведенийКлючЗаписи.КурсыВалют.

Как можно заметить из названий, некоторые типы «похожи» (например, СправочникВыборка.Номенклатура и РегистрСведенийВыборка. КурсыВалют), а некоторые нет (например, СправочникОбъект.Номенклатура и РегистрСведенийНаборЗаписей.КурсыВалют). Это действительно так: типы, создаваемые для различных объектов конфигурации и принадлежащие к одной группе прикладных типов (например, <вид объекта>Выборка.<имя>), имеют схожую функциональность, схожее поведение и одинаковые приемы работы с ними.

Такая организация системы прикладных типов существенно облегчает разработку прикладных решений, поскольку если разработчик освоил работу с выборкой справочника, то он уже без труда, по аналогии, сможет работать и с выборкой регистра сведений, и, например, с выборкой бизнес-процесса (табл. 4.18).

Таблица 4.18. Основные типы для работы сданными объектов конфигурации

Объект конфигурации	Менеджер	Выборка	Ссылка	Объект	Набор записей	Запись	Ключ записи
Константа	+						
Критерий отбора	+						
Журнал документов	+	+					
Перечисление	+		+				
Справочник	+	+	+	+			
Документ	+	+	+	+			
План видов характеристик	+	+	+	+			
План счетов	+	+	+	+			
План видов расчета	+	+	+	+			
Бизнес-процесс	+	+	+	+			
Задача	+	+	+	+			
План обмена	+	+	+	+			
Отчет	+			+			
Обработка	+			+			
Последовательность	+				+	+	
Перерасчет	+				+	+	
Регистр сведений	+	+			+	+	+
Регистр накопления	+	+			+	+	+
Регистр бухгалтерии	+	+			+	+	+
Регистр расчета	+	+			+	+	+

Для некоторых объектов конфигурации создаются дополнительные типы, реализующие специальную функциональность, присущую только данным объектам конфигурации (табл. 4.19).

Таблица 4.19. Дополнительные типы для работы с данными объектов конфигурации

Объект конфигурации	Типы
Константа	КонстантаМенеджерЗначения.<имя> КонстантыНабор
План счетов	ПланСчетовВидыСубконто ,<имя> ПланСчетовВидыСубконтоСтрока.<имя>
План видов расчета	ВедущиеВидыРасчета .<имя> ВедущиеВидыРасчетаСтрока .<имя> ВытесняющиеВидыРасчета ,<имя> ВытесняющиеВидыРасчетаСтрока.<имя> БазовыеВидыРасчета .<имя> БазовыеВидыРасчетаСтрока .<имя>
Регистр сведений	РегистрСведенийМенеджерЗаписи.<имя>
Регистр бухгалтерии	РегистрБухгалтерииСубконто.<имя>

Менеджер объектов

Для каждого вида объектов конфигурации (справочники, документы и т.д.) в системе определен тип менеджера этих объектов (СправочникиМе- неджер, ДокументыМенеджер и т.д.). Эти типы существуют всегда, вне зависимости от того, есть ли в конкретном прикладном решении хоть один такой объект или нет.

Каждый из этих типов имеет единственное значение в конкретном прикладном решении. Это можно проиллюстрировать на следующем примере (листинг 4.40).

Листинг 4.40. Сравнение экземпляров объектов менеджеров

```
МенеджерСлравочников1 =
Справочники; МенеджерСправочников2
= Справочники;

Если МенеджерСправочников1 = МенеджерСлравочников2 Тогда
Сообщение = Новый СообщениеПользователю; Сообщение.Текст
= "Объекты равны"; Сообщение,Сообщить(); КонецЕсли;
```

Менеджеры объектов доступны через соответствующие свойства глобального контекста (например, Справочники или Документы) .

Главная задача этого типа - предоставить доступ к менеджерам конкретных объектов конфигурации (СправочникМенеджер.<имя>, ДокументМенеджер.<имя> и т.д.), рис. 4.33.

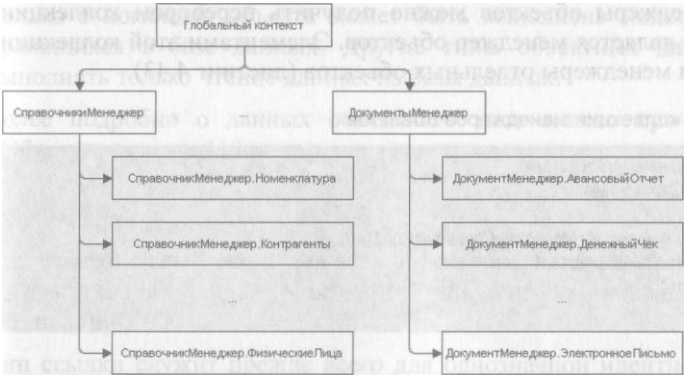


Рис. 4.33. Структура типов

Кроме этого, менеджер объектов позволяет проверить, является ли тип какого- либо значения типом ссылки на объект этого типа (листинг 4.41).

Листинг 4.41. Использование менеджера для проверки принадлежности к типам ссылок

```
ССылкиНаВсеСправочники = Справочники.ТипВсеСсылки();

Сообщение = Новый СообщениеПользователю;
Если ССылкиНаВсеСправочники.СодержитТип(ТипЗнч(АнализируемоеЗначение)) Тогда
    Сообщение.Текст = "Это ссылка на справочник";
Иначе
    Сообщение.Текст = "Значение не является ссылкой на справочник";
КонецЕсли;
Сообщение.Сообщить();
```

В приведенном примере используется метод `ТипВсеСсылкиО` менеджера справочников для того, чтобы в переменной ССылкиНаВсеСправочники получить объект `ОписаниеТипов`, содержащий типы ссылок на все справочники прикладного решения. Затем, используя его метод `СодержитТипО`, проверяется, входит ли тип анализируемого значения в состав этого описания типов или нет.

Менеджеры конкретных объектов конфигурации доступны двумя путями. Во- первых, они доступны как свойства менеджера объектов (листинг 4.42).

Листинг 4.42. Обращение к менеджеру объекта конфигурации

```
МенеджерСправочников = Справочники;
МенеджерСправочникаКонтрагенты = МенеджерСправочников.Контрагенты;
```

Кроме этого, менеджеры объектов можно получить перебором коллекции значений, которой является менеджер объектов. Элементами этой коллекции как раз и являются менеджеры отдельных объектов (листинг 4.43).

Листинг 4.43. Обход коллекции менеджеров объектов

```
Сообщение = Новый СообщениеПользователю;
МенеджерСправочников = Справочники;

Для Каждого МенеджерСправочника Из МенеджерСправочников
Цикл Сообщение.Текст = ТипЗнч(МенеджерСправочника);
Сообщение.Сообщить(); КонецЦикла;
```

Менеджер

Менеджер прикладного объекта можно назвать «точкой входа» в конкретный объект конфигурации в объектной модели встроенного языка. Объекты этого типа предоставляют доступ к общим действиям, относящимся к конкретному объекту конфигурации (например, к справочнику Контрагенты или к документу Авансовый отчет). В основном это:

- получение выборок из данных этого объекта конфигурации;
- создание и поиск элементов данных;
- доступ к предопределенным элементам данных;
- выполнение общих действий, поддерживаемых данным объектом конфигурации;
- получение макетов.

В конкретном прикладном решении всегда существует только один экземпляр каждого менеджера, в чем можно убедиться на следующем примере (листинг 4.44).

Листинг 4.44. Сравнение экземпляров объектов менеджеров объекта конфигурации

```
МенеджерСправочникаКонтрагенты1 = Справочники.Контрагенты;
МенеджерСправочникаКонтрагенты2 = Справочники.Контрагенты;

Если МенеджерСправочникаКонтрагенты1 = МенеджерСправочникаКонтрагенты2 Тогда
Сообщение = Новый СообщениеПользователю; Сообщение.Текст = "Объекты равны";
Сообщение.Сообщить(); КонецЕсли;
```

Объект

Тип объекта, так же как и ссылка, создается не для всех объектов конфигурации, а только для тех, которые хранят в базе данных объектные данные (справочники, документы, планы счетов и пр.).

Только с помощью объекта может быть выполнена модификация данных, хранящихся в базе данных. Другие типы объектных данных позволяют выполнять только чтение данных из базы данных.

Более подробно о данных объектного типа можно прочитать в разделе «Объект», см. стр. 112.

Ссылка

Тип ссылки создается не для всех объектов конфигурации, а только для тех, которые хранят данные объектного типа (справочники, документы, планы счетов и пр.).

Тип ссылки служит прежде всего для однозначной идентификации объекта данных (как совокупности логически связанных данных) в базе данных. Значением ссылки является фактически уникальный внутренний идентификатор, который хранится в поле Ссылка таблиц, создаваемых системой для объекта конфигурации.

Ссылка позволяет обращаться к свойствам объекта базы данных, а также получать сам объект. Более подробно об этом можно прочитать в разделе «Ссылка», см. стр. 106.

Набор записей

Тип набора записей, в отличие от ссылок и объектов, предназначен для работы с необъектными данными, такими как, например, наборы записей различных регистров.

Так же как и объект для объектных данных, набор записей является единственным объектом, с помощью которого в конечном счете выполняется модификация необъектных данных.

Более подробно о необъектных данных можно прочитать в разделе «Необъектные данные», см. стр. 125.

Динамический список

Объект `ДинамическийСписок` предназначен для динамического просмотра данных объекта конфигурации в элементе формы `Таблица`. Динамический список осуществляет считывание данных из базы данных порциями, в процессе навигации пользователя по списку или в процессе обращения к ним из встроенного языка.

С помощью динамического списка построены все формы списка объектов конфигурации, которые содержат основной реквизит типа `ДинамическийСписок`. Динамический список формируется путем запроса к основной

таблице, указанной в соответствующем свойстве реквизита, или путем произвольного запроса к базе данных. Кроме того, объект ДинамическийСписок может быть доступен из встроенного языка.

Динамический список использует систему компоновки данных и позволяет задать группировку, отбор, порядок и условное оформление списка путем установки соответствующих свойств реквизита типа ДинамическийСписок (как интерактивно, так и программно) или путем настройки списка в режиме 1С: Предприятие.

Для примера рассмотрим форму списка документов Расходная накладная, сформированную конструктором форм. В качестве основного реквизита формы платформа добавила реквизит Список типа ДинамическийСписок и заполнила по умолчанию его свойства ОсновнаяТаблица и ДинамическоеСчитываниеДанных (рис. 4.34).

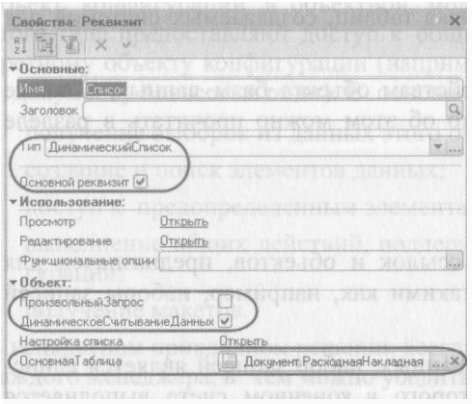


Рис. 4.34. Свойства основного реквизита типа «Динамический список»

Если указана основная таблица, то платформа автоматически будет формировать запрос к этой таблице для получения данных динамического списка. С помощью свойства ДинамическоеСчитываниеДанных динамическому списку указывается на необходимость считывания данных небольшими порциями только при необходимости их отображения на экране.

Раскрыв реквизит Список, можно увидеть поля динамического списка (стандартные и созданные разработчиком реквизиты документа РасходнаяНакладная), а также его свойства: Отбор, Порядок, УсловноеОформление, Группировка и Параметры, доступные для настройки списка как интерактивно, так и программно (рис. 4.35).

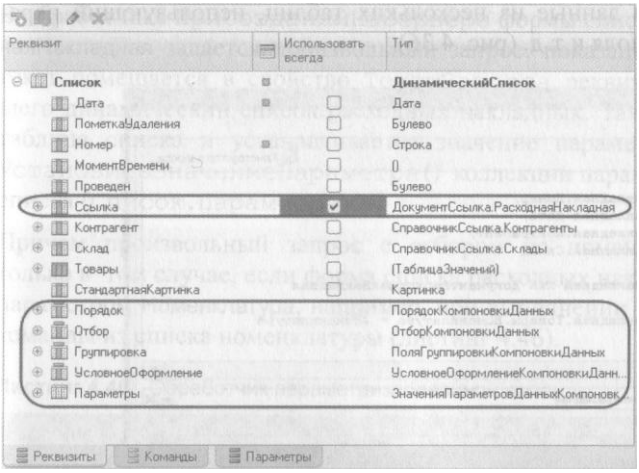


Рис. 4.35. Поля основного реквизита типа «Динамический список»

Обычно далеко не все поля динамического списка используются для отображения в форме. Хотя в списке полей присутствуют все реквизиты документа РасходнаяНакладная, платформа стандартно отображает в форме списка документов только поля Дата и Номер. При необходимости разработчик может перетащить любое поле динамического списка в дерево элементов формы и тем самым отобразить его в форме списка.

Особенность использования динамического списка в том, что запросом из базы данных считываются значения только тех реквизитов, которые отображаются в форме. Это сделано для того, чтобы не передавать на клиента лишнюю информацию и не замедлять работу прикладного решения со списками данных.

Однако если есть необходимость всегда считывать запросом и передавать на клиента значение какого-то поля, то можно включить для него свойство Использовать всегда (см. рис. 4.35) или установить программно признак обязательного использования (`Список.УстановитьОбязательноеИс- пользование("Ссылка",Истина)`). В этом случае указанное поле будет считываться из базы данных и передаваться на клиента, даже если оно не отображается в форме.

Если у динамического списка установить свойство `ПроизвольныйЗапрос`, то станет доступным ручное формирование запроса для получения данных из информационной базы. Открыв окно Настройка списка, можно с помощью конструктора запроса либо вручную создать произвольный запрос к базе

данных, выбирающий данные из нескольких таблиц, использующий параметры, вычисляемые поля и т.д. (рис. 4.36).

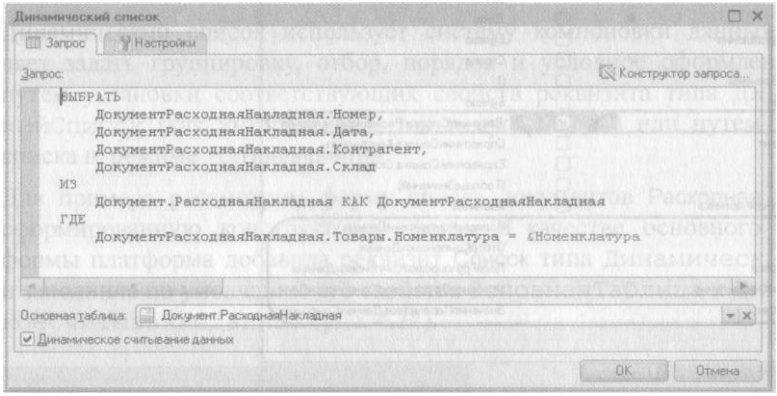


Рис. 4.36. Создание произвольного запроса для динамического списка

Произвольный запрос для динамического списка можно создать и программным образом. Например, требуется открыть список приходных накладных с отбором по номенклатуре (листинг 4.45).

Листинг 4.45. Создание произвольного запроса для динамического списка

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

    Если Параметры.Свойство("Номенклатура") Тогда
        Список.ПроизвольныйЗапрос = Истина;
        Список. ТекстЗалроса = "ВЫБРАТЬ
                    | ДокументРасходнаяНакладная.Номер,
                        ДокументРасходнаяНакладнаяДата,
                    | ДокументРасходнаяНакладная.Контрагент,
                    I ДокументРасходнаяНакладная.Склад
                    |ИЗ
                    | Документ.РасходнаяНакладная КАК
                                                    ДокументРасходнаяНакпадная
                    |ГДЕ
                    | ДокументРасходнаяНакладная.Товары.Номенклатура =
                                                        Номенклатура";

        Список.ОсновнаяТаблица = "Документ.РасходнаяНакладная";
        Список,Параметры.УстановитьЗначениеПараметра("Номенклатура",
                                                    Параметры.Номенклатура)

    КонецЕсли

;
```

В обработчике ПриСозданииНаСервере формы списка документов Расход- наяНакладная задается произвольный запрос, показанный на рис. 4.36. Его текст помещается в свойство ТекстЗапроса реквизита Список, содержащего динамический список расходных накладных. Также задается основная таблица списка и устанавливается значение параметра запроса методом Установить ЗначёниеПараметра () коллекции параметров динамического списка (Список.Параметры).

Причем произвольный запрос с отбором по номенклатуре выполняется только в том случае, если форма списка расходных накладных открывается с параметром Номенклатура, например, при выполнении параметризированной команды из списка номенклатуры (листинг 4.46).

Листинг 4.46. Обработчик параметризированной команды

```
ШаКлиенте
Процедура 0бработкаКоманды(ПараметрКомаиды, ПараметрыВыполненияКоманды)
        ПараметрыФормы = Новый Структура("Номенклатура", ПараметрКоманды);
        ОткрытьФормуС'Документ.РаоходнаяНакладная.ФормаСписка", ПараметрыФормы);
КонецПроцедуры
```

Если форма списка открывается без параметров, например, при открытии списка расходных накладных из панели навигации, то свойства динамического списка останутся без изменения (см. рис. 4.34) и платформой будет автоматически сгенерирован запрос к основной таблице динамического списка, выбирающий данные всей таблицы расходных накладных.

Платформа предоставляет разнообразные и гибкие возможности по настройке динамического списка. Разработчик может настроить группировку, отбор, порядок и условное оформление списка в конфигураторе в окне Настройка списка, нажав в палитре свойств гиперссылку Открыть (рис. 4.37).

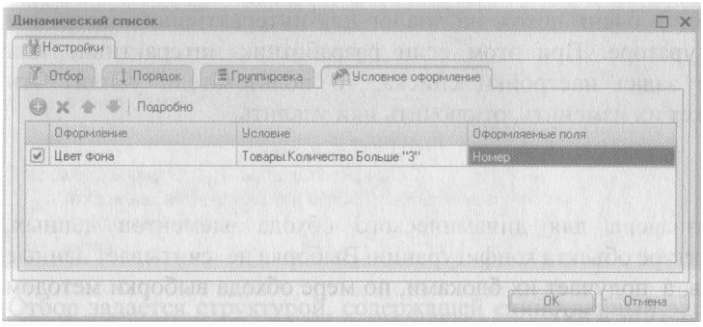


Рис. 4.37. Создание произвольного запроса для динамического списка

На приведенном рисунке установлено, что в списке расходных накладных поле Номер документов, содержащих в табличной части в поле Количество значение больше 3, будет выделено пурпурным цветом фона.

Аналогичного результата можно добиться программным образом (листинг 4.47).

Листинг 4.47. Условное оформление списка

```
&НаКлиенте
Процедура УсловноеОформление(Команда)

    У0 = Список.УсловноеОформление.Элементы;
    ЭлементУО = УО.Добавить();

    II Оформление - цвет фона
    ЭлементУО.Оформление.УстановитьЗначениеПараметра("ЦветФона",
                                                    WebL(ВеТа .Пурпурный);

    II Условие - поле Количество больше трех
    ЭлементУсловия =
        ЭлементУО.Отбор.Элементы.Добавить(Тип("ЭлементОтбораКомпоновкиДанных""
);
ЭлементУсловия.ЛевоеЗначение = Новый ПолеКомпоновкиДанных("Товары. Количество");
ЭлементУсловия.ВидСравнения = ВидСравненияКомпоновкиДанных.Больше;
ЭлементУсловия.ПравоеЗначение = 3;

II Оформляемое поле - Номер
    ОформляемоеПоле = ЭлементУО.Поля.Элементы.Добавить();
    ОформляемоеПоле.Поле = Новый ПолеКомпоновкиДанных("Номер");

КонецПроцедуры
```

Пользователь также может настроить группировку, отбор, порядок и условное оформление динамического списка в режиме 1С:Предприятие, выполнив команду Все действия ► Настроить список... Диалог настройки списка в пользовательском режиме очень похож на диалог для интерактивной настройки списка в конфигураторе. При этом если разработчик интерактивно или программно уже задал настройки списка, то пользователь увидит эти настройки и сможет их изменить, отключить или удалить.

Выборка

Выборка предназначена для динамического обхода элементов данных, хранимых в структуре объекта конфигурации. Выборка не считывает данные целиком в память, а получает их блоками, по мере обхода выборки методом Следующий () (листинг 4.48).

Листинг 4.48. Пример обхода выборки

```
Сообщение = Новый СообщениеПользователю;
ВыборкаВалют =
Справочники.Валюты.Выбрать(); Пока
ВыборкаВалют.Следующий() Цикл
    Сообщение.Текст = ВыборкаВалют.Наименование;
Сообщение.Сообщить(); КонецЦикла;
```

Благодаря тому, что считывание данных осуществляется порциями, выборку можно использовать для обработки очень больших объемов информации.

Каждый блок считываемых данных содержит 25 записей. Для объектных таблиц каждая такая запись содержит все данные объекта, включая все его поля и табличные части. Поэтому, например, при обращении к реквизитам выборки или при получении объекта из выборки не происходит повторного обращения к базе данных.

Выборка осуществляет считывание данных в определенном порядке. Этот порядок может задаваться, например, путем указания параметров Отбор и Порядок метода `Выбрать ()`. Отбор может быть выполнен только по одному полю, причем это может быть только то поле, для которого построен индекс. Кроме полей, которые индексируются системой по умолчанию, это может быть также поле, для которого в конфигураторе установлен признак индексирования в значение Индексировать или Индексировать с дополнительным упорядочиванием. Отбор по таким полям позволяет выполнять выборку быстро и эффективно. Например, приведенный ниже пример (листинг 4.49) позволяет выбрать из справочника Номенклатура только те элементы, для которых не задан артикул.

Листинг 4.49. Пример использования отбора

```
СтруктураОтбора = Новый Структура;
СтруктураОтбора.Вставить("Артикул","")
; Сообщение = Новый
СообщениеПользователю;

ВыборкаНоменклатуры = Справочники.Номенклатура.
                                    Выбрать),, СтруктураОтбора);

Пока ВыборкаНоменклатуры.Следующий() Цикл
    Сообщение.Текст = ВыборкаНоменклатуры.Наименование;
Сообщение.Сообщить(); КонецЦикла;
```

Отбор задается структурой, содержащей единственный элемент, у которого ключ соответствует имени поля, а значение - значению отбора.

Порядок отбираемых записей также может быть задан с использованием полей, которые либо индексируются по умолчанию, либо имеют примитивный тип и для них построение индекса указано в явном виде в конфигураторе. Например, следующий пример (листинг 4.50) позволяет выбрать элементы справочника Договоры контрагентов в порядке убывания даты договора.

Листинг 4.50. Пример использования порядка

```
ВыборкаДоговоров = СправочникиДоговорыКонтрагентов.
                                                    Выбрать(,,, "Дата Убыв");

Сообщение = Новый СообщениеПользователю;
Пока ВыборкаДоговоров.Следующий() Цикл
        Сообщение.Текст = ВыборкаДоговоров.Наименование;
Сообщение.Сообщить(); КонецЦикла;
```

Для иерархических данных (например, справочников, планов видов характеристик, планов счетов) существует также иерархическая выборка, получаемая методом `ВыбратьИерархически()`. Такая выборка отличается от обычной тем, что выдает данные путем обхода записей в соответствии с их иерархией. То есть после выдачи элемента выдается подчиненный ему элемент, если такой существует. Если подчиненных не существует, то выдается следующий элемент. Причем каждое считывание подчиненных элементов реализовано как считывание отдельного блока данных.

Кроме отбора и порядка, выборка позволяет задать также родителя и владельца считываемых данных. При указании конкретного родителя выборка, получаемая методом `Выбрать()`, будет содержать те элементы, родитель которых равен указанному (рис. 4.38).

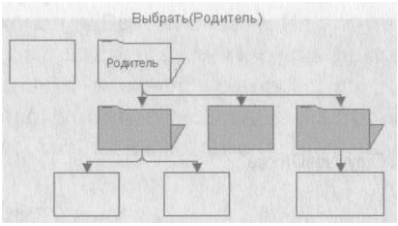


Рис. 4.38. Указание родителя в выборке

В то же время выборка, получаемая методом `ВыбратьИерархическиО`, будет содержать все элементы, находящиеся в иерархии указанного родителя, т. е. в первом, втором и последующих уровнях иерархии (рис. 4.39).

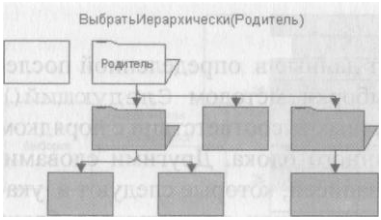


Рис. 4.39. Иерархическая выборка

Если в методе `Выбрать ()` в качестве родителя указать пустую ссылку, то будут выбраны все элементы первого уровня (рис. 4.40).

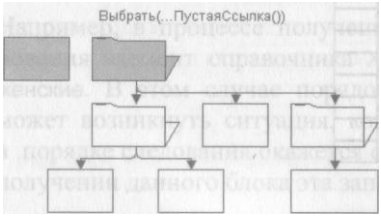


Рис. 4.40. Выборка элементов верхнего уровня

При указании в параметрах выборки конкретного владельца будут отбираться записи подчиненного справочника, владелец которых равен указанному. Например, в следующем примере (листинг 4.51) будут выбраны все договоры контрагента с кодом Ю0004.

Листинг 4.51. Выборка подчиненных элементов

```
Владелец =

Справочники.Контрагенты.НайтиПоКоду("Ю00000004");

ВыборкаДоговоров = Справочники.ДоговорыКонтрагентов.
                                                    ВыбратьИерархически(, Владелец);

Сообщение = Новый СообщениеПользователю;
Пока ВыборкаДоговоров.Следующий() Цикл
        Сообщение.Текст =
```

Особенности использования выборки

Как было показано выше, выборка считывает данные в определенной последовательности сортировки. При обходе выборки методом `Следующий()` выполняется получение очередных блоков данных в соответствии с порядком сортировки и выдачей записей ранее считанного блока. Другими словами, очередной блок считывается как множество записей, которые следуют в указанном порядке после последней полученной записи предыдущего блока (рис. 4.41).

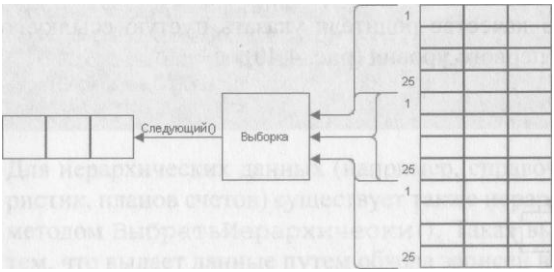


Рис. 4.41. Считывание данных блоками

Особенностью такой схемы работы механизма выборки является то, что разработчик не имеет контроля над актуальностью получаемой информации, даже если выборка инициирована в транзакции. При получении очередной записи нет возможности определить, считывается ли эта запись из уже полученного ранее блока, или же она была только что считана в составе нового блока записей.

Эта особенность работы механизма выборки может приводить к тому, что в выборку могут попасть данные удаленного объекта, некоторые записи могут попасть дважды или не попасть в выборку вообще.

Получение данных удаленного объекта

Допустим, после получения записей очередного блока одна из записей, относящихся к этому блоку, была удалена из базы данных. В этом случае к моменту окончания обхода выборки физически эта запись будет отсутствовать в базе данных, хотя в то же время она будет присутствовать в полученной выборке (рис. 4.42).

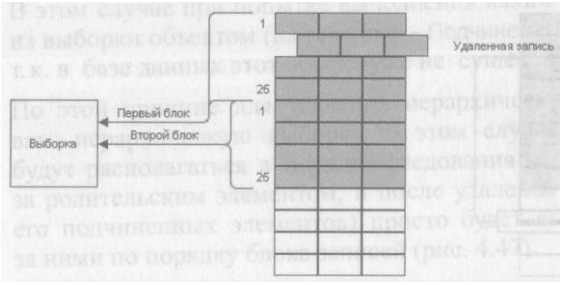


Рис. 4.42. Получение данных удаленного объекта

Множественное попадание записи в выборку

Например, в процессе получения выборки справочника в порядке наименования элемент справочника Женские сапоги был переименован в Сапоги женские. В этом случае порядок следования данной записи изменится, и может возникнуть ситуация, когда переименованный элемент справочника в порядке следования окажется среди записей еще не считанного блока. При получении данного блока эта запись будет считана повторно (рис. 4.43).

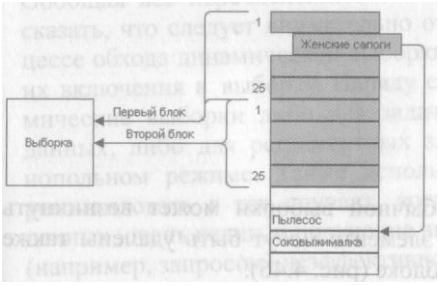


Рис. 4.43. Множественное попадание записи в выборку

Непопадание записи в выборку

Ситуация, когда запись может вообще не попасть в выборку, прямо противоположна предыдущей. Например, в процессе выборки справочника в порядке наименования элемент справочника Сапоги женские, находящийся в порядке следования среди записей еще не считанного блока, был переименован в Женские сапоги и в соответствии с порядком следования оказался среди записей уже считанного ранее блока (рис. 4.44).

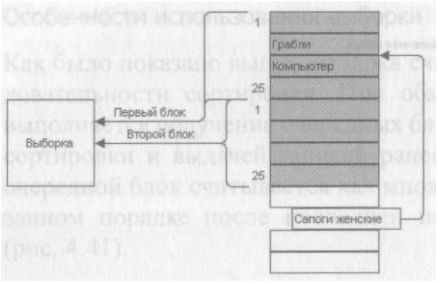


Рис. 4.44. Непопадание записи в выборку

Удаление иерархических данных

Особенность использования выборки для обхода и удаления иерархических данных связана с тем, что при удалении элемента (родителя) удаляются и все его подчиненные элементы (рис. 4.45).

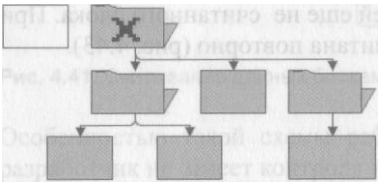


Рис. 4.45. Удаление родителя

Таким образом, при использовании обычной выборки может возникнуть ситуация, когда в результате удаления элемента могут быть удалены также и элементы, уже считанные в текущем блоке (рис. 4.46).

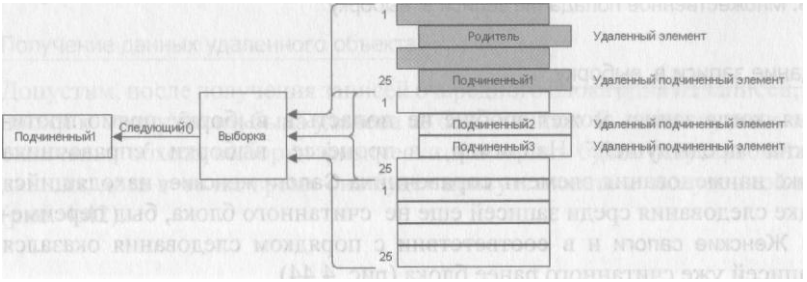


Рис. 4.46. Удаление родителя в процессе выборки

В этом случае при попытке выполнения каких-либо действий с полученным из выборки объектом (на рисунке - Подчиненный1) будет выдаваться ошибка, т. к. в базе данных этот объект уже не существует.
По этой причине для удаления иерархических данных следует использовать иерархическую выборку. В этом случае все подчиненные элементы будут располагаться в порядке следования в одном или нескольких блоках за родительским элементом, и после удаления родителя (и, соответственно, его подчиненных элементов) просто будет выполнено чтение следующего за ними по порядку блока записей (рис. 4.47).

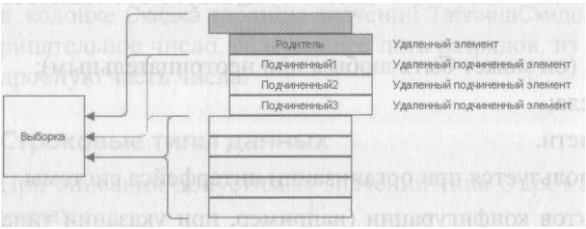


Рис. 4.47. Удаление родителя в процессе иерархической выборки

Обобщая все перечисленные особенности использования выборок, можно сказать, что следует внимательно относиться к изменению объектов в процессе обхода динамической выборки, так как это может повлиять на порядок их включения в выборку. Наряду с этим рекомендуется использовать динамические выборки либо для задач, не требующих ответственного чтения данных, либо для регламентных задач, которые могут выполняться в монопольном режиме. Также использование динамических выборок можно рекомендовать в тех случаях, когда объем данных, выбираемых из базы данных, очень велик, и получение выборки такого объема другими способами (например, запросом) неэффективно.

Типы данных

Тип данных в системе «1С: Предприятие» представляет собой перечень возможных типов значений, которые могут быть использованы тем или иным элементом системы. Типы данных используются при описании таких элементов системы, как:

- объекты конфигурации (реквизиты, измерения, ресурсы и др.);
- реквизиты формы;
- элементы управления, расположенные в форме и в табличном документе;
- некоторые объекты встроенного языка (список значений, колонки таблицы значений и дерева значений и др.).

Тип данных позволяет перечислить типы значений, которые используются, а также для некоторых примитивных типов значений позволяет задать квали- фикаторы, описывающие допустимые значения примитивных типов.

При визуальном конструировании тип данных может быть описан с помощью палитры свойств или в специальном диалоге редактирования типа данных. Во встроенном языке для описания типа данных используется объект встроенного языка `ОписаниеТипов`.

Числовые типы данных

При описании допустимых значений типа `Число` существует возможность указать:

- допустимый знак числа (он может быть любым или неотрицательным);
- общую разрядность числа;
- разрядность дробной части.

Эта возможность часто используется при организации интерфейса системы.
При редактировании объектов конфигурации (например, при указании типа данных реквизита формы) числовой тип данных описывается средствами визуального конструирования (рис. 4.48).

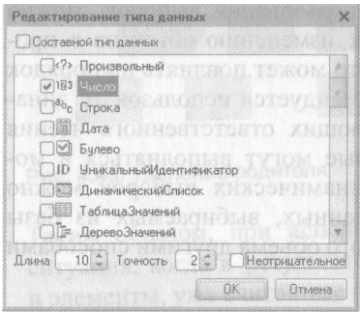


Рис. 4.48. Редактирование типа данных «Число»

Кроме этого, числовой тип данных может быть указан и средствами встроенного языка (например, при создании таблицы значений, которая в дальнейшем будет использована для ввода данных), листинг 4.52. Для этого используется объект `КвалификаторыЧисла`.

Листинг 4.52. Использование квалификаторов числа

```
КвалификаторЧисла =
                Новый КвалификаторыЧисла(5,2, ДопустимыйЗнак.Неотрицательный):
ОписаниеТипа = Новый ОписаниеТилов("Число", КвалификаторЧисла);
```

```
ТаблицаСкидок,Колонки.Добавить("Товар", Новый ОписаниеТипов("СправочникСсылка.Номенклатура"));
ТаблицаСкидок.Колонки.Добавить("Скидка", ОписаниеТипа);
```

В этом примере сначала создается новый объект `КвалификаторыЧисла`, устанавливающий общее количество разрядов - 5, количество разрядов дробной части — 2 и допустимый знак числа - неотрицательный. Затем на основании этого объекта создается новый объект `ОписаниеТипов`, содержащий описание типа `Число` с указанными ограничениями. Далее при добавлении колонки Скидка в таблицу значений ТаблицаСкидок этот объект указывается в качестве описания типа данной колонки. Таким образом, в колонке Скидка таблицы значений ТаблицаСкидок сможет находиться неотрицательное число, содержащее пять разрядов, из которых два отведены под дробную часть числа.

Строковые типы данных

При описании допустимых значений типа `Строка` существует возможность указать:

- какой вариант ограничения длины будет использоваться для строкового значения (фиксированная длина или переменная длина);
- длину строкового значения (в случае фиксированной длины).

В зависимости от длины строки различают строки ограниченной и неограниченной длины. Если длина строки равна нулю, то такая строка имеет неограниченную длину. В противном случае строка будет иметь ограниченную длину. Отличие строк ограниченной и неограниченной длины можно проиллюстрировать следующим примером.

Пусть есть два реквизита: строка длиной 10 (СтрокаОграниченнаяЮ) и строка неограниченной длины (СтрокаНеограниченная). Тогда при присвоении этим реквизитам строкового значения, имеющего длину 14 символов, в первом случае оно будет обрезаться до 10 символов, а во втором случае - нет (листинг 4.53).

Листинг 4.53. Использование строк ограниченной и неограниченной длины

```
Сообщение = Новый СообщениеПользователю;
СтрокаОграниченнаяШ = "новое значение";
Сообщение.Текст = "Строка ограниченной длины (10):" +""" +
                                                    СтрокаОграниченнаяЮ + . ...;
Сообщение.Сообщить();

СтрокаНеограниченная = "новое значение";
Сообщение.Текст = "Строка неограниченной длины:" +""" +
                                                    СтрокаНеограниченная +""";
Сообщение.Сообщить();
```

Результатом выполнения приведенного примера будут следующие сообщения: Строка

ограниченной длины (10): "новое знач" Строка неограниченной длины: "новое значение"

Строки ограниченной длины могут быть двух видов: фиксированной и переменной длины.

Их отличие также лучше всего продемонстрировать на примере.

Пусть есть два реквизита: строка фиксированной длины 10 символов (СтрокаФиксированнаяЮ) и строка переменной длины 10 символов (СтрокаПе- ременнаяЮ). Тогда при присвоении этим реквизитам строкового значения, имеющего длину 5 символов, в первом случае оно будет дополняться пробелами справа до указанной длины, а во втором случае - нет (листинг 4.54).

Листинг 4.54. Использование строк фиксированной и переменной длины

```
Сообщение = Новый СообщениеПользователю; СтрокаФиксированнаяЮ = "новое";
Сообщение.Текст = "Строка фиксированной длины (10):" +""""" +
                                                    СтрокаФиксированнаяЮ +"""";
Сообщение,Сообщить();

СтрокаПеременнаяЮ = "новое";
Сообщение.Текст = "Строка переменной длины (10):"+ ..." +
                                                    СтрокаПеременнаяЮ +"""";
Сообщение.Сообщить();
```

Результатом выполнения приведенного примера будут следующие сообщения: Строка

фиксированной длины (10): "новое Строка переменной длины (10): "новое"

При редактировании объектов конфигурации (например, при указании типа данных реквизита формы) строковый тип данных описывается средствами визуального конструирования (рис. 4.49).

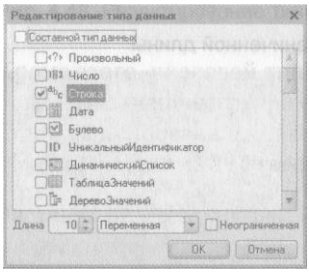


Рис. 4.49. Редактирование типа данных «Строка»

Кроме этого, строковый тип данных может быть указан и средствами встроенного языка. Например, при создании таблицы значений, которая в дальнейшем будет использована для ввода данных (листинг 4.55). Для этого используется объект `КвалификаторыСтроки`.

Листинг 4.55. Использование квалификаторов строки

```
КвашфикаторСтроки = Новый КвзлификаторыСтроки(5, ДолустимаяДлина,Переменная);
ОписаниеТила я Новый ОписаниеТипов("Строка", КвапификаторСтроки);

ТаблицаСкидок.Колонки.Добавить("Товар",
                               Новый ОписаниеТилов("СправочникСсылка.Номенклатура"));
ТаблицаСкидок.Колонки.Добавить(" Артикул", ОписаниеТила);
```

В этом примере сначала создается новый объект `КвалификаторыСтроки`, устанавливающий переменную длину строки в пять символов, затем на основании этого объекта создается новый объект `ОписаниеТипов`, содержащий описание типа `Строка` с указанными ограничениями. Далее при добавлении колонки Артикул в таблицу значений ТаблицаСкидок этот объект указывается в качестве описания типа данной колонки. Таким образом, в колонке Артикул таблицы значений ТаблицаСкидок сможет находиться строка переменной длины, содержащая максимум пять разрядов.

Типы данных, описывающие значения типа «Дата»

При описании допустимых значений типа `Дата` существует возможность указать части даты, которые будут использованы:

ш дата и время,

- дата,
- время.

При редактировании объектов конфигурации (например, при указании типа данных реквизита формы) используются средства визуального конструирования (рис. 4.50).

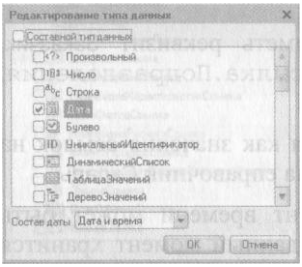


Рис. 4.50. Редактирование типа данных «Дата»

Кроме этого, тип данных, описывающий значения типа Дата, может быть указан и средствами встроенного языка. Например, при создании таблицы значений, которая в дальнейшем будет использована для ввода данных (листинг 4.56). Для этого используется объект КвалификаторыДаты.

Листинг 4.56. Использование квалификаторов даты

```
КвалификаторДаты = Новый КвалификаторыДаты(ЧастиДаты.Дата);
ОписаниеТипаДата = Новый ОписаниеТипов("Дата", КвалификаторДаты);

КвалификаторЧисла = Новый КвалификаторыЧисла(15,2);
ОписаниеТипаЧисло = Новый ОписаниеТипов{"Число", КвалификаторЧисла);

ТаблицаКурсВалют.Колонки.Добавить("Дата", ОписаниеТипаДата);
ТаблицаКурсВалют.Колонки.Добавить("Курс",
ОписаниеТипаЧисло);
```

В этом примере сначала создается новый объект КвалификаторыДаты, который устанавливает использование части даты - Дата. Затем на основании этого объекта создается новый объект ОписаниеТипов, содержащий описание типа Дата с указанными ограничениями. Далее при добавлении колонки Дата в таблицу значений ТаблицаКурсВалют этот объект указывается в качестве описания типа создаваемой колонки. Таким образом, в колонке Дата смогут находиться значения типа Дата, для которых возможно изменение только непосредственно самой даты. Время у этих значений всегда будет равно 00 часов 00 минут 00 секунд.

Составной тип данных

До сих пор мы рассматривали типы данных, содержащие описание только одного типа значений. Однако в общем случае тип данных может содержать перечень нескольких типов значений. Такой тип данных называется *составным.*

Благодаря наличию составного типа в базе данных могут храниться реквизиты, значения которых имеют разный тип в разные моменты времени, однако в один момент времени может храниться значение только одного из перечисленных типов.

Например, документ Внутренний заказ может иметь реквизит Заказчик, имеющий составной тип данных: СправочникСсылка. Подразделения, СправочникСсылка.Склады (рис. 4.51).

Это значит, что в этом реквизите могут храниться как значения ссылок на справочник Подразделения, так и значения ссылок на справочник Склады.

Тип значения такого реквизита в каждый момент времени может быть различным и определяется значением, которое в данный момент хранится в реквизите. Другими словами, если реквизиту присвоить значение ссылки на справочник Подразделения, то тип значения реквизита будет Справоч-

никСсылка.Подразделения; если присвоить ссылку на справочник Склады, то тип значения будет СправочникСсылка.Склады.

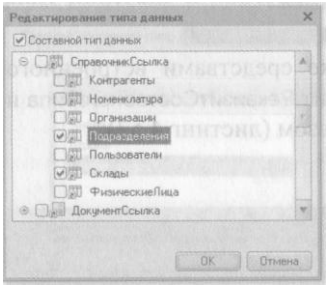


Рис. 4.51. Реквизит составного типа

Описание значений составного типа

Описание составного типа данных может быть выполнено как средствами визуального конструирования, так и средствами встроенного языка.

В процессе визуального описания объектов конфигурации для этого используется окно редактирования типа данных (рис. 4.52).

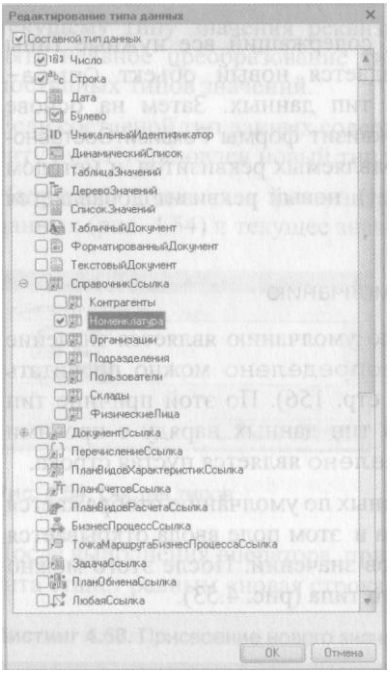


Рис. 4.52. Окно редактирования типа данных

Разработчик может перечислить типы значений и наборы типов, которые должны входить в составной тип данных, а также, при необходимости, задать квалификаторы для числовых, строковых значений и значений типа Дата.

Описывать составной тип данных можно также средствами встроенного языка. Например, у формы можно создать реквизит РеквизитСоставногоТипа и задать его составной тип данных следующим образом (листинг 4.57).

Листинг 4.57. Описание составного типа

```
МассивТипов = Новый Массив;
МассивТилов.Добавить(Тип("Строка"));
МассивТипов.Добавить(Тип("Число"));
МассивТипов.Добавить(Тип("СправочникСсылка.Номенклатура"));
ОписаниеСоставногоТипа = Новый ОписаниеТипов(МассивТипов);

НовыйРеквизит = Новый РеквизитФормы("РеквизитСоставногоТипа",
                                                    ОписаниеСоставногоТипа);
МассивРеквизитов = Новый Массив;
МассивРеквизитов.Добавить(НовыйРеквизит);
ИзменитьРеквизиты(МассивРеквизитов);
```

В этом примере сначала создается массив, содержащий все нужные типы значений, а затем на его основании создается новый объект ОписаниеТипов, который описывает составной тип данных. Затем на основе описания этого типа описывается новый реквизит формы РеквизитСоставногоТипа. После этого формируется массив добавляемых реквизитов, и методом управляемой формы ИзменитьРеквизитыО новый реквизит добавляется в форму.

Значение реквизита составного типа по умолчанию

Для реквизита составного типа значением по умолчанию является значение Неопределено (подробнее о значении Неопределено можно прочитать в разделе «Значение «Неопределено»», см. стр. 156). По этой причине тип Неопределено всегда входит в составной тип данных наряду с прочими типами. Представлением значения Неопределено является пустая строка.

Если значение реквизита составного типа данных по умолчанию отображается в поле ввода, то при нажатии кнопки выбора в этом поле ввода открывается окно для выбора одного из возможных типов значений. После этого можно будет ввести конкретное значение выбранного типа (рис. 4.53).

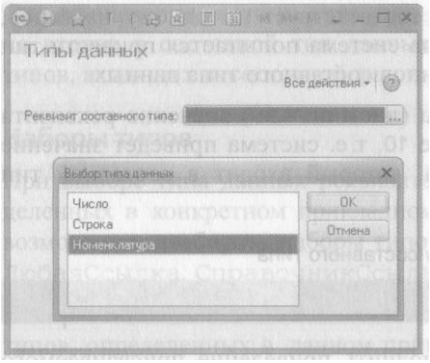


Рис. 4.53. Отображение значения по умолчанию реквизита составного типа

Преобразование значений, хранящихся в реквизитах составного типа Неявное преобразование

При присвоении значений реквизитам, имеющим составной тип данных, могут возникнуть ситуации, когда присваиваемое значение не соответствует текущему типу значения реквизита. В этом случае система будет выполнять неявное преобразование типа присваиваемого значения к одному из доступных типов значений.

Если составной тип данных содержит тип присваиваемого значения, то реквизиту будет установлен новый тип значения и новое значение.

Например, пусть для реквизита СоставнойРеквизит задан составной тип данных (рис. 4.54) и текущее значение реквизита равно `Истина`.

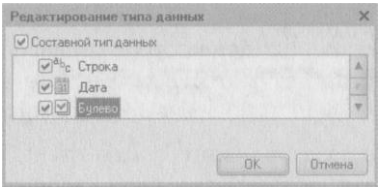


Рис. 4.54. Состав типов

После выполнения оператора, приведенного в листинге 4.58, значение реквизита станет равным «новая строка».

Листинг 4.58. Присвоение нового значения реквизиту составного типа

```
СоставнойРеквизит = "новая строка";
```

Если тип присваиваемого значения не соответствует ни одному из типов, входящих в составной тип данных, тогда система попытается привести тип присваиваемого значения к одному из типов составного типа данных.

Например, после выполнения оператора (листинг 4.59) значение реквизита СоставнойРеквизит станет равным строке 10, т.е. система приведет значение типа Число к значению типа Строка, который входит в составной тип данных.

Листинг 4.59. Присвоение значения реквизиту составного типа

```
СоставнойРеквизит = 10;
```

В случае, если система не сможет выполнить приведение присваиваемого значения ни к одному типу, входящему в составной тип данных, реквизиту будет присвоено значение по умолчанию - Неопределено.

Явное преобразование

Явное преобразование к значению реквизита составного типа данных может быть выполнено с использованием объекта встроенного языка ОписаниеТипов. Этот объект имеет метод ПривестиЗначение(), который позволяет привести значение произвольного типа к значению, соответствующему составному типу данных (листинг 4.60).

Листинг 4.60. Приведение значения

```
МассивТипов = Новый Массив;
МассивТипов.Добавить(Тип("Строка"));
МассивТипов,Добавить(Тип("Число"));
МассивТипов.Добавить(Тип("СправочникСсылка.Номенклатура"));

ОписаниеСоставногоТипа = Новый ОписаниеТипов(МассивТипов);

ПриводимоеЗначение = Справочники.Номенклатура.НайтиПоКоду("00000000Г);

Результат = ОписаниеСоставногоТипа.ПривестиЗначение(ПриводимоеЗначение);
Сообщение = Новый СообщениеПользователю; Сообщение.Текст
="" + ТипЗнч(Результат) +":" + Результат;
Сообщение.Сообщить{);
```

В результате выполнения этого кода будет выведено следующее сообщение:

```
Номенклатура: Сигареты
```

Если несколько видоизменить код и в качестве приводимого значения получить не ссылку на справочник Номенклатура, а объект справочника, то результат будет уже другим:

```
Строка: Сигареты
```

Так происходит потому, что тип СправочникОбъект.Номенклатура не входит в составной тип данных и система приводит его к одному из типов, входящих в составной тип, в данном случае к типу Строка.

Наборы типов

При выборе типа данных реквизита система, помимо выбора типов, определенных в конкретном прикладном решении, предоставляет разработчику возможность выбирать наборы типов. Наборами типов, например, являются
ЛюбаяСсылка, СправочникСсылка, Характеристика.<имя> и др.

Наборы типов, так же как и составной тип данных, содержат некий перечень типов, определенных в данном прикладном решении, однако, в отличие от составного типа, этот перечень формируется системой автоматически, в результате анализа объектов конфигурации.

Проведем сравнение на примере ссылок на справочники. Допустим, в прикладном решении имеются справочники Номенклатура и Контрагенты. Тогда мы можем определить реквизит составного типа данных, в который входят типы СправочникСсылка. Номенклатура и СправочникСсылка. Контрагенты. Наряду с этим мы можем определить реквизит, содержащий набор типов СправочникСсылка. И в том и в другом случае мы будем иметь возможность хранить в реквизите ссылки как на справочник Номенклатура, так и на справочник Контрагенты.

Теперь добавим в конфигурацию новый справочник Цены. В реквизите составного типа мы по-прежнему сможем хранить только ссылки на справочники Номенклатура и Контрагенты, а в реквизите, описанном как набор типов, мы сможем хранить ссылку на любой из справочников, доступных в данной конфигурации, в том числе и на справочник Цены.

При запуске прикладного решения набор типов преобразуется системой, как правило, в составной тип, содержащий все типы, которые должны входить в этот набор. Поэтому во втором случае в набор типов попадет и новый справочник Цены.

Однако набор типов не всегда преобразуется системой в составной тип данных. Если оказывается, что в набор типов входит единственный тип значений, то набор типов будет преобразован в этот самый тип значений. Такая ситуация возможна, например, когда план видов характеристик (назовем его Свойства) имеет единственный тип значений в свойстве ТипЗначе- нияХарактеристик. Тогда набор типов Характеристика.Свойства будет преобразован системой не в составной тип данных, содержащий один тип значений, а в тот единственный тип значений, который указан для плана видов характеристик.

Эта особенность может быть важна, когда, например, выполняется проверка реквизита, тип которого описан как Характеристика.Свойства, на заполненность. В случае, когда Характеристика.Свойства преобразуется системой в составной тип данных, проверять нужно на значение Неопреде- лено, а если Характеристика.Свойства преобразуется в определенный тип значения, то проверять нужно на значение по умолчанию данного типа.

Транзакции

Независимо от выбранного варианта работы (файловый или клиент- серверный) система «1С:Предприятие» обеспечивает работу с информацией, хранящейся в базе данных, с использованием механизма транзакций.

Транзакция - это неделимая с точки зрения воздействия на базу данных последовательность операций манипулирования данными, выполняющаяся по принципу «все или ничего» и переводящая базу данных из одного целостного состояния в другое целостное состояние. Если по каким-либо причинам одно из действий транзакции невыполнимо или произошло какое-либо нарушение работы системы, база данных возвращается в то состояние, которое было до начала транзакции (происходит откат транзакции).

Одним из главных свойств транзакции является ее атомарность, то есть неделимость. Рассмотрим простой пример: интерактивная пометка на удаление группы справочника. Фактически это означает, что нужно выполнить целую последовательность операций:

- для группы справочника установить значение поля ПометкаУдаления;
- записать группу справочника;
- выбрать все элементы справочника, находящиеся в иерархии помечаемой группы;
- для каждого из них установить значение поля ПометкаУдаления;
- выполнить запись каждого из выбранных элементов.

В общем случае нет гарантии, что каждая из перечисленных операций будет успешно выполнена. Например, один из подчиненных элементов справочника может редактироваться другим пользователем, и система в этом случае не сможет изменить данные этого элемента справочника. В результате часть элементов справочника окажется помеченной на удаление, а часть - нет. С точки зрения системы такая ситуация недопустима и нарушает целостность базы данных, т.к. результатом интерактивной пометки на удаление должна быть установка пометки на удаление самого элемента и всех его подчиненных элементов. То есть должны быть выполнены либо все перечисленные операции, либо ни одна из них.

Для того чтобы обеспечить именно такое поведение, установка пометки на удаление выполняется системой в транзакции. Если в процессе выполнения этой последовательности операций произойдет ошибка (например, один из подчиненных элементов будет заблокирован другим пользователем), будет выполнен откат транзакции, и изменения, которые к этому моменту произошли в базе данных, будут отменены (рис. 4.55).

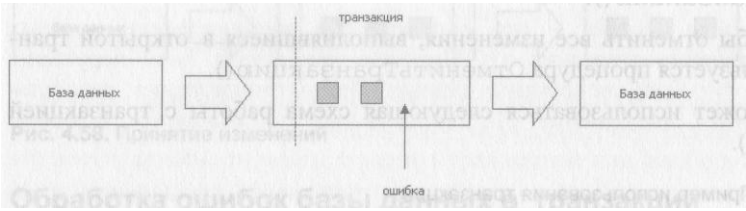


Рис. 4.55. Отказ от изменений, выполненных в транзакции

Таким образом, перечисленная последовательность операций становится неделимой с точки зрения базы данных: изменения будут зафиксированы в базе данных только в том случае, если транзакция успешно завершена, т. е. выполнены все перечисленные операции (рис. 4.56).

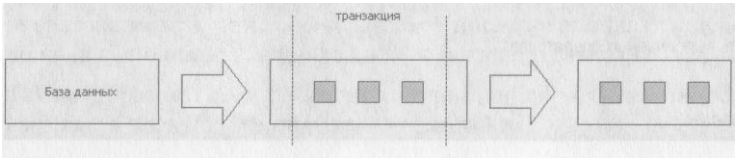


Рис. 4.56. Принятие изменений, выполненных в транзакции

Транзакции могут использоваться как самой системой, так и разработчиком при написании модулей.

Система осуществляет неявный вызов транзакций при выполнении любых действий, связанных с модификацией информации, хранящейся в базе данных. Например, все обработчики событий, расположенные в модулях объектов и наборов записей, связанные с модификацией данных базы данных, вызываются в транзакции.

Наряду с этим разработчик может использовать работу с транзакциями в явном виде. Для этого используются процедуры глобального контекста `НачатьТранзакцию()`, `ЗафиксироватьТранзакциюО` и `ОтменитьТранзакцию()`.

Использование явного вызова транзакций

Процедура НачатьТранзакциюО позволяет открыть транзакцию. После этого все изменения информации базы данных, выполняемые последующими операторами, могут быть либо целиком приняты, либо целиком отвергнуты.

Для принятия всех выполненных изменений используется процедура ЗафиксироватьТранзакцию().

Для того чтобы отменить все изменения, выполнявшиеся в открытой транзакции, используется процедура ОтменитьТранзакцию().

Например, может использоваться следующая схема работы с транзакцией (листинг 4.61).

Листинг 4.61. Пример использования транзакции

```
НачатьТранзакциюО;

// Последовательность выполняемых операторов



II Проверка возможности выполнения операции
Если не <условие> Тогда
        ОтменитьТранзакцию(
);
КонецЕсли;

-- -

ЗафиксироватьТранзакцию{);
```

Суть этой схемы заключается в том, что в процессе выполнения связанной последовательности изменений данных базы данных проверяется возможность выполнения некоторых операций. Если операцию выполнить невозможно, осуществляется откат всех произведенных изменений к состоянию, которое существовало перед выполнением оператора НачатьТранзакциюО (рис. 4.57).



Рис. 4.57. Отмена изменений

Если все операции выполнены успешно, то после выполнения оператора ЗафиксироватьТранзакцию() все произведенные изменения будут зафиксированы в базе данных (рис. 4.58).

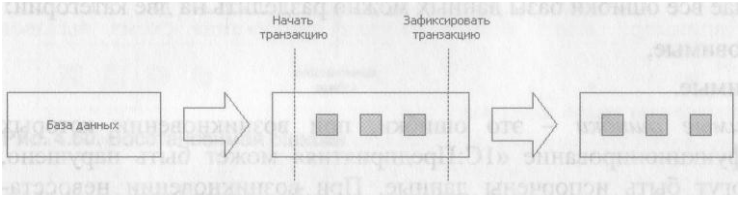


Рис. 4.58. Принятие изменений

Обработка ошибок базы данных в транзакции

Далеко не во всех случаях разработчик заранее может предусмотреть все возможные проверки. По большом счету это и не нужно. Встроенный язык предоставляет разработчику возможность перехватывать и обрабатывать ошибочные ситуации, возникающие в процессе выполнения модуля с помощью конструкции Попытка ... Исключение ... КонецПопытки. В конструкцию Попытка ... КонецПопытки заключаются операторы, при выполнении которых может произойти исключительная ситуация, а после оператора Исключение описывается последовательность действий, которая должна выполняться, если возникает исключительная ситуация (ошибка).

Таким образом, схема работы с транзакцией в более общем виде может выглядеть следующим образом (листинг 4.62).

Листинг 4.62. Обработка исключительных ситуаций

```
НачатьТранзакцию(); II

Последовательность операторов

Попытка
    II Последовательность выполняемых операторов

Исключение
    ОтменитьТранзакцик>(); КонецПопытки;

II Последовательность выполняемых операторов

ЗафиксироватьТрэнзакцию();
```

При использовании такой схемы следует помнить о том, что не все ошибки, возникающие при работе с базой данных, обрабатываются системой одинаково.

В общем случае все ошибки базы данных можно разделить на две категории:

- невосстановимые,
- восстановимые.

Невосстановимые ошибки - это ошибки, при возникновении которых нормальное функционирование «1С:Предприятия» может быть нарушено, например, могут быть испорчены данные. При возникновении невосстановимой ошибки выполнение «1С:Предприятия» прекращается в любом случае.

Если невосстановимая ошибка произошла в процессе выполнения транзакции, то все изменения, сделанные в рамках этой транзакции, отменяются системой (рис. 4.59).

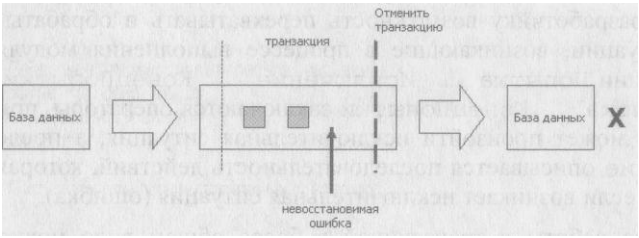


Рис. 4.59. Невосстановимая ошибка

Восстановимые ошибки - это ошибки, не вызывающие серьезных нарушений в работе «1С:Предприятия». В случае возникновения восстановимой ошибки дальнейшая работа системы может быть продолжена. При этом, естественно, сама операция, вызвавшая ошибку, прекращается, и вызывается исключение, которое может быть перехвачено и обработано конструкцией `Попытка ... Исключение ... КонецПопытки`.

Если восстановимая ошибка произошла в процессе выполнения транзакции, то система автоматически не выполняет отмену транзакции, предоставляя разработчику возможность самостоятельно обработать сложившуюся ситуацию (рис. 4.60).

В зависимости от характера произошедшей ошибки возможны различные сценарии обработки этой ситуации.

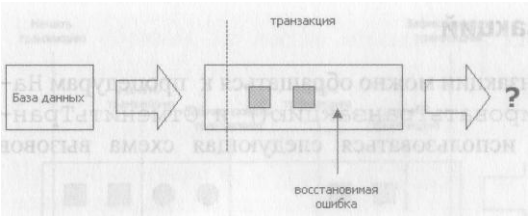


Рис. 4.60. Восстановимая ошибка

Если произошедшая ошибка не связана с базой данных, то возможно продолжение транзакции и дальнейшей работы модуля. Если разработчик считает это необходимым, он может отменить транзакцию или, наоборот, продолжить выполнение транзакции, если произошедшая ошибка не нарушает атомарность транзакции.

Если же исключительная ситуация была вызвана ошибкой базы данных, то система фиксирует факт возникновения ошибки в этой транзакции и дальнейшее продолжение транзакции или ее фиксация становятся невозможны. Единственная операция с базой данных, которую разработчик может произвести в данной ситуации, - это отмена транзакции. После этого он может осуществить попытку выполнения этой транзакции еще раз.

Например, фрагмент кода, реализующий этот подход при записи некоторых данных в базу данных, может выглядеть следующим образом (листинг 4.63).

Листинг 4.63. Обработка исключительных ситуаций

```
// Признак окончания попыток выполнения записи
Записано = Ложь;

// Попытки записи выполняются в цикле
Пока не Записано Цикл

    Попытка
        НачатьТранзакцию();
        Данные.Залисать();
        ЗафиксироватьТранзакцию();

        II в случае фиксации транзакциии прекратить попытки записи
        Записано = Истина;

    Исключение
        II в случае неудачи отменить текущую транзакцию и следующую
    II попытку начать с новой транзакции
    ОтменитьТранзакцию();
    КонецПопытки;

КонецЦикла;
```

Вложенный вызов транзакций

В рамках уже выполняемой транзакции можно обращаться к процедурам `НачатьТранзакцию()`, `ЗафиксироватьТранзакциюО` и `ОтменитьТранзакцию()`. Например, может использоваться следующая схема вызовов (листинг 4.64).

Листинг 4.64. Вложенный вызов транзакций

```
НачатьТранзакцию();

    ' // Вложенный вызов транзакции
        НачатьТранзэкцию();

      ЗафиксироватьТранзакциюО;

    II Вложенный вызов транзакции
        НачатьТранзакцию();

      ЗафиксироватьТранзакциюО;

ЗафиксироватьТранзакциюО;
```

Однако подобное обращение не означает начала новой транзакции в рамках уже выполняющейся. «1С:Предприятие» не поддерживает вложенных транзакций. Это означает, что всегда действует только транзакция самого верхнего уровня. Все транзакции, вызванные внутри уже открытой транзакции, фактически относятся к той же транзакции, а не образуют вложенную транзакцию. Таким образом, отмена изменений, выполняемая во вложенной транзакции, будет приводить в конечном счете не к отмене изменений самой вложенной транзакции (рис. 4.61), а к отмене всех изменений транзакции верхнего уровня (рис. 4.62).

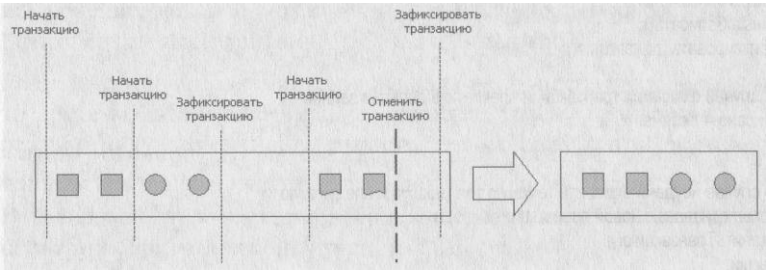


Рис. 4.61. Механика вложенных транзакций

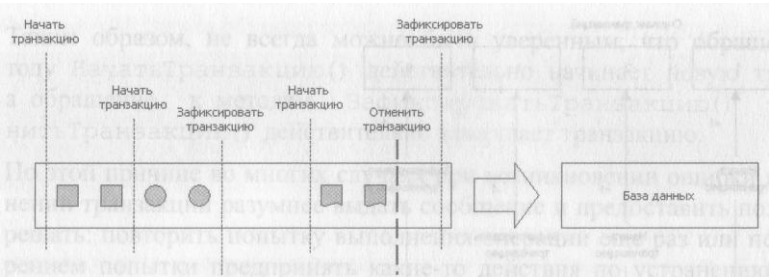


Рис. 4.62. Обработка вложенных вызовов транзакций в «1С:Предприятии»

Рассмотрим подробнее механику вложенных вызовов транзакций, реализуемую «1С:Предприятием».

Когда в рамках выполняемой транзакции происходит вызов процедуры `НачатьТранзакцию()`, фактически выполняется всего лишь увеличение на единицу внутреннего счетчика транзакций. Процедура `НачатьТран- закцию()` действительно начинает новую транзакцию, только если значение внутреннего счетчика транзакций равно нулю (рис. 4.63).

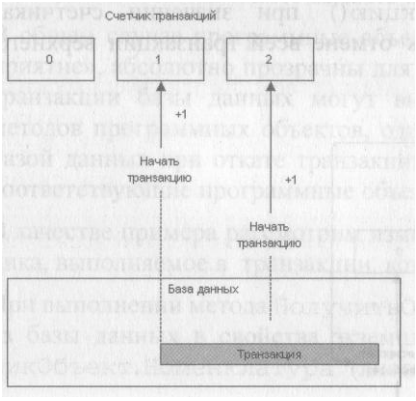


Рис. 4.63. Работа метода «НачатьТранзакциюО»

Обращение к методу `ЗафиксироватьТранзакциюО` приводит к фиксации результата транзакции только в том случае, если значение внутреннего счетчика транзакций равно единице (рис. 4.64).

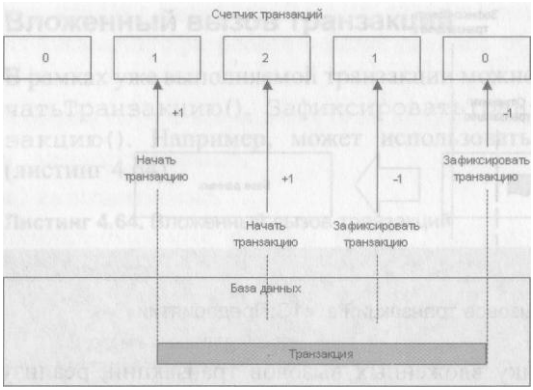


Рис. 4.64. Работа метода «ЗафиксироватьТранзакцию()»

Обращение к методу `ОтменитьТранзакциюО` при значении счетчика транзакций больше единицы приведет не только к уменьшению значения счетчика, но и к установке признака, не позволяющего зафиксировать результаты выполнения всей транзакции в целом. Последующее обращение к процедуре `ЗафиксироватьТранзакциюО` при значении счетчика, равном единице, приведет фактически к отмене всей транзакции верхнего уровня (рис. 4.65).

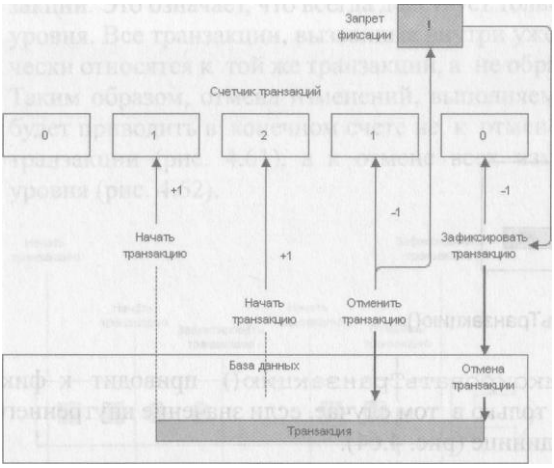


Рис. 4.65. Работа метода «ОтменитьТранзакциюО»

Таким образом, не всегда можно быть уверенным, что обращение к методу НачатьТранзакцию() действительно начинает новую транзакцию, а обращение к методам ЗафиксироватьТранзакцию() и Отме- нитьТранзакцию() действительно завершает транзакцию.

По этой причине во многих случаях при возникновении ошибки при выполнении транзакции разумнее выдать сообщение и предоставить пользователю решать: повторить попытку выполнения операции еще раз или перед повторением попытки предпринять какие-то действия по устранению ситуации, приведшей к ошибке.

Влияние транзакций на работу программных объектов

Как уже отмечалось выше, механизм транзакций обеспечивает атомарность и согласованность изменений, выполняемых в базе данных. Работа с данными базы данных осуществляется в системе «1С:Предприятие» посредством различных программных объектов, которые используются системой непосредственно, а также доступны во встроенном языке.

Результат такого опосредованного взаимодействия не всегда является очевидным и иногда может вызвать непонимание.

В общем случае программные объекты, используемые системой «1С:Пред- приятие», абсолютно прозрачны для транзакций базы данных. Иначе говоря, транзакции базы данных могут вызываться при выполнении различных методов программных объектов, однако, например, действия, выполняемые базой данных при откате транзакции, в общем случае никак не влияют на соответствующие программные объекты.

В качестве примера рассмотрим изменение наименования элемента справочника, выполняемое в транзакции, которая затем отменяется.

При выполнении метода ПолучитьОбъект () данные объекта будут считаны из базы данных в свойства экземпляра программного объекта Справоч- никОбъект.Номенклатура (листинг 4.65).

Листинг 4.65. Получение объекта справочника

```
Товар = справочники.номенклатура,найтипокоду("000000001").получить0бъект();
```

Таким образом, и в базе данных, и в свойстве программного объекта будет находиться одно и то же значение наименования - Старое наименование (рис. 4.66).

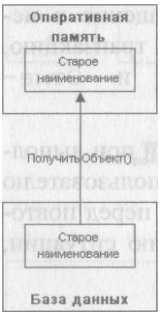


Рис. 4.66. Получение объекта справочника

После этого вызывается транзакция и изменяется значение свойства Наименование программного объекта на Новое наименование (листинг 4.66).

Листинг 4.66. Изменение наименования элемента справочника в транзакции

```
Товар = Справочники.Номенклатура.НайтиПоКоду("ОООООООО").ПолучитьОбъе1ст();

НачатьТранзакцию();
        Товар.Наименование = "Новое наименование";
```

Теперь наименования в базе данных и в свойстве программного объекта отличаются (рис. 4.67).



Рис. 4.67. Изменение наименования элемента справочника в транзакции После этого

выполняется запись программного объекта (листинг 4.67).

Листинг 4.67. Запись объекта справочника

```
Товар = Справочники.Номенклатура.НайтиПоКоду("000000001 ").Получить0бъект();

НачатьТранзакцию{};
        Товар,Наименование = "Новое наименование";
        Товар. ЗаписатьО;
```

После успешной записи данных программного объекта в базу данных наименование элемента справочника и в памяти, и в базе данных будет одинаковым: Новое наименование (рис. 4.68).

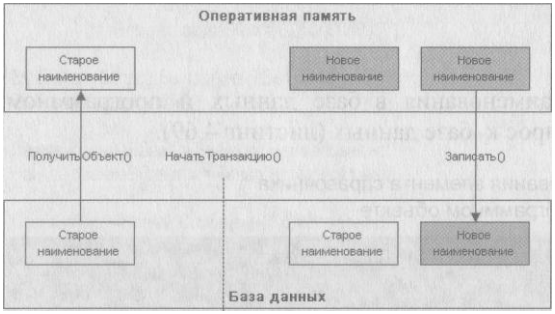


Рис. 4.68. Успешная запись объекта справочника

Затем выполняется отмена транзакции (листинг 4.68). **Листинг**

4.68. Отмена транзакции

```
Товар = Справочники.Номенклатура.НайтиПоКоду("000000001").ПолучитьОбъектО;

НачатьТранзакцию();
        Товар.Наименование ? "Новое наименование";
Товар.Записать(); ОтменитьТранзакциюО;
```

В результате того, что изменения, выполненные в базе данных в транзакции, не принимаются, состояние базы данных возвращается в исходное, т.е. то, которое было до начала транзакции. Это значит, что наименование элемента справочника снова принимает значение Старое наименование.

Однако эти изменения никоим образом не касаются программного объекта: значение свойства Наименование остается равным Новое наименование (рис. 4.69).

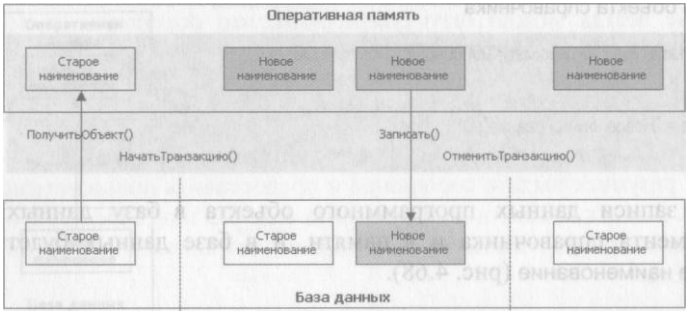


Рис. 4.69. Отмена транзакции

Для того чтобы сравнить наименования в базе данных и программном объекте, можно выполнить запрос к базе данных (листинг 4.69).

Листинг 4.69. Сравнение наименования элемента справочника в базе данных и программном объекте

```
Товар = Справочники. Номенклатура. НайтиПоКоду("ОООООООООГ).ПолучитьОбъект();

НачатьТранзакцию();
        Товар.Наименование = "Новое
наименование";
Товар.Записать();
ОтменитьТранзакцию();

Запрос = Новый Залрос("ВЫБРАТЬ
        | Номенклатура.Наименование
        |ИЗ
        | Справочник.Номенклатура КАК Номенклатура
        (ГДЕ
        | Номенклатура.Код = "000000001""");

Выборка = Запрос.Выполнить().Выбрать();
Выборка.Следующий();

НаименованиеВБазе = Выборка.Наименование:
НаименованиеВПамяти = Товар.Наименование;

Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = НаименованиеВПамяти;
Сообщение.Сообщить();
Сообщение.Текст = НаименованиеВБазе;
Сообщение.Сообщить();
```

Если теперь заново прочитать данные объекта из базы данных в программный объект, то значение наименования в программном объекте станет таким же, как и в базе данных (листинг 4.70).

Листинг 4.70. Перечитывание данных объекта

```
Товар = Справочники.Номенклатура.НайтиПоКоду("ОООООООО").ПолучитьОбъект();

НачатьТранзакцию();
        Товар.Наименование = "Новое
наименование"; Товар.Записать();
ОтменитьТранзакциюО;

Запрос = Новый Запрос("ВЫ БРАТЬ
        | Номенклатура.Наименование [ИЗ
        j Справочник.Номенклатура КАК Номенклатура
        ГДЕ
        | Номенклатура.Код = ""000000001""");

Выборка = Запрос.Выполнить{}.Выбрать(); Выборка.Следующий();

НаименованиеВБазе = Выборка.Наименование; НаименованиеВПамяти =
Товар.Наименование;

Сообщение = Новый СообщениеПользователю; Сообщение.Текст =
НаименованиеВПамяти; Сообщение.Сообщить(); Сообщение.Текст = НаименованиеВБазе;
Сообщение.Сообщить();

Товар.Прочитать();
Сообщение.Текст | Товар.Наименование; Сообщение.Сообщить();
```

Из приведенного примера следует важный вывод: при отмене транзакций базы данных разработчик (если в этом есть необходимость) должен самостоятельно обеспечивать адекватное изменение данных соответствующих программных объектов. Это можно выполнять путем перечитывания всех данных объекта или путем изменения некоторых реквизитов программного объекта (если, например, это необходимо для отображения в интерфейсе).

Однако, как в любом правиле, здесь тоже есть исключения. В силу значительной прикладной специфики программных объектов «1С:Предприятия» в некоторых случаях откат изменений, выполненных в базе данных, все же может влиять на значения свойств соответствующих программных объектов.

Восстановление признака проведенности документа

При отмене транзакции признак проведенности документа восстанавливает значение, которое было до начала транзакции (листинг 4.71).

Листинг 4.71. Восстановление признака проведенное™ документа

```
НачатьТранзакциюО;
        Накладная = Документы.ПриходнаяНакладная.СоздатьДокумент();
Накладная.Дата = ТекущаяДатаО;
Накладная.Записать(РежимЗаписиДокумента.Проведение);
Проведен = Накладная.Проведен;
ОтменитьТранзакциюО;

Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = Строка(Накладная.Проведен);
Сообщение.СообщитьО; II Нет
Сообщение.Текст = Строка(Проведен);
Сообщение.СообщитьО; II Да
```

Очистка ссылки

Если объект был создан в транзакции, то при ее откате очищается значение ссылки (листинг 4.72).

Листинг 4.72. Очистка ссылки

```
НачатьТранзакциюО;
        Товар = Справочники.Номенклатура.СоздатьЭлементО;
Товар.Записать();
Ссылка я Товар.Ссылка;
ОтменитьТранзакциюО;

Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = ?(Товар.Ссылка = Справочники.Номенклатура.ПустаяСсылка(), "Истина", "Ложь");
Сообщение. СообщитьО; //истина
Сообщение.Текст = Строка(Ссылка);
Сообщение.СообщитьО; II <0бъект не найден> (1:9e4b00055d4c?bcf11d934028f79e857)
```

Очистка кода/номера объекта

Если объект создавался вне транзакции и при записи его в транзакции использовался код/номер, сгенерированный автоматически, то при отмене транзакции код/номер очищается (листинг 4.73).

Листинг 4.73. Очистка кода

```
Товар = Справочники,Номенклатура.СоздатьЭлементО;

   НачатьТранзакциюО;
        Товар.ЗаписатЦ);
        Код = Товар.Код;
 ОтменитьТранзакциюО
                    ;

Сообщение = Новый СообщениеПользователю;
Сообщение.Текст = Товар.Код;
Сообщение.СообщитьО; //""
Сообщение.Текст = Код;
Сообщение.СообщитьО; II "000000008"
```

Механизм управляемых блокировок

Возможные проблемы при многопользовательском доступе к одним и тем же данным

Работа в многопользовательской среде требует соблюдения определенного компромисса между требованиями предсказуемости, целостности и непротиворечивости данных информационной базы и требованиями параллельности работы.

Как известно, при одновременном чтении и изменении одних и тех же данных конкурирующими транзакциями могут возникнуть следующие проблемы одновременного доступа:

- проблема потерянного изменения,
- проблема «грязного» чтения,
- проблема неповторяемого чтения,
- проблема чтения фантомов. Рассмотрим подробнее

каждую из этих проблем.

Проблема потерянного изменения (англ. The Lost Update Problem) возникает в случае, если две транзакции изменяют одни и те же данные, взяв в качестве первоисточника начальное значение этих данных. При этом в системе останутся изменения, внесенные той транзакцией, которая записала свои изменения последней, поскольку эти изменения заменят собой все изменения, внесенные до этого.

Рассмотрим следующий пример. Допустим, Транзакция № 1 обратилась к элементу справочника Номенклатура и решила изменить значение реквизита товара ЦенаПродажи с 420 на 450. Одновременно Транзакция № 2 решила у этого же товара изменить значение реквизита ЕдиницаИзмерения со Штука на Коробка. Распределение по времени описанных действий показано на рис. 4.70. Таким образом, в элементе справочника остались только те изменения, которые сделала Транзакция № 2.

Из примера следует вывод: нельзя одновременно изменять одни и те же данные.

Проблема «грязного» чтения (англ. The Uncommitted Dependency Problem) возникает в случае, если одна транзакция начинает считывать некоторые данные, не дождавшись окончания внесения изменений, вносимых в эти данные другой транзакцией. При этом достаточно вероятен случай, когда прочитанные данные будут содержать неверную информацию.

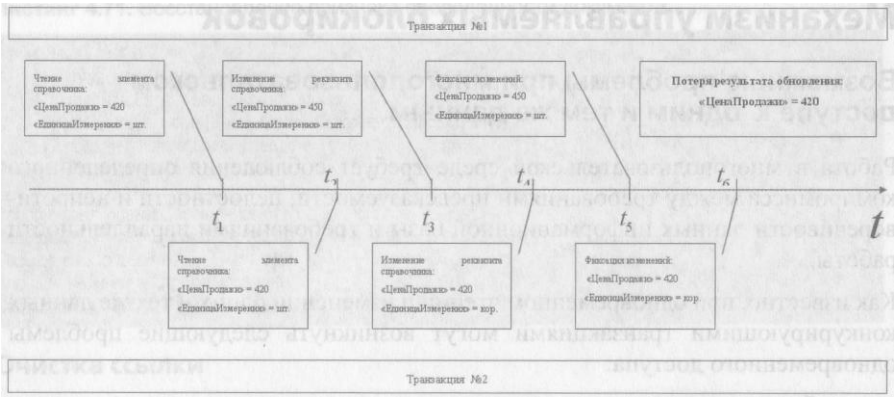


Рис. 4.70. Иллюстрация проблемы потерянного изменения

Вернемся к примеру, рассмотренному выше. Допустим, Транзакция № 1 обратилась к элементу справочника Номенклатура и изменила значение реквизита товара ЦенаПродажи с 420 на 450. Не дождавшись фиксации изменений, Транзакция № 2 использовала значение реквизита этого товара для определения суммы продажи. Однако первая транзакция решила не сохранять внесенные изменения (откат транзакции) и восстановила старые данные. Графическое представление действий транзакций показано на рис. 4.71. Таким образом, Транзакция № 2 в своих расчетах использовала данные, не существующие в системе.

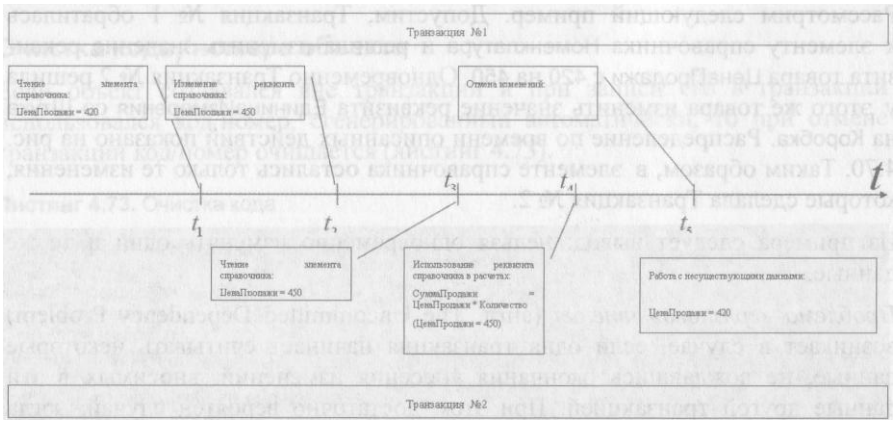


Рис. 4.71. Иллюстрация проблемы «грязного» чтения

Из примера следует вывод: нельзя читать уже измененные, но еще не записанные данные.

Проблема неповторяемого чтения (англ. The Inconsistent Analysis Problem) возникает в случае, если одна транзакция несколько раз считывает одни и те же данные, а вторая вносит изменения в эти данные между циклами чтения данных первой транзакции. При этом при повторном считывании первая транзакция может получить другой набор данных.

Допустим, в нашем примере Транзакция № 1 два раза подряд обращается к элементу справочника Номенклатура и каждый раз считывает значение реквизита ЦенаПродажи. Если в промежуток между первым и вторым чтением вклинится Транзакция № 2 и изменит значение этого реквизита, то в результате получится, что первая транзакция работает с данными, которые с ее точки зрения самопроизвольно изменяются. Графическое представление данной проблемы показано на рис. 4.72.

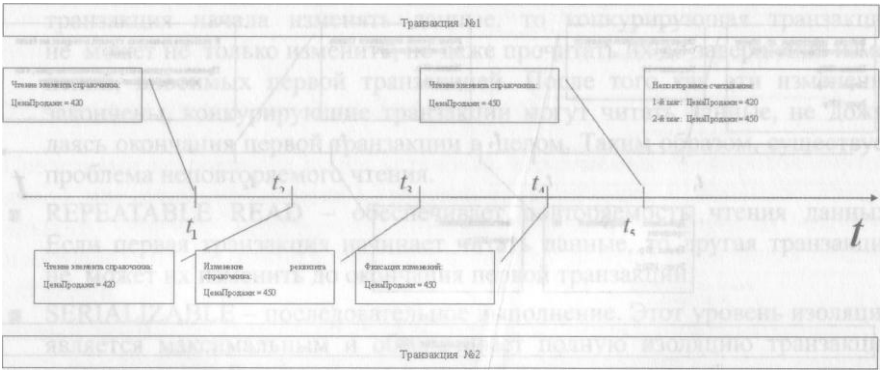


Рис. 4.72. Иллюстрация проблемы неповторяемого чтения

Из примера следует вывод: нельзя повторно читать измененные и записанные данные, если эти же самые данные уже были прочитаны до внесения в них изменений.

Проблема чтения фантомов (англ. The Phantom Read Problem) возникает в случае, если первая транзакция считывает данные и потом на их основе осуществляет определенные действия, а вторая транзакция в этот момент добавляет в эти данные новую информацию. При этом, как и в предыдущем случае, это может привести к некорректному результату.

Рассмотрим следующий пример. Допустим, компания занимается продажей товаров и состоит из нескольких отделов. Если объем продаж сотрудников

одного отдела превышает 1 ООО рублей, то каждый сотрудник отдела получает премию 20 % от суммы своих продаж. В противном случае размер премии составляет 10 %. Очевидно, что процесс начисления премии сотрудникам каждого отдела будет состоять из нескольких операций:

- получения общей суммы продаж по отделу в целом путем суммирования отдельных продаж по каждому из сотрудников;
- определения на основании полученных данных процента премии;
- расчета суммы премии для каждого из сотрудников отдела. Предположим, что данные о продажах вводит Транзакция № 2, а размер премии рассчитывает Транзакция № 1. Тогда при одновременной работе транзакций может возникнуть ситуация, показанная на рис. 4.73. Таким образом, Транзакция № 1 в двух одинаковых выборках строк получила разные результаты.

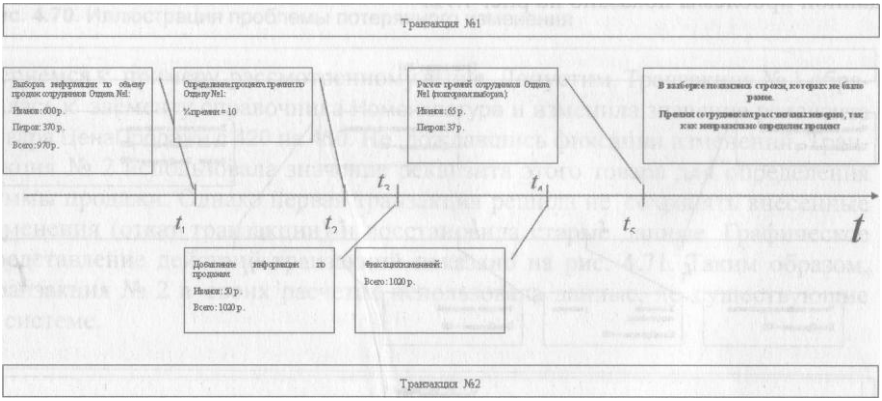


Рис. 4.73. Иллюстрация проблемы чтения фантомов

Из примера следует вывод: нельзя вводить новые данные (удалять имеющиеся), если они могут попасть в уже один раз прочитанные данные при повторном чтении.
Строго говоря, список вышеперечисленных проблем не является окончательным.

Уровни изоляции транзакций

Итак, ради увеличения производительности системы необходимо разрешить параллельное выполнение транзакций. При этом также нужно обеспечить необходимую степень целостности данных (то есть ограничить параллельность транзакций при работе с одними ресурсами). Строгость

этих ограничений может быть различной, в зависимости от решаемой задачи. Поэтому платформе необходим механизм гибкой настройки этих ограничений. В современных СУБД такая возможность реализуется путем применения уровней изоляции транзакций. СУБД, с которыми может взаимодействовать «1С:Предприятие», позволяют использовать следующие уровни изоляции транзакции:

- READ UNCOMMITED - незавершенное чтение. Низший уровень изоляции, обеспечивает максимальную параллельность выполнения транзакций. Данный уровень защищает данные, изменяемые одной транзакцией, от изменений, которые могут внести конкурирующие транзакции. Если другой транзакции необходимо изменить те же самые данные, то она должна ожидать завершения изменения данных первой транзакцией. Однако чтение данных разрешено. Таким образом, этот уровень изоляции допускает чтение незавершенных изменений данных.
- READ COMMITED - обеспечивает запрет «грязного» чтения. Если одна транзакция начала изменять данные, то конкурирующая транзакция не может не только изменить, но даже прочитать их до завершения изменений, вносимых первой транзакцией. После того как эти изменения закончены, конкурирующие транзакции могут читать данные, не дожидаясь окончания первой транзакции в целом. Таким образом, существует проблема неповторяемого чтения.
- REPEATABLE READ - обеспечивает повторяемость чтения данных. Если первая транзакция начинает читать данные, то другая транзакция не может их изменить до окончания первой транзакции.
- SERIALIZABLE — последовательное выполнение. Этот уровень изоляции является максимальным и обеспечивает полную изоляцию транзакций друг от друга. Решаются все рассмотренные выше проблемы, включая проблему фантомов.

В зависимости от используемого уровня изоляции СУБД накладывает различные типы блокировок на различные объекты базы данных на различное время.

Режим управляемых блокировок

Система «1С:Предприятие» стандартно поддерживает управляемый режим блокировок данных в транзакции. При работе в этом режиме используется достаточно низкий уровень изоляции транзакций в СУБД, позволяющий достичь высокой параллельности работы пользователей (табл. 4.20).

Таблица 4.20. Блокировки СУБД, используемые в режиме управляемых блокировок в транзакции

	СУБД			
	Файловая база данных	MS SQL Server	IBM DB2	PostgreSQL
Вид блокировок	Таблиц	Записей	Записей	Записей
Уровень изоляции транзакций	Serializable	Read Committed	Read Committed	Read Committed

Однако этот уровень изоляции транзакций СУБД уже не может сам по себе обеспечить целостность и непротиворечивость данных во всех случаях. Поэтому при модификации данных методами встроенного языка (например, методом `Записать` () у объектных данных) платформа устанавливает собственные управляемые блокировки в транзакции, которые обрабатываются собственным менеджером транзакционных блокировок (рис. 4.74).



Рис. 4.74. Управляемые блокировки в транзакции «1С:Предприятия 8.2»

Эти блокировки также могут быть установлены и разработчиком самостоятельно в тех местах кода, где требуется обеспечить неизменность считываемых в транзакции данных (разделяемая блокировка) или запретить чтение данных другими транзакциями (исключительная блокировка).

Управляемые блокировки «1С:Предприятия» учитывают логическую структуру прикладного решения, поэтому позволяют максимально точно блокировать необходимые области данных. Таким образом, менеджер управляемых блокировок позволяет максимально избежать возникновения «плохих» (избыточных) блокировок. Блокируются только действительно необходимые области данных.

В результате любой запрос к данным прежде всего обрабатывается собственным менеджером транзакционных блокировок «1С:Предприятия» (см. рис. 4.74). Если на уровне «1С:Предприятия» конфликт управляемых блокировок не обнаруживается, то запрос передается далее, на исполнение СУБД. СУБД также использует собственный механизм блокировок для определения конфликтующих транзакций, с низким уровнем изоляции транзакций, указанным в таблице 4.20.

Установка режима управления блокировкой данных в конфигурации

Режим блокировок в конфигурации устанавливается с помощью свойства `Режим управления блокировкой данных`. Это свойство имеет как конфигурация в целом, так и каждый прикладной объект конфигурации.

Для платформы «1С:Предприятие» стандартным является режим управляемых блокировок. Поэтому свойство конфигурации `Режим управления блокировкой данных` для новых конфигураций устанавливается в значение `Управляемый`.

Установка управляемых блокировок

Средствами встроенного языка установка управляемых блокировок внутри явной или скрытой (неявной) транзакции происходит с помощью специального объекта `БлокировкаДанных`, описание доступных свойств и методов которого можно посмотреть в синтакс-помощнике, в ветви `Общие объекты` (рис. 4.75).

Новый экземпляр данного объекта может быть создан с помощью одноименного конструктора и представляет собой коллекцию элементов блокировки данных. Изначально эта коллекция пуста, и задача разработчика состоит в добавлении в эту коллекцию некоторого количества элементов блокировки.

При добавлении нового элемента блокировки для него необходимо указать пространство блокировок, которое будет блокировать данный элемент. Пространства блокировок определены в платформе «1С:Предприятие» и соответствуют структуре прикладных объектов конфигурации. Допустимы следующие имена пространств блокировок и имена полей пространств блокировок (табл. 4.21).

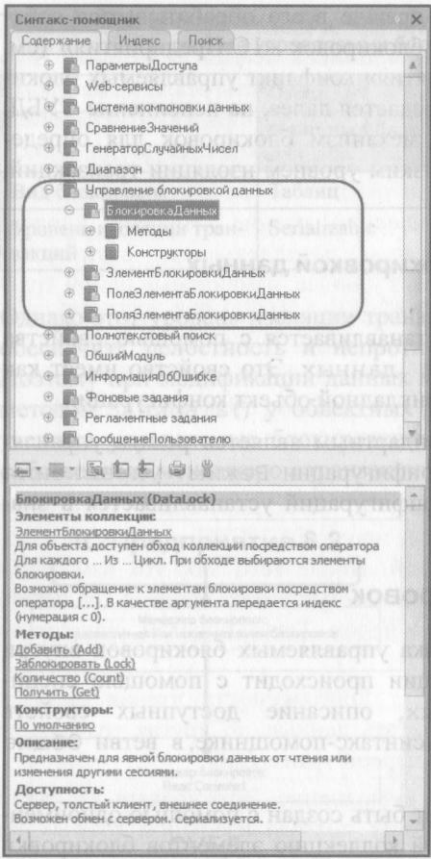


Рис. 4.75. Набор свойств и методов объекта «БлокировкаДанных», доступных в синтакс-помощнике

Таблица 4.21. Имена пространств блокировок и имена полей пространств блокировок

Имя пространства блокировки	**Имя поля пространства блокировки**
Справочник.<имя>	Ссылка
Документ.<имя>	Ссылка
ПланОбмена.<имя>	Ссылка
ПланСчетов.<имя>	Ссылка
БизнесПроцесс.<имя>	Ссылка
Задача.<имя>	Ссылка

Имя пространства блокировки	Имя поля пространства блокировки
ПланВидовРасчета.<имя>	Ссылка
ПланВидовХарактеристик.<имя>	Ссылка
РегистрСведений.<имя>.НаборЗаписей - только для регистра сведений, подчиненного регистратору	Регистратор
РегистрСведений.<имя>	Период - если есть; <имя измерения>
РегистрНакопления.<имя>.НаборЗаписей	Регистратор
РегистрНакопления.<имя>	Период; <имя измерения>
РегистрБухгалтерии.<имя>.НаборЗаписей	Регистратор
РегистрБухгалтерии.<имя>	Период; <имя измерения>; <вид движения> — значение системного перечисления ВидДвиженияБухгалтерии; Счет; Субконто<1Ч>; <вид субконто>
РегистрРасчета.<имя>.НаборЗаписей	Регистратор
РегистрРасчета.<имя>	ПериодРегистрации; ПериодДействия; <имя измерения>
Перерасчет. <имя>.НаборЗаписей	ОбъектПерерасчета
Последовательность.<имя>.НаборЗаписей	Регистратор
Последовательность.<имя>	<имя измерения>
Константа.<имя>	

Как видно из таблицы, для объектных данных (справочник, документ и др.) определено единственное пространство блокировки - сам объект данных. Для необъектных данных (например, регистры) определено по два пространства блокировок, которые имеют разный логический смысл.

Пространство блокировок с суффиксом `НаборЗаписей` используется в тех случаях, когда необходимо заблокировать сами записи данного объекта (например, при добавлении новых записей).

Пространство блокировок без суффикса используется, когда анализируются некоторые данные этого объекта (например, остатки регистра) или когда выполняются какие-либо операции, приводящие к изменению существующих данных объекта (например, восстановление границы последовательности).

После того как элемент блокировки, соответствующий некоторому пространству блокировок, добавлен, следует установить для этого элемента режим блокировки (разделяемая или исключительная) и определить значения полей блокировки, чтобы указать, какие же именно «записи» будут забло

кированы (для каждого пространства блокировок в платформе определены имена полей, значения которых могут задаваться при установке тех или иных блокировок).

Следует понимать, что в данном случае речь не идет о реальных записях базы данных. Несмотря на то, что управляемые блокировки описываются в терминах объектов метаданных и их полей, эти блокировки никак не связаны с реальной структурой хранения данных «1С:Предприятия» в СУБД. Это всего лишь записи о том, что заблокировано «нечто».

Иногда можно провести аналогию между управляемыми блокировками и реальными записями СУБД. Например, для объектных данных блокировка объекта с указанной ссылкой будет «соответствовать» блокировке всех записей, содержащих указанную ссылку, во всех таблицах этого объекта конфигурации (в основной таблице и в таблицах его табличных частей).

Однако в других случаях провести такую аналогию достаточно затруднительно, да и не нужно. Например, блокировка регистра бухгалтерии с указанием значения вида субконто. Достаточно понимать, что, накладывая такую блокировку, мы запрещаем другим транзакциям каким-либо образом изменять «записи» регистра бухгалтерии, у которых значение вида субконто равно указанному нами. Как при этом данное условие «проецируется» на реальную структуру данных регистра бухгалтерии - для нас совершенно не важно.

При установке новых блокировок менеджер анализирует имеющиеся блокировки. Если оказывается, что «нечто», что мы пытаемся заблокировать, уже заблокировано ранее, сравниваются режимы существующей и новой блокировок. Если режимы совместимы, новая блокировка устанавливается. Если режимы не совместимы, новая блокировка ожидает снятия существующей блокировки.

Условия необходимо ставить именно на те поля, имена которых приведены в списке имен пространств блокировок. Для каждого пространства блокировок количество устанавливаемых условий неограниченно. Условия могут быть заданы или на равенство значения поля какому-либо значению, или на вхождение значения поля в указанный диапазон.

Существует два способа задать условия на поля пространств блокировки:

- с помощью явного задания имени поля и его значения;
- с помощью указания источника данных, содержащего необходимые значения.

При явном задании имени поля и его значения необходимо использовать метод `УстановитьЗначение()` объекта `ЭлементБлокировкиДанных`. В этом случае имя и значение указывают в качестве параметров метода, например, так, как показано в листинге 4.74.

Листинг 4.74. Пример установки условия блокировки записей
с помощью явного указания имени поля и его значения

```
II Создать объект блокировка данных БлокировкаДанных = Новый БлокировкаДанных;

II Добавить новый элемент блокировки, блокирующий "нечто" в данных регистра II накопления Остатки
номенклатуры
ЭлементБлокировки = БлокировкаДанных.Добавить("РегистрНакопления.ОстаткиНоменклатуры");

II Установить режим блокировки - исключительный. Другие транзакции, устанавливающие II управляемые
блокировки, не смогут даже начать чтение этих данных ЭлементБлокировки.Режим =
РежимБлокировкиДанных.Исключительный;

II Указать, что именно мы блокируем в данных регистра Остатки номенклатуры - все II
"записи", у которых значение измерения Склад равно значению, содержащемуся в II
переменной Склад
ЭлементБлокировки.УстановитьЗначение("Склад", Склад);
```

Для значений типа Дата или Число в качестве значения может быть задан некоторый диапазон значений. Диапазон значений передается методу с помощью объекта встроенного языка Диапазон. Данный объект позволяет задать верхнюю и нижнюю границы диапазона, причем в диапазон включаются и границы диапазона (листинг 4.75).

Листинг 4.75. Пример установки условия блокировки записей с помощью задания диапазона

```
II Создать объект блокировка данных БлокировкаДанных = Новый БлокировкаДанных;

II Добавить новый элемент блокировки, блокирующий "нечто" в данных регистра II накопления Продажи
ЭлементБлокировки = БлокировкаДанных.Добавить("РегистрНакопления.Продажи");

II Установить режим блокировки - разделяемый. Эти данные гарантированно не будут II изменены другими
транзакциями до окончания существующей транзакции ЭлементБлокировки.Режим =
РежимБлокировкиДанных. Разделяемый;

II Указать, что именно мы блокируем в данных регистра Продажи - все "записи", у
II которых значение измерения Контрагент равно значению, содержащемуся в переменной II Контрагент
ЭлементБлокировки.УстановитьЗначение("Контрагент", Контрагент);

II Создать объект Диапазон, описывающий интервал от начала месяца, к которому II принадлежит
указанная дата, до указанной даты Диапазон = Новый Диалазон(НачалоМесяца(Дата), Дата);

II Указать, что именно мы блокируем в данных регистра Продажи - все "записи", у II которых значение
измерения Контрагент равно значению, содержащемуся в переменной II Контрагент, и значение поля
Период содержится в указанном диапазоне ЭлементБлокировки.УстановитьЗначение("Период", Диапазон);
```

При указании источника данных сначала необходимо задать свойство Источ- никДанных объекта ЭлементБлокировкиДанных, после чего, используя метод ИспользоватьИзИсточникаДанныхО, настроить соответствие полей области блокировки данных полям источника данных (листинг 4.76).

Листинг 4.76. Пример установки условия блокировки записей с помощью источника данных

```
II Создать объект блокировка данных
БлокировкаДанвых = Новый БлокировкаДанных;

II Добавить новый элемент блокировки, блокирующий "нечто" в данных регистра II
накопления Остатки номенклатуры
ЭлементБлокировки = БлокировкаДанных.Добавить("РегистрНакопления.ОстаткиНоменклатуры");

II Установить режим блокировки - исключительный. Другие транзакции, устанавливающие II
управляемые блокировки, не смогут даже начать чтение этих данных
ЭлементБлокировки.Режим = РежимБлокировжДанных.Исключительный;

II Указать, что именно мы блокируем в данных регистра Остатки номенклатуры - все
II "записи", у которых значение измерения Склад равно значению, содержащемуся в II переменной Склад
ЭлементБлокировки.УстановитьЗначение("Склад", Склад);

II Указать источник данных, который содержит данные для установки ограничений на
II другие поля этого элемента блокировки - в данном случае таблица значений II СписокНоменклатуры
ЭлементБлокировки.ИсточникДанных = СписокНоменклатуры;

II Указать, что именно мы блокируем в данных регистра Остатки номенклатуры - все II
"записи", у которых значение измерения Склад равно значению, содержащемуся в II
переменной Склад и у которых значение измерения Номенклатура равно какому-либо II
значению, содержащемуся в колонке Номенклатура указанного источника данных
ЭлементБлокировки.ИспользоватьИзИсточникаДанных("Номенклатура", "Номенклатура");
```

В качестве источника данных можно указывать результат запроса, табличную часть, набор записей или таблицу значений. При установке соответствия полей именами полей источника будут являться имена колонок результата запроса, имена реквизитов табличной части, имена измерений или имена колонок таблицы значений соответственно. Заметим, что объект Диапазон также может являться значением поля источника данных.

Для установки всех созданных нами блокировок используется метод объекта БлокировкаДанных - Заблокировать (). На рисунке 4.76 показано действие данного метода в случае использования его внутри транзакции и вне нее.



Рис. 4.76. Схема вызова метода «Заблокировать() объекта «БлокировкаДанных»

Как следует из рисунка, если этот метод выполняется внутри транзакции (явной или неявной), то блокировки устанавливаются в момент вызова метода. При окончании транзакции они будут сняты автоматически. Если же метод `Заблокировать ()` выполняется вне транзакции, то блокировки установлены не будут.

Глава 5. Клиент-серверный вариант работы

Общие сведения о клиент-серверном варианте работы

Клиент-серверный вариант работы системы «1С:Предприятие» предполагает использование трехуровневой архитектуры: клиент - кластер серверов «1С:Предприятия» - сервер баз данных. В этом варианте работы клиентское приложение, работающее у пользователя (толстый клиент, тонкий клиент или веб-клиент), взаимодействует с кластером серверов «1С:Предприятия», а кластер, при необходимости, обращается к серверу баз данных (Microsoft SQL Server, PostgreSQL, IBM DB2 или Oracle Database), рис. 5.1.

Трехуровневая архитектура «1С:Предприятия» разработана таким образом, что пользователю не требуется доступ ни к каким файловым ресурсам, связанным с «1С:Предприятием». Пользователю клиентского приложения «1С:Предприятия» также не требуется доступ к базе данных сервера баз данных. Для доступа к информационной базе клиентское приложение взаимодействует с кластером серверов «1С:Предприятия», который образует промежуточный программный слой между клиентским приложением и сервером баз данных.

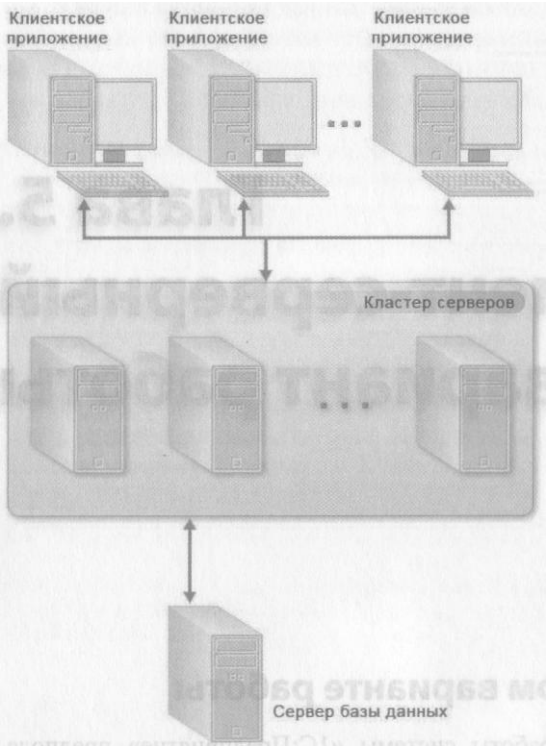


Рис. 5.1. Схема работы в клиент-серверном варианте

При этом физически кластер серверов «1С:Предприятия» и сервер баз данных могут располагаться как на одном компьютере, так и на разных. Это позволяет администратору при необходимости распределять нагрузку между разными компьютерами. Наиболее желательным является размещение всех приложений на разных компьютерах.

Программные компоненты клиент-серверной архитектуры «1С:Предприятия»

В общем случае под клиент-серверным вариантом работы «1С:Предприятия» понимается архитектура, включающая следующие программные уровни (рис. 5.2):

- клиентское приложение (толстый клиент, тонкий клиент или веб-клиент);
- кластер серверов «1С:Предприятия»;

- веб-сервер (для веб-клиента и тонкого клиента, подключенного через вебсервер);
- сервер баз данных.

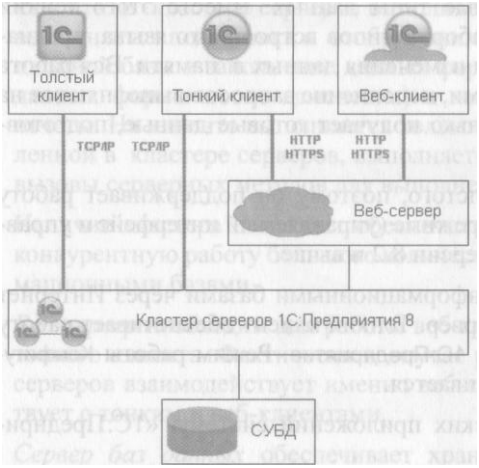


Рис. 5.2. Общая схема работы в клиент-серверном варианте

Толстый клиент - это одно из клиентских приложений системы «1С:Предприятие». Исполняемый файл этого приложения - Icv8.exe.

«Толстым» клиент называется потому, что может исполнять практически всю функциональность, предоставляемую встроенным языком, в том числе умеет работать с прикладными типами данных, такими как
`СправочникОбъект.<имя>, ДокументОбъект.<имя>` и т.д.

Но по этой же причине он требует значительного количества аппаратных ресурсов на компьютере пользователя и может «общаться» с базой данных или с сервером «1С: Предприятия» только посредством файлового доступа или по локальной сети.

Помимо работы в пользовательском режиме 1С: Предприятие, толстый клиент может работать в режиме Конфигуратор, в котором выполняется администрирование информационных баз и разработка прикладных решений.

Толстый клиент поддерживает работу приложения в обычном режиме. Поэтому его можно использовать для работы с прикладными решениями, созданными в старых версиях платформы - 8.0 и 8.1, которые не использовали управляемый интерфейс и управляемые формы.

Тонкий клиент - это одно из клиентских приложений системы «1С:Предпри- ятие». Исполняемый файл этого приложения - lcv8c.exe.

«Тонким» клиент называется потому, что умеет исполнять ограниченный набор функциональности встроенного языка. В частности, на тонком клиенте недоступны все прикладные типы данных. Вместо этого тонкий клиент оперирует ограниченным набором типов встроенного языка, предназначенным лишь для отображения и изменения данных в памяти. Вся работа с базой данных, объектными данными, исполнение запросов выполняются на стороне сервера. Тонкий клиент только получает готовые данные, подготовленные для отображения.

Тонкий клиент гораздо «легче» толстого, поэтому он поддерживает работу приложения только в управляемом режиме (управляемый интерфейс и управляемые формы), реализованном в версии 8.2 и выше.

Тонкий клиент может работать с информационными базами через Интернет и по локальной сети, и через веб-сервер. Тонкий клиент обеспечивает работу только в пользовательском режиме 1С: Предприятие. Режим работы Конфигуратор тонким клиентом не поддерживается.

Веб-клиент — это одно из клиентских приложений системы «1С:Предприятие».

У веб-клиента нет исполняемого файла. Для начала работы ему не нужно иметь никаких файлов на компьютере пользователя.

Веб-клиент, в отличие от толстого и тонкого клиентов, исполняется не в среде операционной системы компьютера, а в среде интернет-браузера (Windows Internet Explorer, Mozilla Firefox, Google Chrome или Safari). Поэтому любому пользователю достаточно всего лишь запустить свой браузер, ввести адрес веб-сервера, на котором опубликована информационная база, и веб-клиент сам «приедет» к нему на компьютер и начнет исполняться.

Веб-клиент использует DHTML, HTTPRequest и элементы технологии AJAX (Asynchronous JavaScript and XML - «асинхронный JavaScript и XML»). Вся функциональность клиента в этом режиме исполняется системой автоматически с помощью языка JavaScript.

Веб-клиент может работать с информационными базами через Интернет только через веб-сервер.

Кластер серверов «1С:Предприятия» обеспечивает взаимодействие между клиентскими приложениями и системой управления базами данных. Кластер серверов «1С: Предприятия» представляет собой множество рабочих процессов, агентов сервера и менеджеров кластера, обслуживающих один и тот же набор информационных баз.

Рабочие процессы кластера серверов «1С:Предприятия» могут функционировать как на одном компьютере (в простейшем виде), так и на нескольких

компьютерах, работающих под управлением различных операционных систем как Windows, так и Linux.

Архитектура системы ориентирована на максимальный перенос выполнения всей функциональности на кластер серверов и максимальное «облегчение» клиента. В кластере серверов выполняется вся работа прикладных объектов, выполняется подготовка к отображению форм (чтение объектов из информационной базы и заполнение данных формы, расположение элементов, запись данных формы после изменения) и командного интерфейса, формируются отчеты. На клиенте выполняется только отображение информации, подготовленной в кластере серверов, выполняется взаимодействие с пользователем и вызовы серверных методов для выполнения необходимых действий.

Наличие кластера позволяет обеспечить бесперебойную, отказоустойчивую, конкурентную работу большого количества пользователей с крупными информационными базами.

Веб-сервер необходим для работы веб-клиента и одного из вариантов работы тонкого клиента. Фактически в случае работы через веб-сервер с кластером серверов взаимодействует именно веб-сервер, а уже веб-сервер взаимодействует с тонким и веб-клиентами.

Сервер баз данных обеспечивает хранение данных информационных баз «1С:Предприятия». В качестве сервера баз данных в «1С:Предприятии» могут использоваться различные СУБД:

- Microsoft SQL Server,
- PostgreSQL,
- IBM DB2,
- Oracle Database.

Варианты использования

При использовании клиент-серверного варианта работы допускаются различные варианты взаимного расположения сервера баз данных, кластера серверов «1С:Предприятия» и клиентского приложения на компьютерах.

Наиболее желательным является размещение всех приложений на разных компьютерах (рис. 5.3).

В этом случае как сервер баз данных, так и кластер серверов «1С:Пред- приятия» смогут полноценно использовать аппаратные ресурсы и работать максимально производительно. Также этот вариант использования предоставляет лучшие возможности для масштабирования системы, поскольку модификацию аппаратных средств можно производить независимо, исходя из реальной загрузки того или иного сервера.

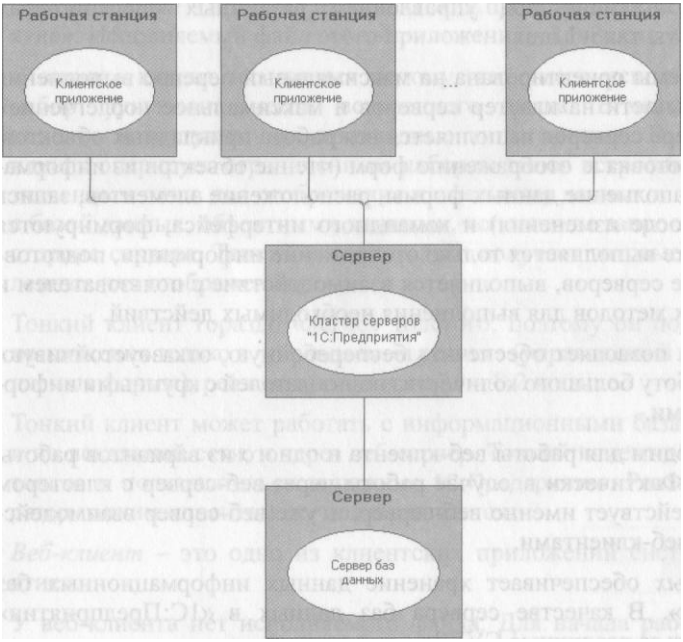


Рис. 5.3. Размещение приложений на разных компьютерах

При умеренной нагрузке на прикладное решение и при небольших объемах вычислений, производимых на сервере «1С:Предприятия», возможно размещение сервера баз данных и кластера серверов «1С:Предприятия» на одном компьютере (рис. 5.4).

Такой вариант является более дешевым, однако менее эффективным с точки зрения производительности. Работа серверов на одном компьютере предъявляет, в частности, повышенные требования к объему оперативной памяти, которая активно используется как одним, так и другим приложением.

Также возможны и другие варианты взаимного размещения приложений, например, на одном и том же компьютере (рис. 5.5).

Этот вариант вряд ли имеет смысл использовать для реальной многопользовательской работы, однако он может часто применяться, например, при разработке прикладных решений одним разработчиком.

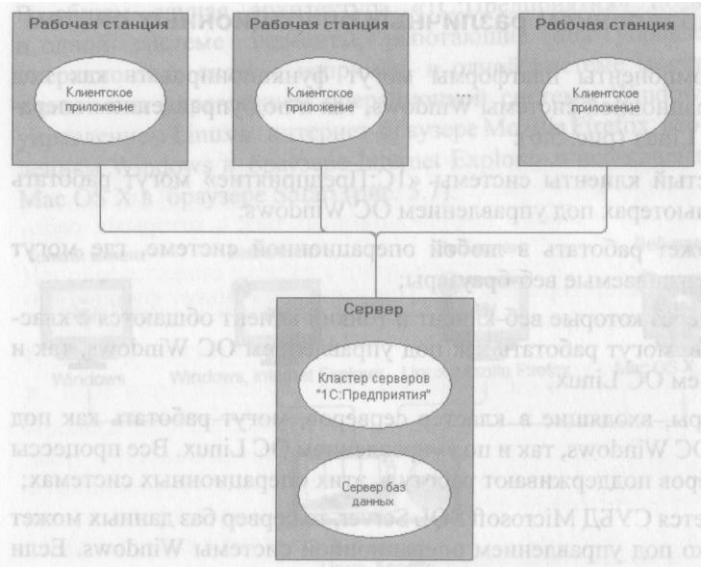


Рис. 5.4. Размещение сервера баз данных и кластера серверов «1С:Предприятия» на одном компьютере



Рис. 5.5. Размещение сервера баз данных и кластера серверов «1С:Предприятия» на рабочей станции

Работа под управлением различных операционных систем

Все основные компоненты платформы могут функционировать как под управлением операционной системы Windows, так и под управлением операционной системы Linux (рис. 5.6):

- тонкий и толстый клиенты системы «1С:Предприятие» могут работать только на компьютерах под управлением ОС Windows;
- веб-клиент может работать в любой операционной системе, где могут работать поддерживаемые веб-браузеры;
- веб-серверы, через которые веб-клиент и тонкий клиент общаются с кластером серверов, могут работать как под управлением ОС Windows, так и под управлением ОС Linux;
- рабочие серверы, входящие в кластер серверов, могут работать как под управлением ОС Windows, так и под управлением ОС Linux. Все процессы кластера серверов поддерживают работу в этих операционных системах;
- если используется СУБД Microsoft SQL Server, то сервер баз данных может работать только под управлением операционной системы Windows. Если же используются другие СУБД (кроме Microsoft SQL Server), то сервер баз данных может работать как под управлением ОС Windows, так и под управлением ОС Linux.

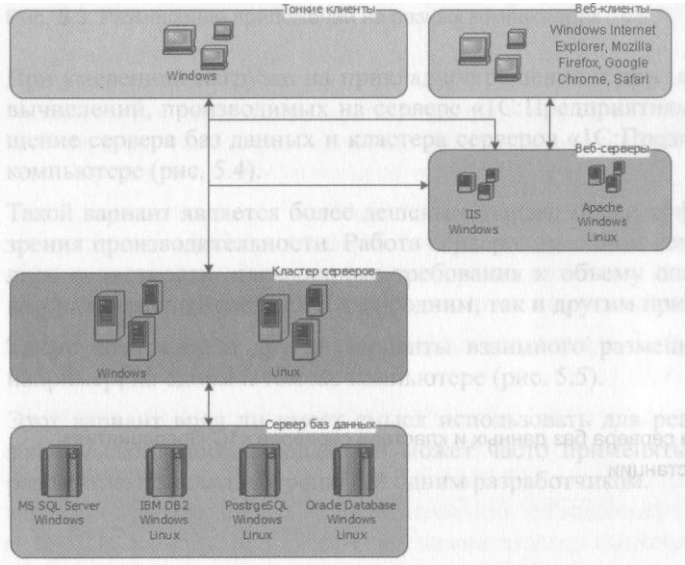


РИС. 5.6. Многоплатформенность системы «1С: Предприятие»

В общем случае архитектура «1С:Предприятия» позволяет смешивать в одной системе элементы, работающие под управлением различных операционных систем, например, в одной системе могут работать тонкий клиент под управлением операционной системы Windows, веб-клиент под управлением Linux в интернет-браузере Mozilla Firefox, веб-клиент под управлением Windows в браузере Internet Explorer и веб-клиент под управлением Mac OS X в браузере Safari (рис. 5.7).

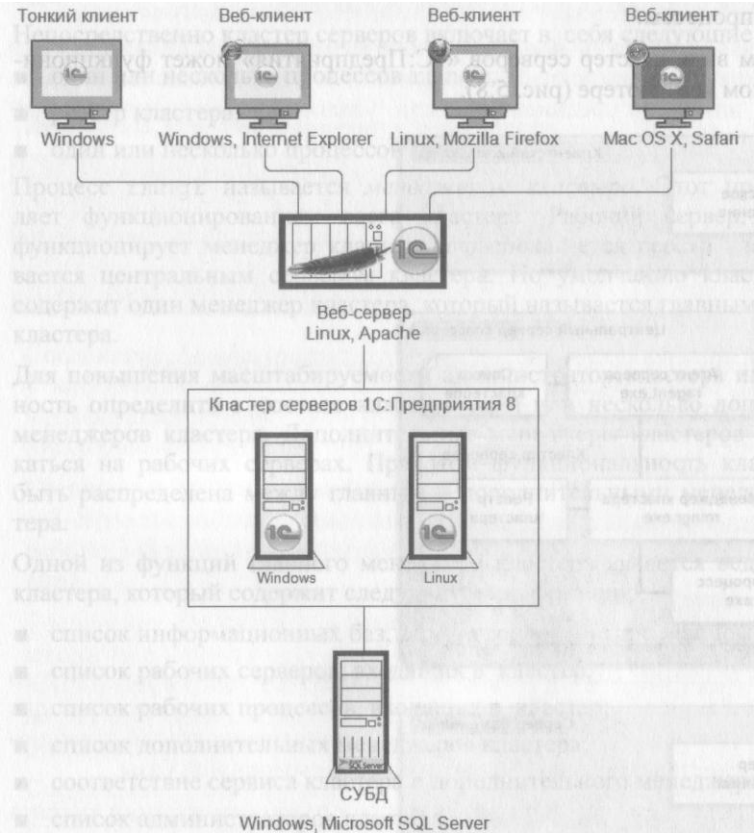


Рис. 5.7. Элементы системы «1С:Предприятия», работающие под управлением различных операционных систем

Кластер серверов

Состав кластера

Кластер серверов «1С:Предприятия» является логическим понятием и представляет собой совокупность одного или нескольких рабочих процессов, агентов сервера и менеджеров кластера, функционирующих на одном или нескольких компьютерах, и списка информационных баз, с которыми работают данные процессы.

В простейшем виде кластер серверов «1С:Предприятия» может функционировать на одном компьютере (рис. 5.8).

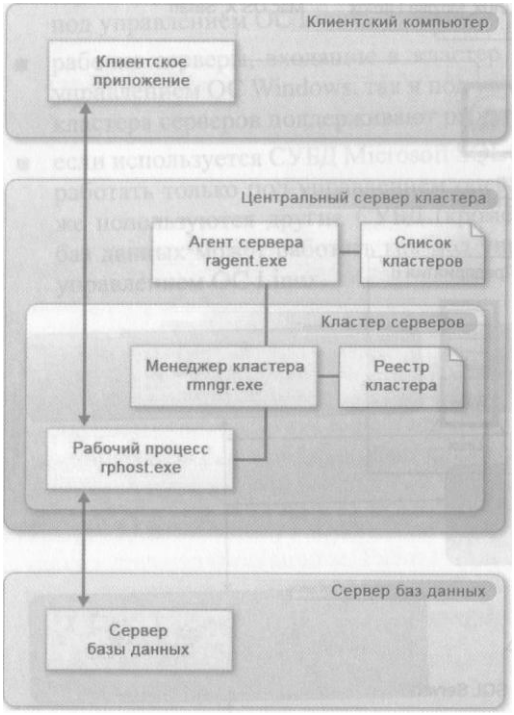


Рис. 5.8. Состав простого кластера

В работе кластера серверов используются следующие элементы:

- процессы кластера серверов: ragent, rmngr, rphost;
- хранилища данных: список кластеров, реестр кластера.

Процесс ragent, который называется *агентом сервера,* обеспечивает функционирование компьютера в составе кластера. Соответственно, компьютер, на котором запущен агент сервера, называется рабочим сервером. Одной из функций агента сервера является ведение списка кластеров, расположенных на данном рабочем сервере.

Агент сервера и список кластеров не входят в состав кластера серверов, а лишь обеспечивают работу сервера и кластеров, которые расположены на нем.

Непосредственно кластер серверов включает в себя следующие элементы:

- один или несколько процессов rmngr,
- реестр кластера,
- один или несколько процессов rphost.

Процесс rmngr называется *менеджером кластера.* Этот процесс управляет функционированием всего кластера. Рабочий сервер, на котором функционирует менеджер кластера и располагается реестр кластера, называется центральным сервером кластера. По умолчанию кластер серверов содержит один менеджер кластера, который называется главным менеджером кластера.

Для повышения масштабируемости администратор кластера имеет возможность определить в составе кластера один или несколько дополнительных менеджеров кластера. Дополнительные менеджеры кластеров могут запускаться на рабочих серверах. При этом функциональность кластера может быть распределена между главным и дополнительными менеджерами кластера.

Одной из функций главного менеджера кластера является ведение реестра кластера, который содержит следующую информацию:

- список информационных баз, зарегистрированных в данном кластере;
- список рабочих серверов, входящих в кластер;
- список рабочих процессов, входящих в кластер;
- список дополнительных менеджеров кластера;

ш соответствие сервиса кластера и дополнительного менеджера кластера;

- список администраторов кластера.

Процесс rphost называется *рабочим процессом.* Рабочий процесс обслуживает непосредственно клиентские приложения, взаимодействует с сервером баз данных, и в нем, в частности, могут исполняться процедуры серверных модулей конфигурации.

Подробнее об этом можно прочитать в книге «1С:Предприятие 8.2. Клиент- серверный вариант. Руководство администратора», раздел 2.1.

Сеансы и соединения

В процессе взаимодействия пользователя с информационной базой посредством кластера серверов используются такие понятия, как сеанс и соединение. Это не одно и то же.

Сеанс определяет активного пользователя информационной базы.

Соединение обеспечивает доступ сеансов к кластеру серверов «1С:Предприятия» и не отождествляется с активным пользователем.

Для обращения клиента к кластеру сеанс назначается соединению. Все время, пока клиент не выполняет обращений к кластеру, сеанс может быть не назначен никакому соединению. Если удаляется сеанс, назначенный соединению, то происходит разрыв этого соединения.

Виды соединений

Можно выделить два вида соединений: соединения с информационной базой и служебные соединения с рабочими процессами кластера.

Соединения с информационной базой

Возможны следующие виды соединений с информационной базой:

- Толстый клиент,
- Тонкий клиент,
- Конфигуратор,
- Модуль расширения веб-сервера,
- COM-соединение,
- WS-соединение,
- Фоновое задание.

Соединение *Толстый клиент* представляет собой соединение толстого клиента с информационной базой. Это соединение предназначено для модификации данных информационной базы и выполнения другой функциональности, предоставляемой конфигурацией информационной базы.

Соединение создается в результате интерактивного запуска толстого клиента в режиме 1С:Предприятие или в результате подключения к информационной базе с использованием технологии `Automation Client/Server`, например (листинг 5.1):

Листинг 5.1. Установка соединения «Толстый клиент»

```
II Создать Automation сервер 1С: Предприятия
AutomationCepeep = Новый COM06beKT("V82.Application");
```

```
II Установить соединение с информационной базой II TestBase в
кластере 1541 центрального сервера TestSrv
AutomationCepBep.Connect("Srvr="TestSrv";Ref="TestBase");
```

Соединение *Тонкий клиент* представляет собой соединение тонкого клиента с информационной базой. Это соединение предназначено для модификации данных информационной базы и выполнения другой функциональности, предоставляемой конфигурацией информационной базы.

Соединение создается в результате интерактивного запуска тонкого клиента или в результате подключения к информационной базе с использованием технологии Automation Client/Server, например (листинг 5.2):

Листинг 5.2. Установка соединения «Тонкий клиент»

```
II Создать Automation сервер 1С:Предприятия AutomationCepeep =
Новый C0M06beKT("V82c.Applicati0n");
II Установить соединение с информационной базой //TestBase в
кластере 1541 центрального сервера TestSrv
AutomationCepBep.Connect("SrvF"TestSrv";Ref=TestBase");
```

Соединение *Конфигуратор* представляет собой соединение конфигуратора с информационной базой. Это соединение предназначено для создания и модификации конфигурации информационной базы и для выполнения административных и регламентных действий.

Соединение *Модуль расширения веб-сервера* представляет собой соединение веб-сервера с рабочим процессом сервера. Это соединение предназначено для работы веб-клиента, Web-сервисов, а также тонкого клиента (по протоколу http).

Соединение создается в момент обращения к Web-сервису или при обращении веб-клиента или тонкого клиента (по протоколу http) к серверу «1С:Предприятия». Соединение существует до перезапуска веб-сервера или до тех пор, пока соединение находится в пуле соединений модулей расширений веб-сервера.

Соединение *COM-соединение* представляет собой соединение процесса, использующего функциональность внешнего соединения «1С:Предприятия», с информационной базой. Это соединение предназначено для модификации данных информационной базы и выполнения другой функциональности, предоставляемой конфигурацией информационной базы.

COM-соединение создается в результате подключения к информационной базе с использованием технологии COM, например (листинг 5.3):

Листинг 5.3. Установка внешнего соединения

```
// Создать Automation сервер 1С: Предприятия
COMСоединитель = Новый COMОбъект{"У82.COMСоппес1ог");

II Установить соединение с информационной базой
II TestBase в кластере 1541 центрального сервера TestSrv
СоединениеСИнформационнойБазой = COMСоединитель,Соппес(("8г¥г="Те5(8гу";Ке(="Те$!Ва8е"};
```

Соединение *WS-соединение* представляет собой соединение веб-сервера с информационной базой. Это соединение предназначено для выполнения кода модуля Web-сервиса «1С:Предприятия».

WS-соединение возникает в процессе обращения к Web-сервису, опубликованному на веб-сервере. Соединение существует до тех пор, пока соединение находится в пуле WS-соединений (пока не закончится время жизни соединения в пуле или пока данное соединение не будет вытеснено из пула другими соединениями).

Подробнее о Web-сервисах можно прочитать в книге «Технологии интеграции «1С:Предприятия 8.2» из серии «Профессиональная разработка».

Соединение *Фоновое задание* представляет собой соединение рабочего процесса кластера с информационной базой. Это соединение предназначено для выполнения кода процедуры фонового задания.

Соединение фонового задания создается в результате запуска фонового задания на выполнение. Такой запуск может выполняться системой «^Предприятие» в результате автоматического запуска регламентного задания (регламентное задание порождает соответствующее фоновое задание) или разработчиком средствами встроенного языка, например (листинг 5.4):

Листинг 5.4. Установка соединения «Фоновое задание»

```
II Выполнить фоновое задание, описанное в процедуре
//ОбновлениеИндексаПолнотекстовогоПоиска II общего модуля
РегламентныеПроцедуры ФоновоеЗадание =
ФоновыеЗадания.Выполнить)
                    "РегламентныеПроцедуры.ОбновлениеИндексаПолнотекстовогоПоиска");
```

Соединение фонового задания существует до тех пор, пока существует контекст исполняемой процедуры фонового задания. После того как процедура выполнена, соединение фонового задания закрывается.

Подробнее о фоновых заданиях см. книгу «1С:Предприятие 8.2. Руководство разработчика».

Служебные соединения

Возможны следующие виды служебных соединений:

- Планировщик заданий,
- Отладчик,
- Консоль кластера,
- COM-администратор.

Соединение *Планировщик заданий* представляет собой соединение планировщика заданий с рабочим процессом. Это соединение предназначено для управления работой фоновых заданий, в том числе для запуска регламентных заданий по расписанию.

Соединение *Отладчик* представляет собой соединение отладчика с рабочим процессом кластера, находящимся в режиме отладки. Это соединение предназначено для управления ходом отладки и поиском предметов отладки, имеющихся в настоящий момент. Соединение создается при подключении предмета отладки или при поиске предметов отладки и существует до тех пор, пока предмет отладки не будет отключен или не завершит свою работу.

Соединение *Консоль кластера* представляет собой соединение утилиты администрирования клиент-серверного варианта с рабочим процессом. Это соединение предназначено для администрирования информационных баз кластера серверов.

Соединение консоли кластера создается в момент обращения к данным рабочего процесса (например, при получении параметров информационной базы, при получении подробного списка соединений информационной базы и пр.).

Соединение *COM-администратор* представляет собой соединение с рабочим процессом сервера с использованием технологии COM. Это соединение предназначено для администрирования информационных баз кластера серверов.

Соединение COM-администратора создается при подключении к выбранному рабочему процессу с использованием технологии COM, например (листинг 5.5):

Листинг 5.5. Установка соединения «COM-администратор»

```
II Создать COMСоединитель 1С: Предприятия COMСоединитель
= Новый COШбъект( WCOMConnector");

II Установить соединение с рабочим процессом 1562 II в кластере 1541
центрального сервера TestSrv СоединениеСРабочимПроцессом =
COMCoeflHHMTenb.ConnectWorkingProcess(
                                                    "tcp://TestSrv:1562");
```

Виды сеансов

Возможны следующие виды сеансов:

- *Толстый клиент, Тонкий клиент, Веб-клиент* - создаются при обращении, соответственно, толстого, тонкого и веб-клиента к информационной базе;
- *Конфигуратор* - создается при обращении конфигуратора к информационной базе;
- *COM-соединение* — создается при обращении к информационной базе через внешнее соединение;
- *WS-соединение* — создается при обращении веб-сервера к информационной базе в результате обращения к Web-сервису, опубликованному на веб-сервере;
- *Фоновое задание* - создается при обращении рабочего процесса кластера к информационной базе. Предназначен для выполнения кода процедуры фонового задания;
- *Консоль кластера* - создается при обращении утилиты администрирования клиент-серверного варианта к рабочему процессу;
- *COM-администратор* — создается при обращении к рабочему процессу через внешнее соединение.

Подробнее об этом можно прочитать в книге «1С:Предприятие 8.2. Клиент- серверный вариант. Руководство администратора», раздел 2.1.5.

Масштабируемость кластера

Масштабируемость кластера серверов выражается в перераспределении нагрузки между несколькими менеджерами и рабочими процессами кластера и может осуществляться за счет:

- увеличения количества менеджеров кластера и распределения между ними сервисов;
- увеличения количества рабочих процессов, функционирующих на конкретном рабочем сервере;
- увеличения количества рабочих серверов, входящих в состав кластера.

Перераспределение сервисов между менеджерами кластера

В платформе «1С:Предприятие» определен состав независимых сервисов кластера серверов, которые заключают в себе всю функциональность менеджера кластера.

Три сервиса могут использоваться только на главном менеджере кластера: ***я*** *сервис конфигурации кластера* - хранит все настройки кластера;

- *сервис блокировок кластера* - хранит блокировки информационных баз, сведения об активных процессах и другую динамическую информацию о работе кластера;
- *сервис управления предметами отладки* - управляет подсоединением отладчика к серверным предметам отладки.

Другие сервисы могут быть назначены на дополнительные менеджеры кластера:

- *сервис журналов регистрации* — поддерживает доступ к журналу регистрации;
- *сервис полнотекстового поиска* - выполняет полнотекстовый поиск и осуществляет индексирование;
- *сервис заданий* - управляет запуском и отслеживанием времени жизни фоновых и регламентных заданий;
- *сервис нумерации* - обеспечивает генерацию уникальных номеров и кодов объектов;
- *сервис времени* - поддерживает получение оперативной отметки времени и некоторые вспомогательные функции;
- *сервис пользовательских настроек* - обеспечивает доступ к файлам, в которых размещаются некоторые пользовательские настройки;
- *сервис блокировок объектов* - хранит пессимистические (нетранзакци- онные) блокировки объектов;
- *сервис транзакционных блокировок* - содержит транзакционные блокировки управляемого режима;
- *сервис сеансовых данных* - обеспечивает хранение и кеширование сеансовой информации, например информация форм управляемого приложения. Обеспечивает получение клиентских лицензий.

В целях повышения масштабируемости кластера серверов наиболее ресурсоемкие сервисы могут быть назначены различным менеджерам кластера, что позволяет распределить всю нагрузку между несколькими процессами.

Подробнее о сервисах кластера можно прочитать в книге «1С:Предприятие 8.2. Клиент-серверный вариант. Руководство администратора», раздел 2.1.3.

Использование нескольких рабочих процессов

Использование нескольких рабочих процессов, с одной стороны, позволяет снизить нагрузку на каждый конкретный рабочий процесс. С другой стороны, запуск нескольких рабочих процессов позволяет более эффективно использовать аппаратные ресурсы рабочего сервера. Кроме этого, запуск нескольких рабочих процессов позволяет повысить надежность сервера, изолировав группы клиентов, работающих с разными информационными базами. В кластере, в котором возможен запуск нескольких рабочих процессов, рабочий процесс может быть автоматически перезапущен по истечении времени, установленного администратором кластера.

Подробнее об этом можно прочитать в книге «1С:Предприятие 8.2. Клиент- серверный вариант. Руководство администратора», раздел 2.1.10.1.

Использование нескольких рабочих серверов

Кластер серверов может функционировать как на одном, так и на нескольких рабочих серверах. Разделение кластера серверов по нескольким компьютерам позволяет повысить производительность кластера в целом посредством распределения нагрузки между компьютерами.

Подробнее об этом можно прочитать в книге «1С:Предприятие 8.2. Клиент- серверный вариант. Руководство администратора», раздел 2.1.10.2.

Отказоустойчивость кластера

Отказоустойчивость кластера серверов выражается в обеспечении бесперебойной работы пользователей при программных и аппаратных сбоях в кластере серверов и достигается за счет:

- резервирования самого кластера,
- резервирования рабочих процессов кластера,
- устойчивости к обрыву канала связи.

Резервирование кластера

Несколько кластеров могут быть объединены в группу резервирования. Кластеры, перечисленные в списке резервирования и имеющие одинаковые списки резервирования, образуют группу резервирования (рис. 5.9).

Кластеры, находящиеся в одной группе резервирования, синхронизируются автоматически. Пользователей обслуживает только один кластер из группы. Он считается активным. Остальные кластеры группы считаются резервными и обеспечивают синхронизацию наиболее важных данных в актуальном состоянии.

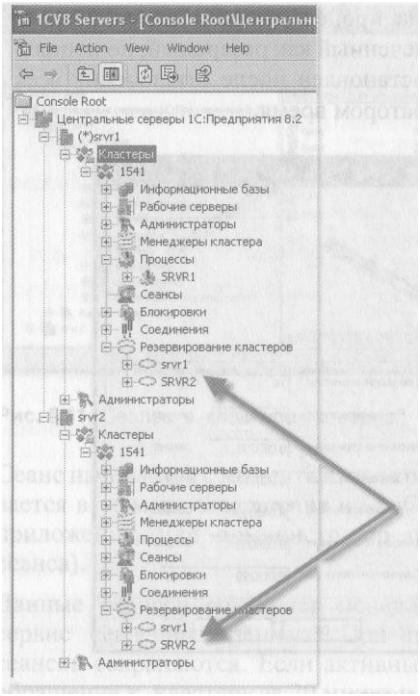


Рис. 5.9. Группа резервирования кластеров

При выходе из строя активного кластера активным становится следующий работоспособный кластер группы. При восстановлении работоспособности кластера, который находится в группе раньше активного, активность передается ему после автоматической синхронизации данных.

Подробнее об этом можно прочитать в книге «1С:Предприятие 8.2. Клиент- серверный вариант. Руководство администратора», раздел 2.1.7.1.

Резервирование рабочих процессов

Каждому рабочему процессу можно указать вариант его использования: `Использовать`, `Использовать как резервный` или `Не использовать` (рис. 5.10).

Агент центрального сервера запускает все рабочие процессы (свойство `Использование = Использовать`). Если запуск процесса закончился с ошибкой, то кластер запускает вместо него один из неактивных процессов, помеченных как резервный, при этом попытки запуска используемого

процесса продолжаются до тех пор, когда процесс запуска не увенчается успехом. Тогда самый старый процесс, помеченный как резервный, но используемый системой, выключается и будет остановлен после отключения всех соединений или через заданное администратором время.

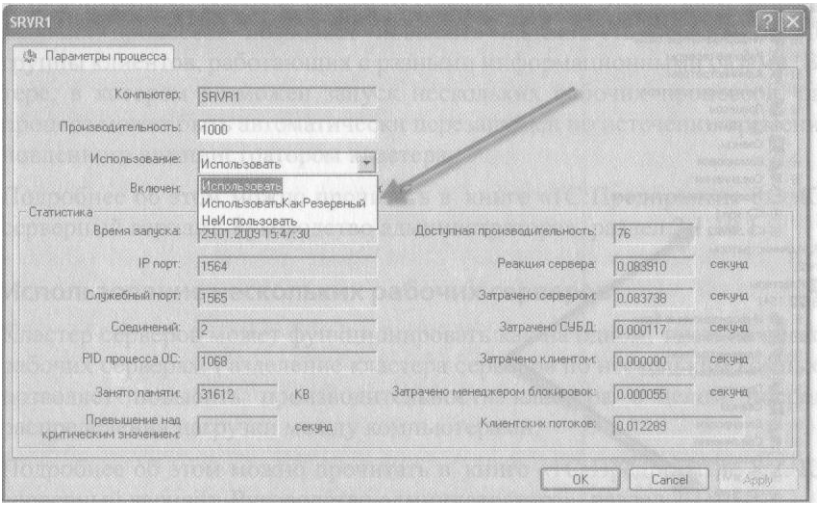


Рис. 5.10. Варианты использования рабочего процесса

Если у кластера нет резервных рабочих процессов (нет вовсе или все резервные процессы уже используются) и перезапуск неактивного рабочего процесса окончился неудачей, то система попытается перераспределить нагрузку по оставшимся рабочим процессам.

Подробнее об этом можно прочитать в книге «1С:Предприятие 8.2. Клиент- серверный вариант. Руководство администратора», раздел 2.1.7.2.

Устойчивость к обрыву канала связи

Устойчивость к обрыву канала связи реализуется за счет того, что кластер «запоминает» подключившихся пользователей и состояние выполняемых ими действий благодаря тому, что для каждого пользователя создается собственный сеанс.

Сеанс является отражением активного пользователя в кластере серверов. Не следует путать сеансы с соединениями. В процессе работы пользователя соединения могут устанавливаться, могут разрываться, но сеанс продолжает существовать (рис. 5.11).

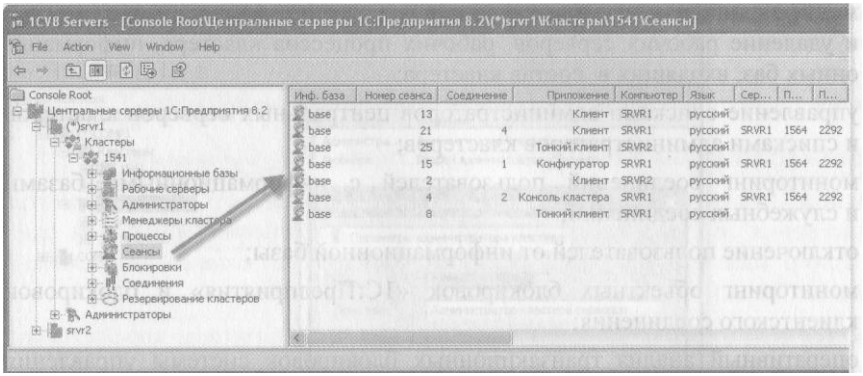


Рис. 5.11. Сеансы в кластере серверов

Сеанс начинается с момента начала работы конечного пользователя и заканчивается в момент завершения им работы (пользователь закрывает клиентское приложение, или администратор принудительно завершает работу этого сеанса).

Данные сеансов сохраняет менеджер кластера. Для этого предусмотрен сервис сеансовых данных. При перезапуске кластера серверов данные сеансов сохраняются. Если активный пользователь не выполнил ни одного обращения к кластеру за 20 минут и сеанс не назначен соединению, то сеанс удаляется вместе с данными сеанса. Для поддержания сеанса тонкий клиент и веб-клиент обеспечивают обращение к кластеру не реже 1 раза в 10 минут.

В случае потери физического соединения кластер будет ожидать восстановления соединения с этим пользователем. В подавляющем большинстве случаев после восстановления соединения пользователь сможет продолжить работу с того «места», на котором она была прекращена. При этом не потребуется повторное подключение к информационной базе.

Подробнее о сеансах и соединениях можно прочитать в книге «1С:Предприятие 8.2. Клиент-серверный вариант. Руководство администратора», раздел 2.1.5.

Управление кластером серверов

Утилита администрирования кластера серверов

Утилита администрирования кластера серверов позволяет управлять кластером серверов и выполнять такие действия, как:

■ создание, изменение и удаление кластеров серверов;

- модификация существующих кластеров: создание, изменение параметров и удаление рабочих серверов, рабочих процессов кластера, информационных баз, входящих в состав кластера;
- управление списками администраторов центральных серверов кластеров и списками администраторов кластеров;
- мониторинг соединений пользователей с информационными базами и служебных соединений;
- отключение пользователей от информационной базы;
- мониторинг объектных блокировок «1С:Предприятия» и блокировок клиентского соединения;
- оперативный анализ транзакционных блокировок системы управления базами данных;
- управление блокировкой установки соединений пользователей с информационной базой;
- управление блокировкой выполнения регламентных заданий.

Утилита представляет собой подключаемый модуль MMC (Microsoft Management Console) и может быть использована на компьютерах, на которых установлено соответствующее программное обеспечение (для операционных систем Windows 2000/XP/Server 2003 это программное обеспечение является стандартным).

Работа со списком резервирования кластеров и рабочих процессов описана в разделе «Отказоустойчивость кластера» на стр. 234. Рассмотрим некоторые другие функции по администрированию кластера серверов.

Работа со списком администраторов центрального сервера

Утилита администрирования предоставляет возможность создать список администраторов центрального сервера для того, чтобы ряд административных действий (например, добавление нового кластера, просмотр списка администраторов центрального сервера) могли выполнять только аутентифи- цированные пользователи (рис. 5.12).

Также можно просмотреть и изменить свойства администраторов центрального сервера или удалить какого-либо администратора.

Работа со списком кластеров центрального сервера

Утилита администрирования предоставляет возможность создать список кластеров центрального сервера (рис. 5.13).

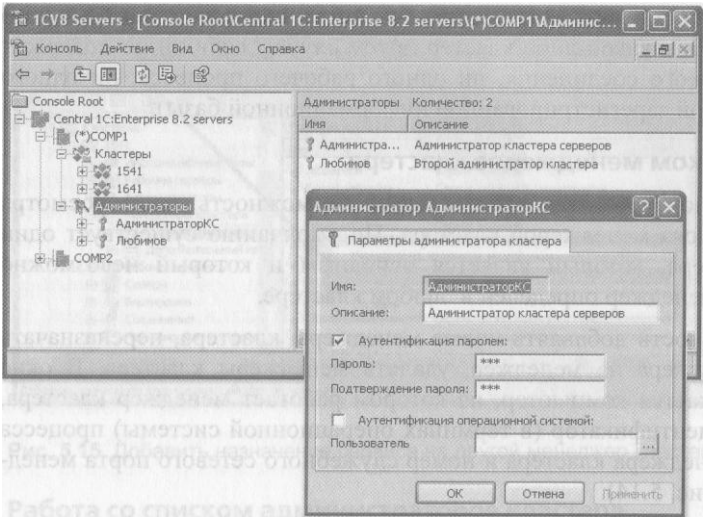


Рис. 5.12. Список администраторов кластера серверов

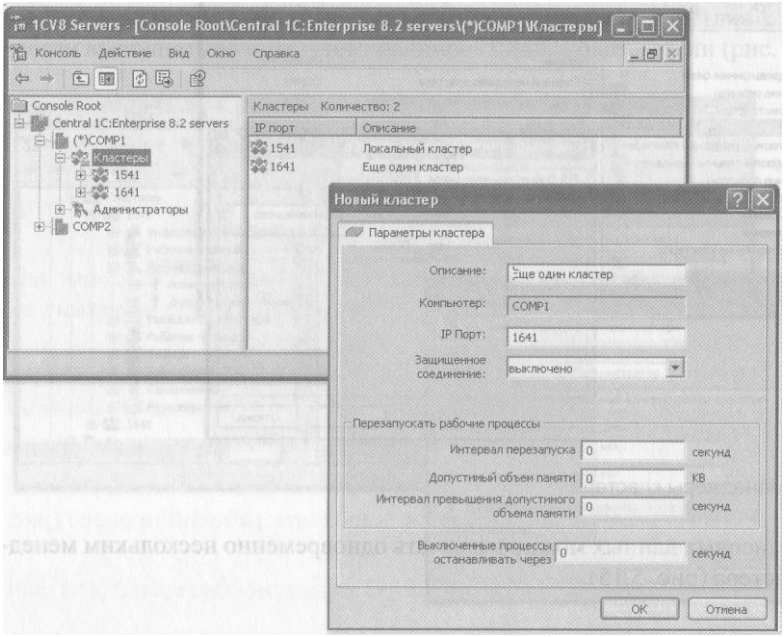


Рис. 5.13. Список кластеров

Также можно просмотреть и изменить свойства кластеров центрального сервера или удалить какой-либо кластер (в том случае, если он не содержит ни одного активного соединения, ни одного рабочего процесса, ни одного сервера и ни одной зарегистрированной информационной базы).

Работа со списком менеджеров кластера

Утилита администрирования предоставляет возможность для просмотра и изменения списка менеджеров кластера. По умолчанию существует один менеджер кластера, который является основным и который невозможно удалить. Такой менеджер определен в любом кластере.

Имеется возможность добавлять новые менеджеры кластера, переназначать сервисы с менеджера на менеджер, удалять менеджеры кластера. В окне свойств отображается компьютер, на котором работает менеджер кластера, его описание, идентификатор (в терминах операционной системы) процесса `rmngr`, порт менеджера кластера и номер служебного сетевого порта менеджера кластера (рис. 5.14).

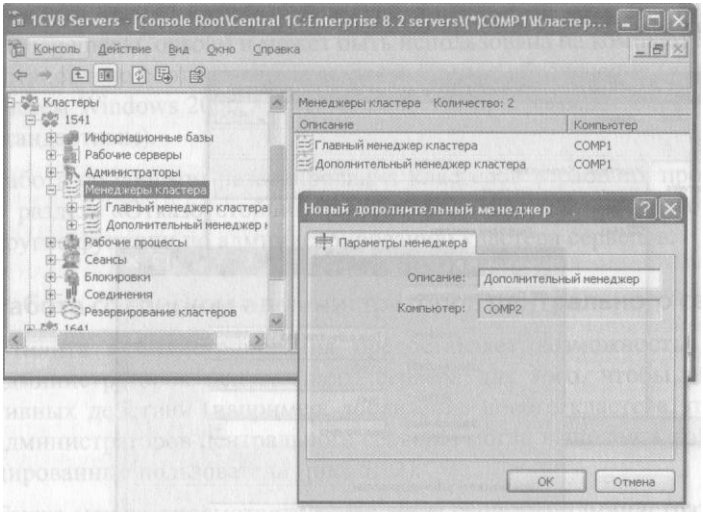


Рис. 5.14. Менеджеры кластера

Сервис сеансовых данных можно назначать одновременно нескольким менеджерам кластера (рис. 5.15).

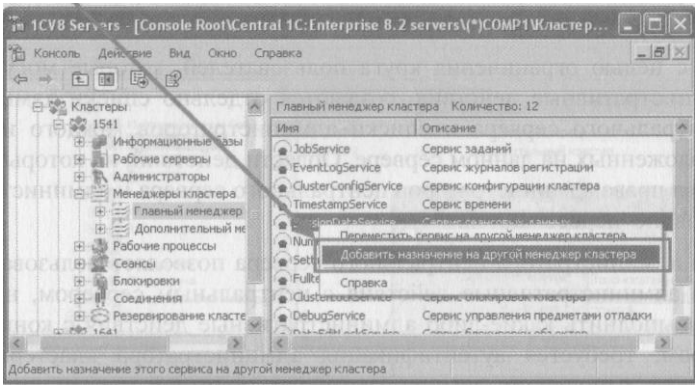


Рис. 5.15. Добавить назначение сервиса на другой менеджер кластера **Работа**

со списком администраторов кластера

Утилита администрирования предоставляет возможность создать отдельный список администраторов для каждого кластера, зарегистрированного на центральном сервере, для того, чтобы административные действия с кластером могли выполнять только аутентифицированные пользователи (рис. 5.16).

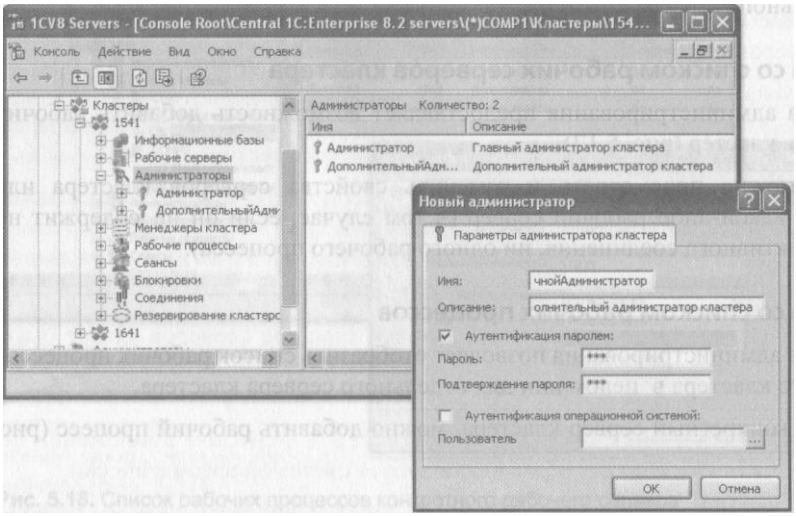


Рис. 5.16. Список администраторов кластера

Также можно просмотреть и изменить свойства администраторов кластера или удалить какого-либо администратора.

Таким образом, с целью ограничения круга пользователей, которые могут выполнять административные действия, создаются отдельно списки администраторов центрального сервера и списки администраторов каждого из кластеров, расположенных на данном сервере. Области действий, на которые распространяются права администраторов центрального сервера и администраторов кластеров, не пересекаются.

Аутентификация администратора центрального сервера позволяет пользователю выполнять административные действия с центральным сервером, но для того, чтобы выполнить какие-либо административные действия с конкретным кластером, требуется аутентификация администратора кластера. В то же время для выполнения административных действий с кластером пользователь должен аутентифицироваться как администратор кластера, но дополнительная аутентификация в качестве администратора сервера для этого не требуется.

Механизм аутентификации администраторов центрального сервера/кластера включается системой автоматически, как только в списке администраторов центрального сервера/кластера создается хотя бы один администратор. В этом случае при попытке выполнения действия, требующего аутентификации, системой автоматически будет запрошена аутентификация администратора центрального сервера/кластера.

Работа со списком рабочих серверов кластера

Утилита администрирования предоставляет возможность добавить рабочий сервер в кластер (рис. 5.17).

Также можно просмотреть и изменить свойства серверов кластера или удалить какой-либо рабочий сервер (в том случае, если он не содержит ни одного активного соединения, ни одного рабочего процесса).

Работа со списком рабочих процессов

Утилита администрирования позволяет отобразить список рабочих процессов для всего кластера в целом или для отдельного сервера кластера.

Выбрав конкретный сервер кластера, можно добавить рабочий процесс (рис. 5.18).

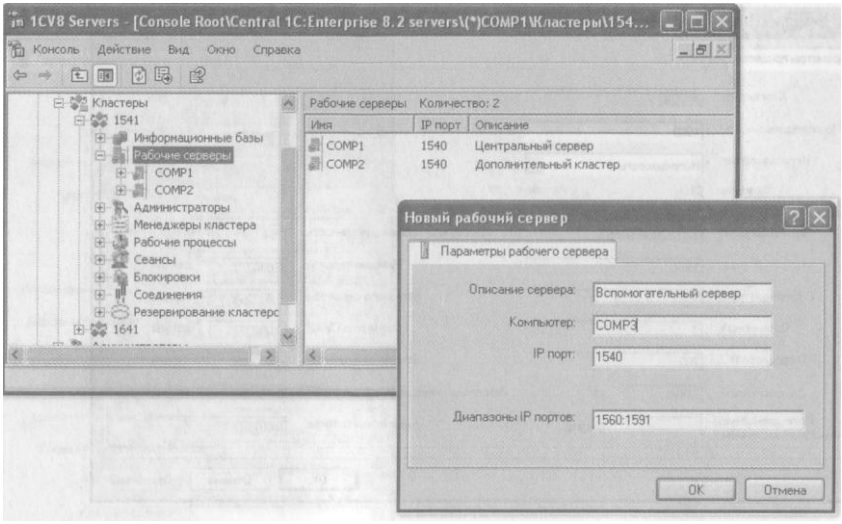


Рис. 5.17. Список рабочих серверов

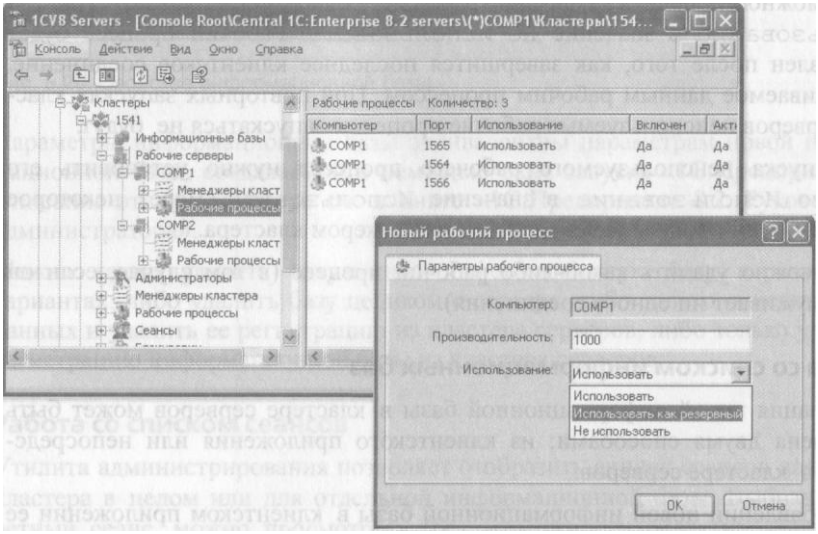


Рис. 5.18. Список рабочих процессов конкретного рабочего сервера

Выбрав конкретный рабочий процесс, можно просмотреть его свойства и изменить некоторые из них (рис. 5.19).

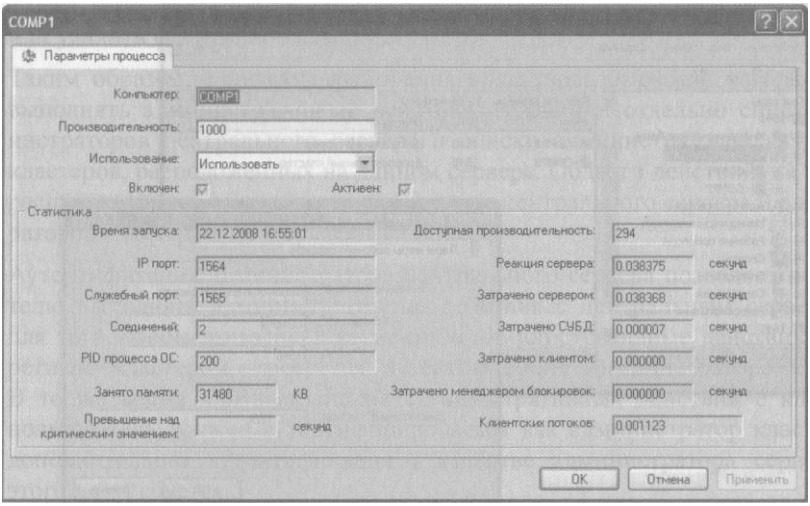


Рис. 5.19. Свойства рабочего процесса

Также можно отключить какой-либо рабочий процесс, установив его свойство `Использование` в значение `Не использовать`. Рабочий процесс будет остановлен после того, как завершится последнее клиентское соединение, обслуживаемое данным рабочим процессом. При повторных запусках кластера серверов неиспользуемые рабочие процессы запускаться не будут.

Для запуска неиспользуемого рабочего процесса нужно установить его свойство `Использование` в значение `Использовать`. Через некоторое время рабочий процесс будет запущен менеджером кластера.

Также можно удалить какой-либо рабочий процесс (в том случае, если он не обслуживает ни одного соединения).

Работа со списком информационных баз

Регистрация новой информационной базы в кластере серверов может быть выполнена двумя способами: из клиентского приложения или непосредственно в кластере серверов.

При добавлении новой информационной базы в клиентском приложении ее регистрация в кластере серверов выполняется системой автоматически.

Утилита администрирования позволяет зарегистрировать новую информационную базу в кластере серверов, просмотреть и изменить ее свойства (рис. 5.20).

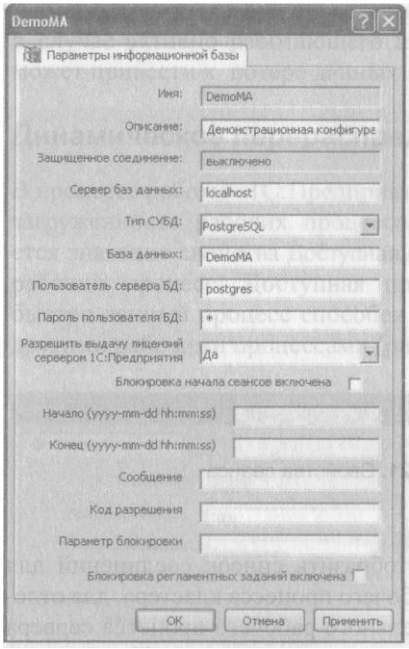


Рис. 5.20. Свойства информационной базы

Параметры информационной базы эквивалентны параметрам новой информационной базы, создаваемой с помощью окна запуска «1С:Предприятия». Описание этих параметров см. в книге «1С:Предприятие 8.2. Руководство администратора».

Можно также удалить какую-либо информационную базу в различных вариантах: либо удалить базу целиком с сервера, либо только очистить базу данных и удалить ее регистрацию из кластера серверов, либо только удалить регистрацию информационной базы из кластера серверов.

Работа со списком сеансов

Утилита администрирования позволяет отобразить список сеансов для всего кластера в целом или для отдельной информационной базы. Выбрав конкретный сеанс, можно просмотреть (без возможности редактирования) его свойства (рис. 5.21).

Администратор кластера серверов может также удалить сеанс, но в случае активно работающего пользователя с информационной базой это может привести к потере данных.

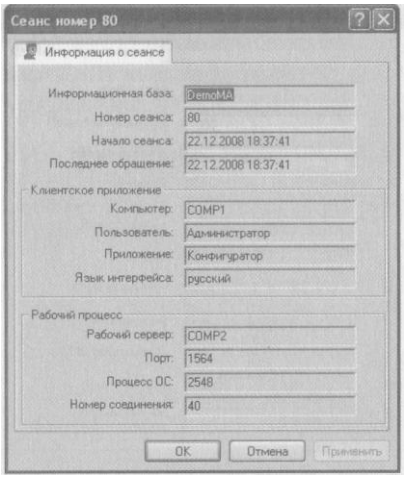


Рис. 5.21. Свойства сеанса

Работа со списком соединений

Утилита администрирования позволяет отобразить список соединений для всего кластера в целом, для отдельного рабочего процесса кластера, для отдельной информационной базы или для отдельного рабочего процесса сервера кластера. Выбрав конкретное соединение, можно просмотреть (без возможности редактирования) его свойства (рис. 5.22).

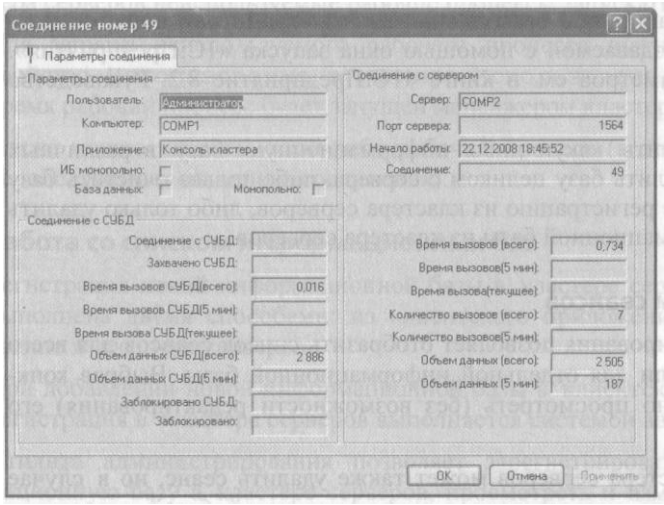


Рис. 5.22. Свойства соединения

Администратор кластера серверов может также разорвать соединение, но в случае активно работающего пользователя с информационной базой это может привести к потере данных

Динамическое перераспределение нагрузки

В процессе работы «1С:Предприятия» платформа автоматически анализирует загруженность рабочих процессов кластера серверов. При этом оценивается значение свойства Доступная производительность, которое имеет каждый рабочий процесс. Доступная производительность показывает, насколько быстро данный процесс способен выполнить вызов сервера по сравнению с другими рабочими процессами (рис. 5.23).

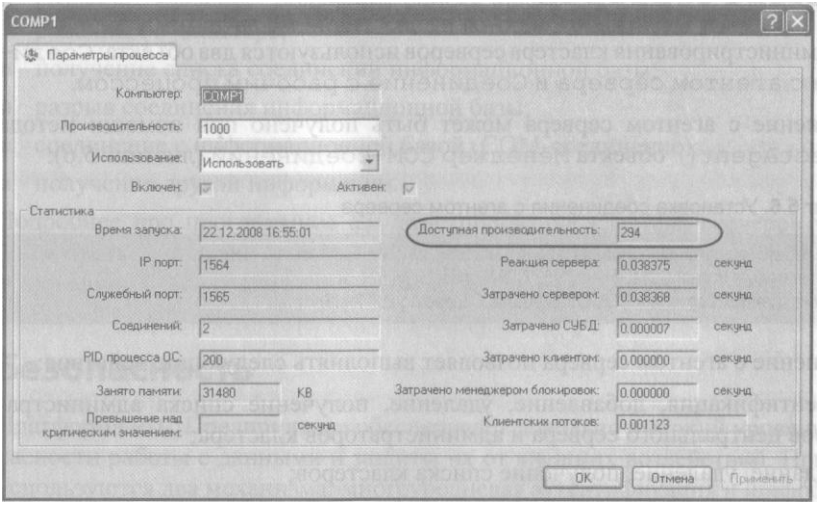


Рис. 5.23. Свойства рабочего процесса

Клиенты распределяются между рабочими процессами так, чтобы сделать доступную производительность всех рабочих процессов примерно одинаковой.

При установке нового соединения с сервером «1С:Предприятия» выбирается процесс с наибольшей доступной производительностью. При изменении соотношения между доступной производительностью рабочих процессов клиенты динамически перераспределяются между рабочими процессами.

В случае, если текущий рабочий процесс выключен или в кластере серверов появился рабочий процесс с не менее чем в два раза большей доступной

производительностью, чем у данного рабочего процесса, происходит переключение клиента на более производительный рабочий процесс. Причем выполняется это совершенно незаметно для пользователя.

Таким образом, в кластере серверов автоматически происходит динамическая балансировка нагрузки, что способствует более надежной и эффективной работе системы.

Подробнее об этом можно прочитать в книге «1С:Предприятие 8.2. Клиент- серверный вариант. Руководство администратора», раздел 2.1.8.

Программные средства администрирования кластера серверов

Для администрирования кластера серверов используются два объекта: `Соединение с агентом сервера` и `Соединение с рабочим процессом`.

Соединение с агентом сервера может быть получено при помощи метода `Connect Agent` () объекта `Менеджер COM-соединений` (листинг `5.6`):

Листинг 5.6. Установка соединения с агентом сервера

```
//Создать COMСоедмнитель 1С: Предприятия COMСоединитель = Новый
COMОбъект("У82.COMСоппес!or"); СоединениеСАгентомСервера =
COMСоеflННННТenb.ConneotAgent("TestSrv");
```

Соединение с агентом сервера позволяет выполнять следующие действия:

- аутентификация, добавление, удаление, получение списка администраторов центрального сервера и администраторов кластера;
- создание, удаление, получение списка кластеров;
- создание, удаление, получение списка серверов;
- создание, удаление, получение списка рабочих процессов кластера;
- получение списка сервисов кластера;
- получение списка сеансов информационной базы;
- получение списка соединений кластера;
- получение списка соединений информационной базы;
- получение списка информационных баз, зарегистрированных в кластере;
- получение списка блокировок кластера;
- получение списка резервирования;
- получение другой информации.

Соединение с рабочим процессом может быть получено при помощи метода `ConnectWorkingProcess()` объекта `Менеджер COM-соединений` (листинг 5.7).

Листинг 5.7. Установка соединения с рабочим процессом

```
II Создать COMСоединитель 1 С: Предприятия COMСоединитель = Новый COMОбъект("У82,COMСоппес1ог");
СоединениеСРабочимПроцессом = COMCoeflHHOTenb.ConnectWorkingProcess("TestSrv:1562");
```

Соединение с рабочим процессом позволяет выполнять следующие действия:

- аутентификация пользователей информационных баз;
- создание, удаление, получение списка информационных баз, зарегистрированных в кластере;
- получение списка соединений информационной базы;
- разрыв соединения информационной базы;
- соединение с информационной базой (COM-соединение);
- получение другой информации.

Подробнее про программное администрирование кластера серверов можно посмотреть в синтакс-помощнике: Средства интеграции и администрирования ► Менеджер COM-соединений ► Администрирование кластера серверов.

Безопасность

Платформа «1С:Предприятие» обеспечивает довольно высокий уровень безопасности работы с данными и защиты их от внешних воздействий. При этом используются два механизма: многоуровневая аутентификация и шифрование передаваемых данных.

Многоуровневая аутентификация

Многоуровневая аутентификация используется при подключении к базе данных через веб-сервер, то есть при работе «1С:Предприятия» в режиме веб-клиента и в режиме тонкого клиента, подключенного через веб-сервер (рис. 5.24).

Первый уровень аутентификации обеспечивается на веб-сервере средствами самого веб-сервера в момент соединения с веб-сервером. При этом пользователь должен пройти процедуру аутентификации.

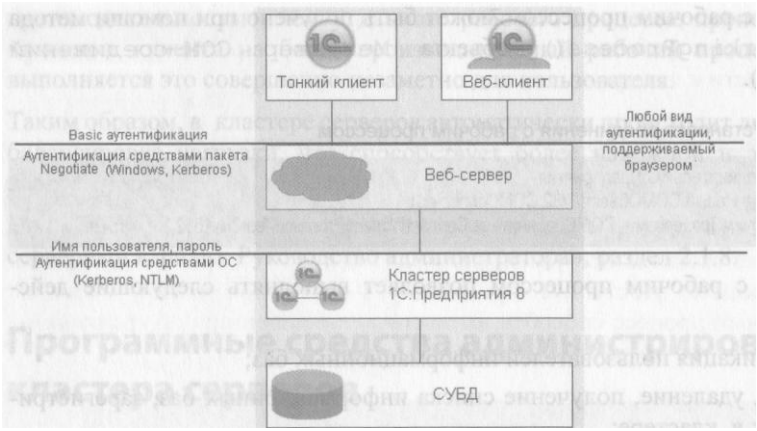


Рис. 5.24. Многоуровневая аутентификация

При подключении к веб-серверу с помощью веб-клиента можно использовать любые виды аутентификации, поддерживаемые браузером пользователя.

При подключении с помощью тонкого клиента можно использовать два вида аутентификации: Basic аутентификацию и аутентификацию с помощью пакета Negotiate.

Если выполняется *Basic аутентификация,* то пользователь просто вводит свое имя и пароль.

Если выполняется *аутентификация с помощью пакета Negotiate,* то подключающийся клиент сопоставляется одному из доменных пользователей. При этом если пользователь находится в том же домене, что и веб-сервер, то аутентификация будет выполнена автоматически, незаметно для пользователя.

Если же подключающийся пользователь хочет аутентифицироваться на веб-сервере не как текущий пользователь операционной системы, а как пользователь того домена, в котором находится веб-сервер, то нужно, чтобы вместо аутентификации текущего пользователя в операционной системе тонкий клиент использовал диалог для ввода имени и пароля пользователя вручную. Для этого можно использовать настройки информационной базы (рис. 5.25) или параметр `WSA-` при запуске «1С:Предприятия» из командной строки.

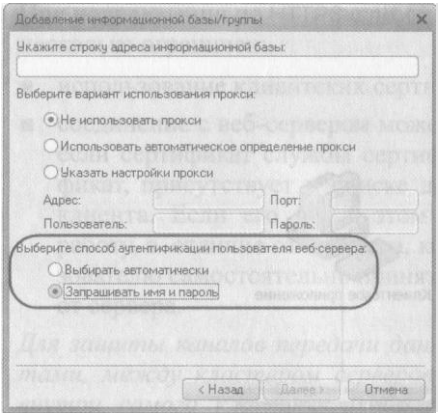


Рис. 5.25. Установка способа аутентификации в настройках информационной базы

Подробнее о параметрах запуска из командной строки можно посмотреть во встроенной справке - Справка ► Содержание справки ► 1С: Предприятие ► Запуск 1С:Предприятия 8 и параметры запуска ► Общие параметры запуска.

Второй уровень аутентификации выполняется средствами самого «1С:Пред- приятия» при установке соединения с информационной базой.

Платформа поддерживает два вида аутентификации: аутентификация «1С:Предприятия» и аутентификация операционной системы.

В случае аутентификации средствами «1С:Предприятия» пользователь при начале работы с прикладным решением должен указать имя пользователя и соответствующий этому имени пароль.

В случае аутентификации средствами операционной системы аутентификация будет «прозрачна» для пользователя. Система анализирует, от имени какого пользователя операционной системы выполняется подключение к прикладному решению, и на основании этого определяет соответствующего пользователя «1С:Предприятия». При этом диалог аутентификации «1С:Предприятия» не отображается, если не указан специальный параметр командной строки.

Шифрование передаваемых данных

Шифрование передаваемых данных служит для защиты передаваемой информации от несанкционированного доступа к ней извне. В процессе клиент-серверного взаимодействия можно выделить различные области передачи данных, нуждающиеся в защите (рис. 5.26).

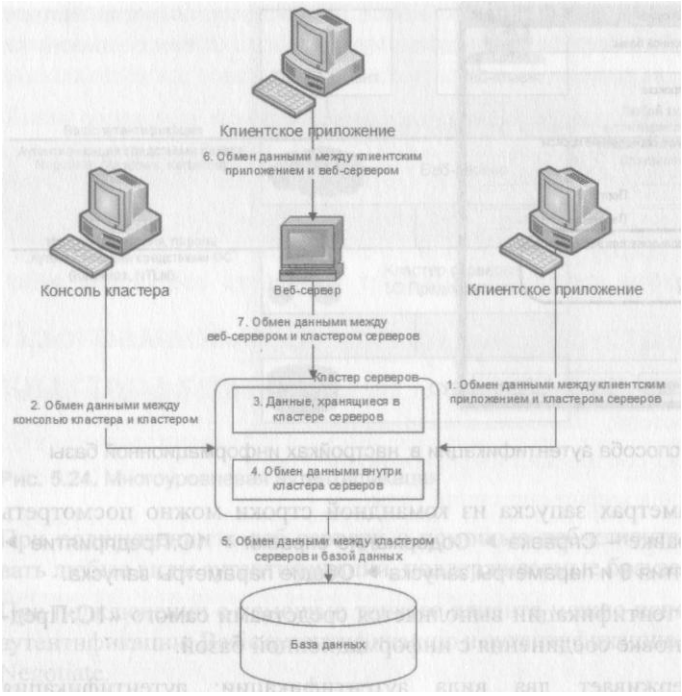


Рис. 5.26. Общая схема безопасности

Безопасность этих участков передачи данных обеспечивается путем применения различных алгоритмов шифрования данных.

Для защиты канала передачи данных между веб-клиентом и веб-сервером или между тонким клиентом и веб-сервером используются криптографические протоколы SSL или TLS.

Поддержку этих протоколов обеспечивает HTTPS соединение с веб-сервером. Для этого на сервере должен находиться действительный серверный сертификат, который гарантирует клиенту подлинность открытого ключа сервера, используемого для шифрования данных.

При подключении по HTTPS с помощью веб-клиента можно использовать любые возможности этого протокола, поддерживаемые веб-браузером. Например, можно использовать клиентские сертификаты, гарантирующие серверу подлинность клиента.

При подключении по HTTPS с помощью тонкого клиента набор возможностей несколько ограничен:

- использование клиентских сертификатов не поддерживается;
- соединение с веб-сервером может быть установлено только в том случае, если сертификат службы сертификации, подписавшей серверный сертификат, присутствует в списке доверенных служб сертификации тонкого клиента. Если его нет в этом списке, тонкий клиент завершит свою работу, в отличие от браузера, который в этой ситуации предложит пользователю самостоятельно принять или отклонить сертификат, полученный от сервера.

Для защиты каналов передачи данных между кластером серверов и клиентами, между кластером серверов и веб-сервером и для защиты каналов внутри самого кластера используются алгоритмы шифрования данных, реализуемые системой «1С:Предприятие»: RSA и Triple DES. При этом может быть выбран один из трех уровней безопасности: `Постоянно`, `Установка соединения` или `Выключено`.

Использование уровня безопасности `Постоянно` позволяет полностью защитить весь поток данных (как пароли, так и непосредственно данные), но при этом возможно значительное снижение производительности системы.

Использование уровня безопасности `Установка соединения` позволяет частично защитить весь поток данных (только пароли). Этот уровень безопасности является компромиссом между безопасностью и производительностью.

Уровень безопасности `Выключено` является самым низким и самым производительным. Практически все данные передаются без использования шифрования.

Рассмотрим схему взаимодействия клиента и кластера серверов при использовании уровня безопасности Постоянно (рис. 5.27).

Протокол взаимодействия одинаков как для менеджера кластера (rmngr), так и для рабочего процесса (rphost): после установки соединения первый обмен данными выполняется с использованием шифрования по алгоритму RSA, дальнейший обмен данными выполняется с использованием шифрования по алгоритму Triple DES.

Уровень безопасности задается при создании информационной базы. Эта информация сохраняется как на клиенте (в списке информационных баз), так и в кластере серверов (в реестре сервера). После создания информационной базы ее параметры, сохраненные в кластере серверов, изменить уже нельзя. Однако можно изменить уровень безопасности, указанный для этой информационной базы на клиенте.

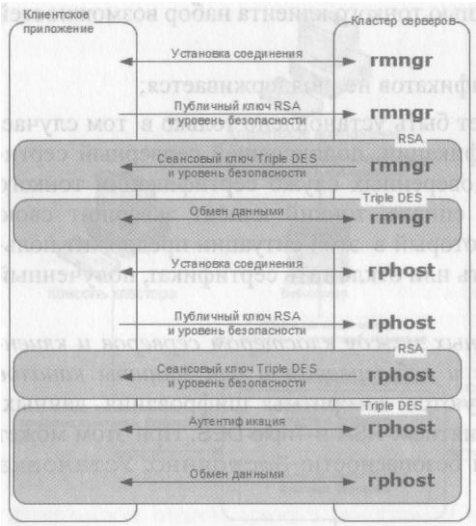


Рис. 5.27. Уровень безопасности «Постоянно»

Поэтому после установки соединения клиент генерирует приватный и публичный ключи для шифрования RSA и передает в кластер серверов публичный ключ и уровень безопасности, который указан для данной информационной базы на клиенте. Это желаемый уровень безопасности.

Кластер серверов выбирает максимальный уровень безопасности из переданного клиентом и указанного для этой информационной базы в реестре кластера. Это фактический уровень безопасности. Кроме этого, кластер серверов генерирует сеансовый ключ для шифрования Triple DES и вместе с фактическим уровнем безопасности передает его клиенту, предварительно зашифровав эти данные с помощью публичного ключа клиента.

Дальнейший обмен данными выполняется в соответствии с фактическим уровнем безопасности, при этом как клиент, так и сервер шифруют передаваемые данные с использованием сеансового ключа по алгоритму Triple DES.

Особенностью защиты передачи данных между кластером и веб-сервером является то, что соединение между кластером и веб-сервером защищается только кластером на основании свойств информационной базы, к которой выполняется подключение.

Защита канала передачи данных между кластером серверов и СУБД осуществляется средствами той СУБД, которая используется. Все поддерживаемые СУБД позволяют настроить свои клиентские компоненты, находящиеся

в кластере, так, чтобы трафик между ними и самой СУБД был зашифрован. Все поддерживаемые СУБД могут использовать протокол SSL.

Подробнее об этом можно прочитать в книге «1С:Предприятие 8.2. Клиент- серверный вариант. Руководство администратора», раздел 2.1.12.

Работа встроенного языка на сервере

Организация выполнения кода на сервере или на клиенте

Используя возможность управления компиляцией модулей, исполнение кода передается с клиента на сервер. После выполнения вызванной процедуры или функции система продолжит исполнение кода на клиенте.

Как уже говорилось ранее, архитектура системы ориентирована на максимальный перенос выполнения всей функциональности на сервер и максимальное «облегчение» клиента. На сервере выполняется вся работа прикладных объектов, выполняется подготовка к отображению форм (чтение объектов из информационной базы и заполнение данных формы, расположение элементов, запись данных формы после изменения) и командного интерфейса, формируются отчеты. На клиенте выполняется только отображение информации, подготовленной на сервере, выполняется взаимодействие с пользователем и вызовы серверных методов для выполнения необходимых действий.

После запуска прикладного решения выполнение кода всегда осуществляется на клиенте. Передача выполнения с клиента на сервер может быть выполнена путем вызова серверных процедур, описанных с директивой компиляции `&НаСервере`, `&НаСервереБезКонтекста` из клиентских процедур модуля формы или команды, а также путем вызова процедуры общего модуля, исполняющегося на сервере. После того как вызываемая процедура завершит свою работу, выполнение будет передано обратно на клиента.

Клиентские и серверные процедуры в модуле формы, команды

Простейшим примером передачи выполнения кода на сервер может служить вызов серверных процедур, описанных с директивой компиляции `&НаСер- вере`, `&НаСервереБезКонтекста` из клиентских процедур модуля формы или команды.

Как уже рассказывалось в третьей главе, модуль формы и модуль команды существуют одновременно как на клиенте, так и на сервере. Таким образом, процедуры, содержащиеся в этих модулях, исполняются или в контексте клиента, или в контексте сервера.

Поэтому все процедуры и функции, создаваемые разработчиком в модуле формы или в модуле команды, должны иметь явное указание на то, в каком контексте они будут исполняться. Для этого используются *директивы компиляции,* которыми должно предваряться описание процедур.

Более подробно о директивах компиляции в модуле формы и в модуле команды рассказывается в разделах «Модуль управляемой формы» на стр. 88 и «Модуль команды» на стр. 94.

В модуле управляемой формы могут быть использованы следующие директивы компиляции:

- `&НаКлиенте` - процедура/функция исполняется в управляемом клиенте;
- `&НаСервере` - процедура/функция исполняется на сервере, при этом доступен контекст формы;
- `&НаСервереБезКонтекста` - процедура/функция исполняется на сервере, при этом недоступен контекст формы;
- `&НаКлиентеНаСервереБезКонтекста` - процедура/функция может исполняться и в управляемом клиенте, и на сервере, при этом недоступен контекст формы.

В модуле команды могут быть использованы следующие директивы компиляции:

- `&НаКлиенте` - процедура/функция исполняется в управляемом клиенте;
- `&НаСервере` - процедура/функция исполняется на сервере;
- `&НаКлиентеНаСервере` - процедура/функция может исполняться и в управляемом клиенте, и на сервере.

Если по ошибке никакая директива компиляции перед процедурой не указана, то платформа будет использовать директиву `&НаСервере`.

В целом исполнение модуля формы/команды на клиенте и на сервере характеризуется тем, что из клиентских процедур модуля формы/команды (с директивой компиляции `&НаКлиенте`) можно вызывать серверные (с директивой компиляции `&НаСервере`, `&НаСервереБезКонтекста`), тем самым передавая выполнение с клиента на сервер. После выполнения серверных процедур исполнение кода возвращается на клиента. Принудительно передать исполнение кода в обратную сторону, с сервера на клиента, невозможно, то есть нельзя из серверных процедур модуля формы/команды вызывать клиентские.

Рассмотрим пример. Пусть в форме накладной при изменении товара нужно подставлять цену товара из регистра сведений. Работа с прикладными объектами, в частности с регистром сведений, на клиенте невозможна. В этом

случае на клиенте, в обработчике события при изменении товара, можно вызвать серверную (лучше внеконтекстную) процедуру, которая возвращает актуальную цену выбранного товара на указанную дату (листинги 5.8, 5.9).

Листинг 5.8. Процедура «ТоварыТоварПриИзменении()»

```
&НаКлиенте
Процедура ТоварыТоварПриИзменении(Элемент)
    //
    ЦенаТовара = РозничнаяЦена(Товар, Дата);
КонецПроцедуры
```

Листинг 5.9. Функция «РозничнаяЦенаО»

```
&НаСервереБезКонтекста
Функция РозничнаяЦена(ВыбранныйТовар, АктуальнаяДата)
    //
    Возврат Цена:
КонецФункции
```

Взаимодействие клиентского и серверного контекста исполнения в модуле формы показано на рис. 5.28.

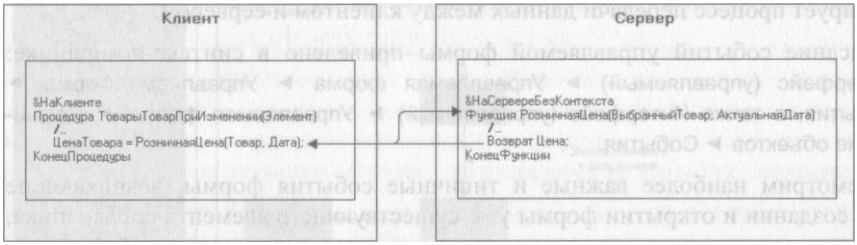


Рис. 5.28. Передача исполнения кода на сервер в модуле формы

Если процедура часто используется, имеет смысл поместить ее в общем серверном модуле и вызывать ее из серверной процедуры модуля формы.

Для реализации нужной функциональности форм разработчик может использовать клиентские и серверные процедуры в модуле формы. В целях оптимизации клиент-серверного взаимодействия нужно соблюдать следующие рекомендации:

- Основной объем кода должен содержаться в серверных процедурах (с директивой компиляции `&НаСервере`, `&НаСервереБезКонтекста`). Клиентские процедуры (`&НаКлиенте`) должны содержать небольшой и несложный для исполнения код.

- Нужно стремиться к минимизации серверных вызовов, то есть ситуаций, когда из клиентских процедур вызываются серверные.
- Везде, где возможно, нужно использовать внеконтекстные серверные вызовы (`&НаСервереБезКонтекста`) вместо вызовов сервера с контекстом формы (`&НаСервере`). Это быстрее и эффективнее.

Этому вопросу посвящена четвертая глава книги «Разработка управляемого интерфейса» из серии «Профессиональная разработка».

Клиентские и серверные обработчики событий формы, команды

В процессе жизнедеятельности у формы возникают события, связанные с ее созданием, открытием, чтением и записью данных, закрытием, обработкой выбора, внешним событием и т. д. Причем часть этих событий принадлежит самой форме, а часть (события, связанные с чтением и записью данных) поставляется расширением формы в зависимости от того объекта, который является основным реквизитом формы.

Часть событий формы выполняется на клиенте, а часть - на сервере. При этом платформа сама управляет клиент-серверным взаимодействием и оптимизирует процесс передачи данных между клиентом и сервером.

Описание событий управляемой формы приведено в синтакс-помощнике: Интерфейс (управляемый) ► Управляемая форма ► УправляемаяФорма ► События, а также Интерфейс (управляемый) ► Управляемая форма ► Расширение объектов ► События.

Рассмотрим наиболее важные и типичные события формы, возникающие при создании и открытии формы уже существующего элемента справочника, а также записи этого элемента и закрытии окна формы командой Записать и закрыть. Последовательность этих событий представлена на рис. 5.29.

Прежде всего, на сервере создается программный объект формы и программный объект, соответствующий типу основного реквизита формы, а из базы данных читаются данные этого объекта.

Затем платформа преобразовывает данные объекта в данные формы, и вызывается событие `ПриЧтенииНаСервере`. Это событие вызывается только для существующих в информационной базе объектов. Причем поставляется это событие не самой формой, а ее расширением, определяемым типом основного реквизита формы. В обработчике этого события, в параметре `ТекущийОбъект` разработчику доступен прикладной объект, содержащийся в форме, со всей его функциональностью. То есть здесь можно вызвать экспортируемые методы этого объекта или получить значения его реквизитов,

не конвертируя данные формы в объект. Таким образом, в обработчике этого события разработчик может подготовить данные формы, зависящие от данных объекта, к открытию.

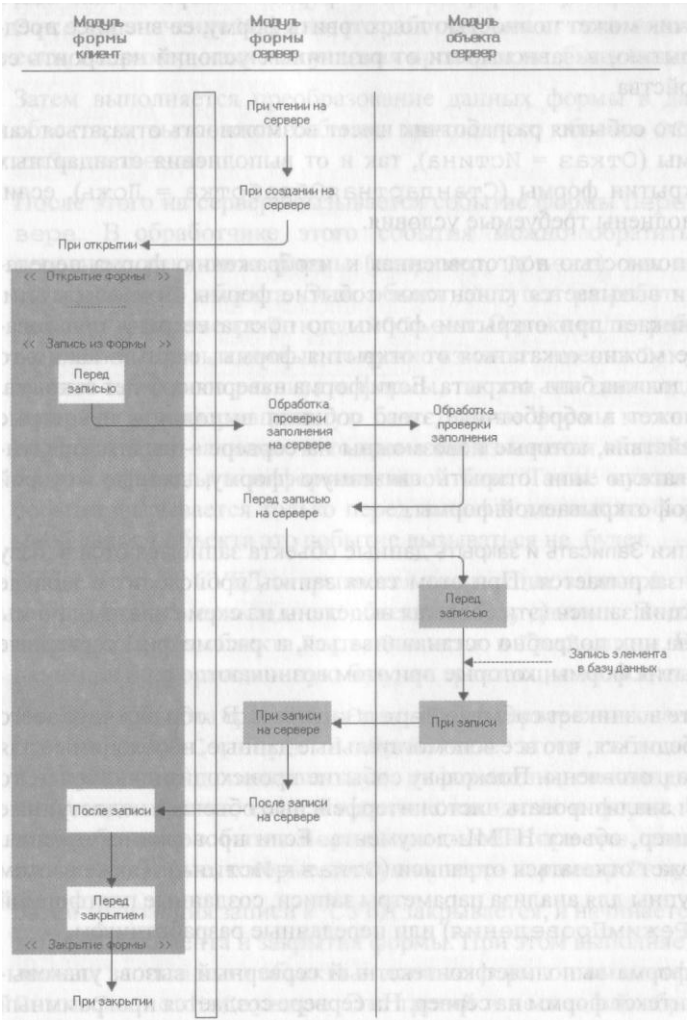


Рис. 5.29. Последовательность вызова событий модуля формы

Затем прикладной объект удаляется из памяти, и вызывается серверное событие формы ПриСозданииНаСервере. Это событие вызывается всегда при открытии форм и новых, и существующих объектов. Здесь уже недоступен прикладной объект, отображаемый в форме. В обработчике этого события разработчик может полностью подготовить форму, ее внешнее представление, к открытию, в зависимости от различных условий настроить ее интерфейсные свойства.

В обработчике этого события разработчик имеет возможность отказаться как от открытия формы (Отказ = Истина), так и от выполнения стандартных действий при открытии формы (СтандартнаяОбработка = Ложь), если, например, не выполнены требуемые условия.

После этого уже полностью подготовленная к изображению форма передается на клиента, и вызывается клиентское событие формы ПриОткрытии. Это событие возникает при открытии формы до показа ее окна пользователю. Здесь также можно отказаться от открытия формы, если по каким-то причинам она не должна быть открыта. Если форма наверняка будет открыта, то разработчик может в обработчике этого события выполнить некоторые интерактивные действия, которые невозможны на сервере - выдать предупреждение пользователю или открыть связанную форму, данные которой зависят от основной открываемой формы.

При нажатии кнопки Записать и закрыть данные объекта записываются в базу данных, и форма закрывается. При этом сама запись происходит в модуле объекта в транзакции записи (эти события выделены на схеме цветом), но мы сейчас не будем на них подробно останавливаться, а рассмотрим серверные и клиентские события формы, которые при этом возникают.

Сначала на клиенте возникает событие ПередЗаписью. В обработчике этого события можно убедиться, что все вспомогательные данные, необходимые для записи объекта, подготовлены. Поскольку событие происходит на клиенте, то здесь имеет смысл анализировать чисто интерфейсные объекты, недоступные на сервере, например, объект HTML-документа. Если проверка не прошла, то разработчик может отказаться от записи (Отказ = Истина). Также в этом обработчике доступны для анализа параметры записи, созданные платформой (РежимЗаписи, РежимПроведения) или переданные разработчиком.

После этого платформа выполняет контекстный серверный вызов, упаковывает и передает контекст формы на сервер. На сервере создается программный объект формы и заполняется из полученного контекста. И затем отрабатывает механизм проверки заполнения.

Сначала вызывается серверное событие формы ОбработкаПроверкиЗаполненияНаСервере. В обработчике этого события разработчик может

реализовать собственные алгоритмы проверки заполнения реквизитов формы, добавить в массив проверяемых реквизитов дополнительные реквизиты, чтобы платформа выполнила их проверку, или очистить массив проверяемых реквизитов, отказавшись от их стандартной проверки. Установив параметр Отказ в значение Истина, разработчик может отказаться от записи объекта, если, например, какие-то условия проверки не были выполнены.

Затем выполняется преобразование данных формы в данные прикладного объекта, и вызывается событие прикладного объекта ОбработкаПровер- киЗаполнения.

После этого на сервере вызывается событие формы ПередЗаписьюНаСер- вере. В обработчике этого события можно обратиться как к данным основного реквизита формы (например, Объект), так и к данным самого записываемого объекта. Этот объект доступен разработчику в обработчике события в параметре ТекущийОбъект. Однако если требуется модифицировать записываемые данные, то нужно изменять именно ТекущийОбъект, а не данные основного реквизита формы, так как эти данные все равно не будут записаны в информационную базу. Данные формы можно только проанализировать перед записью, в то время как изменения объекта ТекущийОбъект будут сохранены в информационной базе. Также нужно понимать, что это событие вызывается только перед записью данных из формы. При программной записи объекта это событие вызываться не будет.

После этого в СУБД открывается транзакция записи и начинается процесс записи данных прикладного объекта в базу данных. Последовательно вызываются события объекта ПередЗаписью и ПриЗаписи. Выполняется запись данных в информационную базу, но транзакция записи еще не закрывается.

После этого, пока транзакция открыта, на сервере вызывается событие формы ПриЗаписиНаСервере. В обработчике этого события можно записать в базу данных дополнительную информацию, связанную с данными записываемого объекта, если эти данные находятся в форме. Здесь так же, как и в обработчике события ПередЗаписьюНаСервере, имеет смысл изменять только те данные, которые доступны через параметр ТекущийОбъект.

Затем транзакция записи в СУБД закрывается, и начинается процесс передачи данных на клиента и закрытия формы. При этом выполняется преобразование данных записанного объекта в данные формы и вызывается событие формы ПослеЗаписиНаСервере. Это третий и последний обработчик, в котором доступны по отдельности данные формы и объект, который был записан. Однако здесь, наоборот, следует работать с данными, которые доступны через основной реквизит формы, а не через параметр ТекущийОбъект, так как данные записанного объекта уже помещены в форму, и ТекущийОбъект будет уничтожен при выходе из обработчика. В обработчике этого события

можно описывать какие-либо действия, которые должны быть выполнены только в случае, когда объект наверняка записан.

После этого в памяти уничтожается прикладной объект, форма передается на клиента и программный объект формы уничтожается на сервере. В заключение на стороне клиента вызываются два события формы — ПередЗакрытием и ПриЗакрытии.

Событие ПередЗакрытием возникает при закрытии формы до закрытия окна формы. В обработчике этого события разработчик имеет возможность отказаться как от закрытия формы (Отказ = Истина), так и от выполнения стандартных действий при закрытии формы (СтандартнаяОбработка = Ложь), если, например, не выполнены требуемые условия.

Событие ПриЗакрытии возникает при закрытии формы после закрытия окна формы. В обработчике этого события можно описывать алгоритмы, которые должны быть выполнены только в случае, когда форма будет наверняка закрыта. Например, если нужно закрыть вспомогательную форму, существующую при открытой основной форме.

Итак, мы рассмотрели последовательность наиболее типичных клиентских и серверных событий в жизни формы и механизм взаимодействия программных объектов, обеспечиваемый платформой.

Для реализации нужной функциональности форм разработчик может использовать клиентские и серверные обработчики событий формы. В целях оптимизации клиент-серверного взаимодействия нужно соблюдать следующие рекомендации:

- Большая часть кода должна быть реализована в серверных обработчиках событий формы. Например, форма должна быть максимально подготовлена для открытия в обработчиках ПриЧтенииНаСервере и ПриСозда- нииНаСервере, а в обработчике ПриОткрытии нужно выполнять только действия, недоступные на сервере, - Предупреждение, Вопрос и т. п.

м При открытии формы настоятельно не рекомендуется выполнять обращений к серверу из кода модуля формы в обработчиках клиентских событий формы, таких как ПриОткрытии и ПриПовторномОткрытии. При необходимости обращения из них к серверным данным следует размещать эти данные в реквизитах формы, в обработчике события ПриСозданииНаСервере. Особенность исполнения модуля команды заключается в том, что выполнение команды всегда начинается на клиенте, в процедуре обработчика ОбработкаКоманды(), которая обязательно должна предваряться директивой компиляции &НаКлиенте.

Вызов процедур из серверных модулей

Передача выполнения кода на сервер может быть осуществлена посредством вызова экспортируемой процедуры общего модуля, у которого установлено свойство Сервер и свойство Вызов сервера. Экземпляр этого модуля будет скомпилирован только на стороне сервера. При вызове в модуле формы отчета этой экспортной процедуры выполнение будет передано на сервер, в экспортируемую процедуру общего модуля. После ее завершения управление вернется на клиента (рис. 5.30).

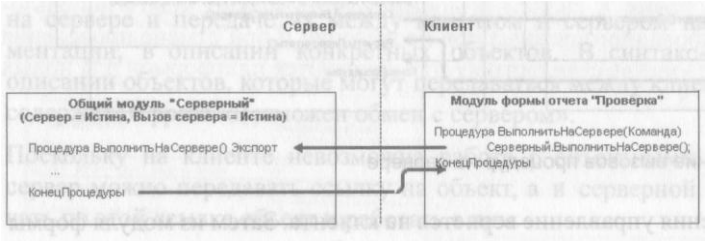


Рис. 5.30. Передача выполнения кода на сервер

Например, можно вызвать процедуру неглобального серверного общего модуля (с установленными свойствами Сервер и Вызов сервера) из клиентского обработчика команды в модуле формы (листинг 5.10).

Листинг 5.10. Передача исполнения кода на сервер путем вызова процедуры серверного общего модуля

```
&НаКлиенте
Процедура ВыполнитьНаСервере(Команда)
Серверный.ВыполнитьНаСервере();
КонецПроцедуры
```

После того как выполнение передано на сервер, все остальные вызовы процедур и функций будут выполняться также на стороне сервера (рис. 5.31).

Например, общий модуль Расчеты, который компилируется как на стороне сервера, так и на стороне клиента, содержит экспортируемую процедуру Пересчитать(). Как видно из рисунка, сначала на клиенте в модуле формы отчета вызывается экспортируемая процедура общего серверного модуля Серверный, и управление передается на сервер.

Затем из этой процедуры вызывается процедура Пересчитать(). При этом выполнение кода продолжится на сервере, и процедура Пересчитать () будет вызвана из серверного экземпляра модуля Расчеты.

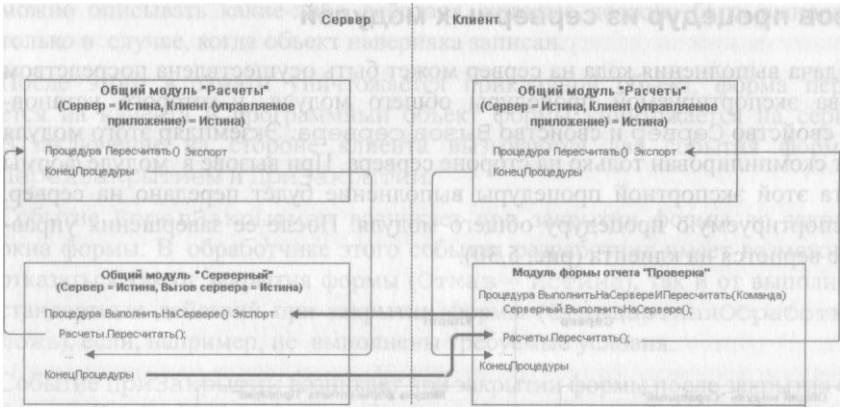


Рис. 5.31. Исполнение вызовов процедур на сервере

После ее завершения управление вернется на клиента. Затем из модуля формы снова вызывается процедура `Пересчитать()`. Однако теперь эта процедура будет вызвана из клиентского экземпляра модуля Расчеты.

Важно отметить, что разработчик не может «управлять» передачей выполнения с сервера обратно на клиента. Выполнение будет передано только после завершения выполнения вызванной процедуры или функции. Другими словами, если выполнение осуществляется на сервере, и вызываемая процедура не найдена в скомпилированном на стороне сервера коде, то будет выдано сообщение об ошибке даже в том случае, если вызываемая процедура присутствует в экземпляре, скомпилированном на стороне клиента (рис. 5.32).

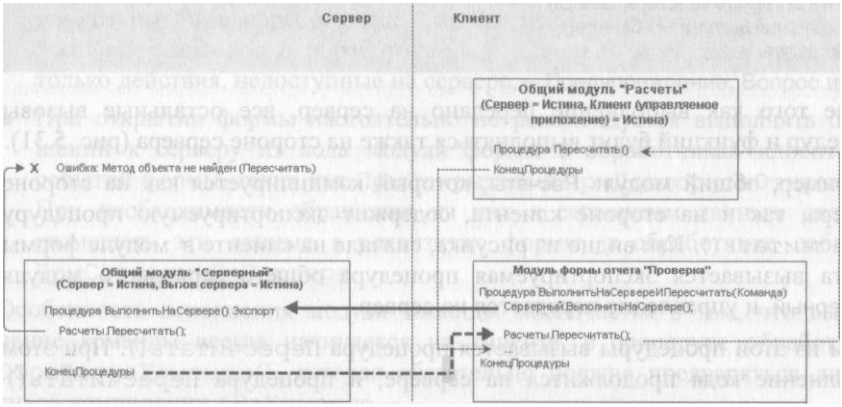


Рис. 5.32. Исполнение вызовов процедур на сервере

Передача параметров на сервер и возврат значений

При передаче выполнения кода на сервер, как правило, выполняется передача некоторых параметров вызываемой процедуры или функции. Также в случае вызова функции будет выполняться передача результата работы функции с сервера на клиента.

Важно учитывать, что не все типы значений и объекты, работа с которыми возможна на сервере, могут быть переданы с клиента на сервер и обратно.

Точная информация о возможности использования типов значений и объектов на сервере и передаче их между клиентом и сервером находится в документации, в описании конкретных объектов. В синтакс-помощнике при описании объектов, которые могут передаваться между клиентом и сервером, содержится фраза «Возможен обмен с сервером».

Поскольку на клиенте невозможна работа с объектными данными, то на сервер можно передавать ссылку на объект, а в серверной процедуре получать по этой ссылке объект и работать с ним.

Обмен с сервером возможен некоторым набором типов, для которых такой обмен поддерживается, а для обмена «специфическими» данными (например, файлами, картинками, табличными частями документа) предназначено временное хранилище. Методом `ПоместитьВоВременноеХранилище()` сериализуемое значение сохраняется во временном хранилище, а затем методом `ПолучитьИзВременногоХранилища()` значение извлекается из временного хранилища по указанному адресу. Например, временное хранилище может быть использовано для передачи данных между формами, минуя клиента, что особенно эффективно в случае больших объемов передаваемых данных.

Привилегированный и безопасный режимы исполнения кода

В процессе функционирования прикладного решения могут возникать ситуации, когда некоторые действия нужно выполнить максимально быстро, без проверки прав пользователя на используемые данные, а при выполнении некоторых действий, наоборот, нужно оградить систему от выполнения «ненадежного» кода, представляющего потенциальную угрозу для ее безопасности. Для этого платформа поддерживает привилегированный и безопасный режимы исполнения кода.

Привилегированный режим

В привилегированном режиме код исполняется на сервере без проверки прав. Использование привилегированного режима позволяет, во-первых, ускорить работу, так как не будут накладываться ограничения на доступ к данным, а во-вторых, позволяет выполнять операции с данными от лица пользователей, которым эти данные недоступны.

Для поддержки привилегированного режима в платформе существует несколько возможностей.

Для документов прямо в свойствах конфигурации можно задать использование привилегированного режима при проведении и при отмене проведения документа. Эти свойства стандартно устанавливаются платформой для новых документов (рис. 5.33).

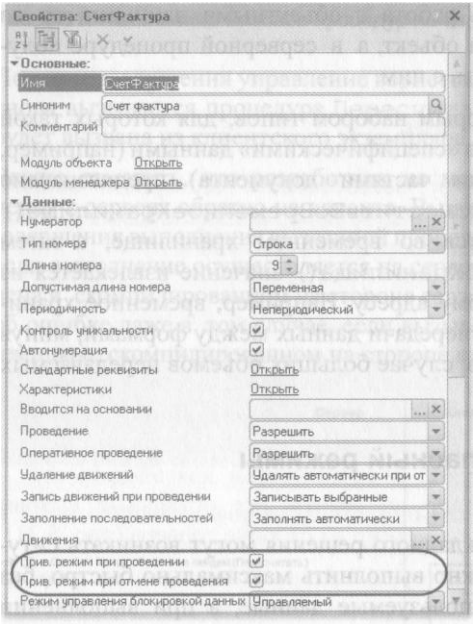


Рис. 5.33. Установка привилегированного режима при проведении и отмене проведения документа

Для бизнес-процессов также прямо в свойствах конфигурации можно задать использование привилегированного режима при создании задач. Эти свойства стандартно устанавливаются платформой для новых бизнес-процессов (рис. 5.34).

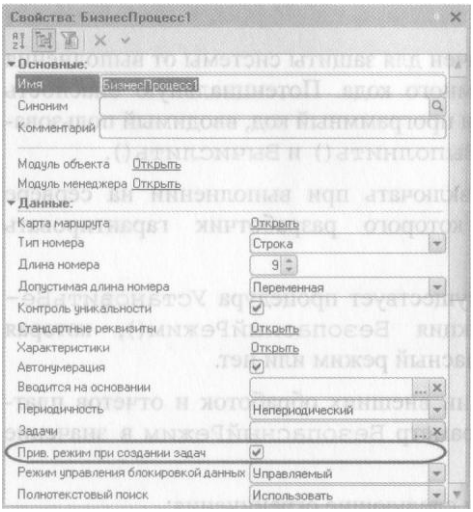


Рис. 5.34. Установка привилегированного режима при создании задач

Привилегированный режим исполнения кода можно включить/выключить средствами встроенного языка. Для этого в глобальном контексте существует процедура `УстановитьПривилегированныйРежим0`, а также функция `ПривилегированныйРежим()`, которая позволяет определить, включен привилегированный режим или нет.

Привилегированный режим рекомендуется использовать тогда, когда работа с данными от лица некоторого пользователя не нарушает установленные для этого пользователя права доступа.

Например, кладовщик имеет доступ к документу, регистрирующему продажи, но не имеет доступа к самому регистру продаж, в котором содержатся движения документа, возникающие при его проведении. В этом случае использование привилегированного режима при проведении документа не делает данные регистра продаж доступными кладовщику, но позволяет проводить документ и сохранять в регистре движения документа.

Можно также поместить код, не требующий проверки прав, в привилегированные общие модули, у которых установлено свойство `Привилегированный`. В такие модули могут быть перенесены операции, использующие данные, на которые у текущего пользователя нет прав.

Подробнее об этом можно прочитать в книге «1С:Предприятие 8.2. Руководство разработчика», раздел 5.5.4.2.

Безопасный режим

Безопасный режим работы предназначен для защиты системы от выполнения на сервере «ненадежного» программного кода. Потенциальную опасность представляют внешние обработки или программный код, вводимый пользователем для использования в методах `Выполнить()` и `Вычислить()`.

Безопасный режим рекомендуется включать при выполнении на сервере программного кода, надежность которого разработчик гарантировать не может.

Для этого в глобальном контексте существует процедура `УстановитьБезопасныйРежим()`, а также функция `БезопасныйРежим()`, которая позволяет определить, включен безопасный режим или нет.

А также при создании и подключении внешних обработок и отчетов платформа стандартно устанавливает параметр `БезопасныйРежим` в значение

`Истина`.

В безопасном режиме накладываются следующие ограничения:

- привилегированный режим отменяется, если он был установлен;
- попытки перехода в привилегированный режим игнорируются;
- запрещены операции с COM-объектами;
- запрещена загрузка и подключение внешних компонентов;
- запрещен доступ к файловой системе (кроме временных файлов);
- запрещен доступ к Интернету.

Подробнее об этом можно прочитать в книге «1С:Предприятие 8.2. Руководство разработчика», раздел 5.5.4.3.