**Информационная система**

**производственного участка «Фабрика кухни»**

**РУКОВОДСТВО АДМИНИСТРАТОРА**

Минск  
2021

Оглавление

[Введение 4](#_Toc88838516)

[Глава I. Установка и запуск 7](#_Toc88838517)

[Настройка параметров программы 16](#_Toc88838518)

[Глава II. Организация пользовательского 23](#_Toc88838519)

[интерфейса 23](#_Toc88838520)

[Главное окно программы 25](#_Toc88838521)

[Исследователь системы 31](#_Toc88838522)

[Форма просмотра 35](#_Toc88838523)

[Форма просмотра мастер-дитэйл 39](#_Toc88838524)

[Форма просмотра документа 42](#_Toc88838525)

[Форма в режиме выбора 42](#_Toc88838526)

[Поиск данных 44](#_Toc88838527)

[Диалоговое окно 60](#_Toc88838528)

[Управляющие элементы 63](#_Toc88838529)

[Глава III. Архитектура платформы 89](#_Toc88838530)

[Платформа Гедымин 97](#_Toc88838531)

[Структура программы на Гедымине 103](#_Toc88838532)

[Настройки 106](#_Toc88838533)

[Глава IV. Hello world! 109](#_Toc88838534)

[Обращаемся к базе данных 114](#_Toc88838535)

[Создаем экранную форму 116](#_Toc88838536)

[Заключение 123](#_Toc88838537)

[Глава V. Среда разработки 124](#_Toc88838538)

[Редактор скрипт-объектов 126](#_Toc88838539)

[Глава VI. Язык VBScript 145](#_Toc88838540)

[Основы VBScript 147](#_Toc88838541)

[Встроенные функции 172](#_Toc88838542)

[Обработка ошибок 201](#_Toc88838543)

[Использование TCreator 203](#_Toc88838544)

[Глава VII. Краткий курс SQL 207](#_Toc88838545)

[Введение в реляционные БД 213](#_Toc88838546)

[Глава VIII. Бизнес-объект 235](#_Toc88838547)

[Глава IX. Экранные формы 239](#_Toc88838548)

[Глава X. Документ 242](#_Toc88838549)

[Тип документа 247](#_Toc88838550)

[Глава XI. Константы 253](#_Toc88838551)

[Заключение 261](#_Toc88838552)

[Глава XII. Организация бухгалтерского 262](#_Toc88838553)

[Бухучет в рамках платформы Гедымин 265](#_Toc88838554)

[Журнал хозяйственных операций 274](#_Toc88838555)

[Связь между документом и проводками 275](#_Toc88838556)

[Практический пример 276](#_Toc88838557)

[Глава XIV. Структура базы 288](#_Toc88838558)

[GD\_VALUE 294](#_Toc88838559)

[GD\_GOODGROUP 294](#_Toc88838560)

[GD\_GOOD 295](#_Toc88838561)

[GD\_GOODBARCODE 295](#_Toc88838562)

[GD\_CONST 295](#_Toc88838563)

[GD\_CONSTVALUE 296](#_Toc88838564)

[AC\_ACCOUNT 297](#_Toc88838565)

[AC\_ACCVALUE 298](#_Toc88838566)

[AC\_COMPANYACCOUNT 298](#_Toc88838567)

[AC\_TRANSACTION 298](#_Toc88838568)

[AC\_TRRECORD 299](#_Toc88838569)

[AC\_RECORD 300](#_Toc88838570)

[AC\_ENTRY 300](#_Toc88838571)

[AC\_QUANTITY 301](#_Toc88838572)

[AC\_AUTOENTRY 301](#_Toc88838573)

[Глава XVI. Хранилище 303](#_Toc88838574)

[Глобальное хранилище 307](#_Toc88838575)

[Пользовательское хранилище 308](#_Toc88838576)

[Хранилище компании 308](#_Toc88838577)

[Хранилище рабочего стола 308](#_Toc88838578)

[Окно Хранилища 309](#_Toc88838579)

[Таблицы хранилища 313](#_Toc88838580)

[GD\_COMPANYSTORAGE 313](#_Toc88838581)

[GD\_GLOBALSTORAGE 313](#_Toc88838582)

[GD\_USERSTORAGE 314](#_Toc88838583)

[GD\_DESKTOP 314](#_Toc88838584)

[Работа с хранилищем из макросов 316](#_Toc88838585)

[Заключение 333](#_Toc88838586)

[Глава XVII. Безопасность 335](#_Toc88838587)

[Средства разграничения доступа сервера Interbase 336](#_Toc88838588)

[Разграничение прав доступа на платформе Гедымин 338](#_Toc88838589)

[Приложения 369](#_Toc88838590)

[Список горячих клавиш 374](#_Toc88838591)

# Введение

Краткое описание содержания глав.

**Глава 1. Установка и запуск** подробно описывает процесс инсталяции платформы на компьютере пользователя в автоматическом и ручном режимах. Объясняется различие между локальным и сетевым подключением к базе данных. Рассматриваются вопросы перехода с одной версии СУБД на более новую, обновления структуры базы данных и клиентской части платформы. Дается обзор опциональных настроек программы.

**Глава 2. Пользовательский интерфейс.** Здесь объясняются принципы построения пользовательского интерфейса Гедымина и основные способы взаимодействия с программой. Подробно рассматриваются основные типы экранных форм и управляющих элементов.

**Глава 3. Архитектура платформы** формирует у разработчика представление о внутреннем устройстве Гедымина. Рассматриваются основные типы объектов, на которых основана платформа и прикладные решения: бизнес-классы, объекты мета-данных, экранные формы, отчеты, макросы и т.д.

**Глава 4. Hello World!** Первая программа на Гедымине. Краткое знакомство с синтаксисом языка VBScript и средой разработки. Первое обращение к информации в базе данных.

**Глава 5. Среда разработки** содержит подробную информацию о среде разработке платформы Гедымин, в том числе о редакторе программного кода, отладчике и профайлере. При рассмотрении дизайнера экранных форм даются основные понятия визуального программирования.

**Глава 6. VBScript** является подробным руководством по синтаксису, типам данных, константам и встроенным функциям языка программирования Visual Basic Script Edition. Рассматриваются вопросы создания надежных программ, обработки ошибок и исключительных ситуаций, высвобождения выделенных ресурсов.

**Глава 7. Краткий курс SQL** предназначена для специалистов Глава 8. Бизнес-объект

Глава 9. Экранная форма Глава 10. Документ

Глава 11. Константы

Глава 12. Организация бухгалтерского учета Глава 13. Отчеты

Глава 14. Структура базы данных

Глава 15. Организация складского учета Глава 16. Хранилище

Глава 17. Безопасность

Приложения

# Глава I. Установка и запуск

## Настройка параметров программы

Основные параметры Гедымина можно просмотреть или изменить, если вызвать команду Опции из выпадающего меню Сервис, которое расположено в главном меню. Диалоговое окно Опции имеет четыре закладки: Общие, Аудит, Политики, Архивное копирование.

В зависимости от того, входит ли текущий пользователь в группу Администраторы или нет некоторые закладки и опции системы могут быть недоступными для изменения.

Рассмотрим содержимое каждой закладки.

#### Общие

##### Использовать Enter как Tab в диалогах

Во многих старых ДОСовских программах клавиша Enter служила для перемещения к следующему полю при вводе данных в экранную форму. Поскольку, многие пользователи испытывают определенный дискомфорт при переходе со старого текстового интерфейса на графический интерфейс операционной системы Windows в Гедымине предусмотрен режим эмуляции ДОС, при котором курсор перемещается между полями ввода в диалоговых окнах не только с помощью клавиши табуляции, но и при нажатии на клавишу Enter.

Для того, чтобы подтвердить ввод и закрыть окно при включенном режиме эмуляции используя клавиатуру необходимо либо установить фокус на кнопку Готово (Ок) и нажать Enter, либо воспользоваться комбинацией клавиш Ctrl+Enter. Обратите внимание, что если курсор находится в мемо поле, то нажатие клавиши Enter приведет к переходу на новую строку, а не к перемещению на следующее поле в окне.

Опция устанавливается только для текущего пользователя системы.

##### Применять магическое перемещение окон

Режим магического перемещения окон позволяет быстро и красиво расположить окна программы на рабочем столе. Если режим включен, то:

 При двойном щелчке на заголовке исследователя он занимает место слева под главным окном программы, причем его высота устанавливается во всю доступную высоту экрана;

 При двойном щелчке на заголовке формы просмотра, она занимает все доступное место на экране, ограниченное с двух сторон главным окном программы и окном исследователя;

 При изменении размеров формы просмотра, они будут автоматически выравниваться по другим формам, присутствующим на экране;

 При перемещении формы просмотра, ее положение будет автоматически выравниваться по другим формам, присутствующим на экране;

Если надо временно отключить режим магического перемещения, например, для того, чтобы быстро раскрыть окно на весь экран или переместить его произвольным образом, достаточно нажать и удерживать клавишу Ctrl во время выполнения соответствующих действий.

Опция устанавливается только для текущего пользователя системы.

##### Выводить лог при загрузке/сохранении в поток

При сохранении бизнес объекта на диске или загрузке его с диска в базу данных на экране отображается окно с информацией о ходе процесса. Если в нем нет особой нужды, просто снимите данную опцию.

Опция устанавливается только для текущего пользователя системы.

##### Отображать всплывающую подсказку в таблице

Если поле содержит данных больше, чем можно отобразить в клетке таблицы, то для того, чтобы просмотреть все данные необходимо навести указатель мыши на такую клетку и подождать около секунды. На экране появится всплывающая подсказка. Если она создает неудобства в работе, то ее можно отключить с помощью данной опции программы.

Опция устанавливается только для текущего пользователя системы.

##### Сохранять рабочий стол при выходе

Как следует из названия опции, если она установлена, то текущая конфигурация окон или, по-другому, рабочий стол, будет запоминаться при выходе из программы и восстанавливаться при следующем входе в нее.

Опция устанавливается только для текущего пользователя системы.

##### Сохранять значения полей в диалоговых окнах

Включение данной опции позволяет облегчить и упростить ввод большого объема однородных данных. Значения, введенные пользователем при добавлении записи, будут запоминаться и подставляться в соответствующие поля при вводе следующей записи.

Опция устанавливается только для текущего пользователя системы.

##### Разрешать скрытие главной панели

Данная опция влияет на то, как будет вести себя программа, если в окне, отображающем данные, связанные как мастер-дитэйл, скрыть главную панель. Если опция установлена, то Гедымин запомнит состояние панели и при следующей загрузке она будет скрыта. Если нет — то при следующей загрузке программы главная панель будет открыта. Напомним, что скрытие главной панели позволяет отобразить в детальном списке все имеющиеся в базе данных записи, а не только те, которые относятся к конкретной записи в главном списке.

Опция устанавливается только для текущего пользователя системы.

##### Показывать нули в таблице

Если опция не установлена, то для числовых полей в таблицах не будут выводиться значения, если они равны нулю.

Опция устанавливается только для текущего пользователя системы.

##### При загрузке Гедымина переключать клавиатуру на

Данная опция позволяет выбрать язык ввода, на который будет переключаться операционная система при загрузке Гедымина.

Опция устанавливается только для текущего пользователя системы.

##### При смене рабочей организации рабочий стол

С помощью данной опции можно настроить поведение рабочего стола при смене активной рабочей организации: рабочий стол будет сохраняться, не сохраняться или пользователь будет запрашиваться о необходимом действии.

Опция устанавливается только для текущего пользователя системы.

##### Запрещать ввод дублирующихся организаций

Серьезной проблемой при эксплуатации системы является появление двух и более записей в базе данных, представляющих один и тот же объект. Например, один пользователь может ввести организацию как ООО «Рога и Копыта», а второй — «Рога и Копыта» ООО. Гедымин позволяет избежать

появления дублирующихся записей путем проверки информации при вводе.

Опция устанавливается для всех пользователей системы.

###### Проверять организацию по УНН (ИНН)

Проверка будет осуществляться по коду УНН (ИНН, УНП) организации. Если в базе уже есть организация с кодом, как у вновь вводимой, то на экран будет выдано предупреждение. Пользователь может настоять на сохранении записи с дублирующимся кодом или вернуться в окно редактирования с тем, чтобы изменить код или вообще отказаться от ввода новой записи.

Опция устанавливается для всех пользователей системы.

###### Проверять организацию по наименованию

Аналогично проверке по коду УНН, только проверка будет осуществляться по наименованию организации.

Опция устанавливается для всех пользователей системы.

##### Проверять корректность банковского счета

Если проверка корректности номера банковского счета включена, то при попытке ввести некорректный номер счета пользователю будет выдано предупреждение.

В Беларуси применяются 13-тизначные банковские счета, в номере которых первые 12 цифр содержат, собственно, номер счета, а последняя, тринадцатая, является контрольной, рассчитываемой особым образом на основании 12-ти предыдущих и трехзначного кода банка. Таким образом, вероятность того, что ошибочно введенный номер счета пройдет проверку на корректность не превышает 10%.

Очевидно, что данную проверку стоит отключить при эксплуатации программы за пределами Беларуси.

Опция устанавливается для всех пользователей системы.

##### Окно дат, при проверке на корректность

Часто, пользователи допускают ошибки при вводе дат, набирая не тот год, например, 2094 вместо 2004. Количество таких ошибок можно существенно снизить, если установить окно корректных дат.

Опция устанавливается для всех пользователей системы.

##### Предупреждать об отсутствии прав на изменение записи Запрашивать подтверждения:

##### На изменение нескольких записей

Гедымин позволяет вносить изменения в несколько записей одновременно. Для этого необходимо выделить нужные строки в таблице удерживая клавишу Ctrl и щелкая по ним левой кнопкой мыши или используя Shift и перемещаясь с помощью стрелок на клавиатуре и выбрать команду Изменить. Если данная опция включена, то при попытке открыть для редактирования более одной записи на экран будет выдано соответствующее предупреждение.

Опция устанавливается только для текущего пользователя системы.

###### На изменение формы не под администратором

В Гедымине можно изменить (настроить) любую экранную форму. Изменения, выполненные под учетной записью администратора, распространяются на всех пользователей, в то время как изменения, сделанные под любой другой учетной записью будут действовать только для нее. Соответствующее предупреждение будет выведено на экран, при попытке изменить экранную форму не под администратором, если рассматриваемая опция включена.

Опция устанавливается только для текущего пользователя системы.

###### Прочие подтверждения

Данная опция позволяет управлять выводом на экран всех прочих предупреждений.

Опция устанавливается только для текущего пользователя системы.

##### Связь по LB, RB мастера для интервальных деревьев

Как известно, в интервальных деревьях связь между родительской и дочерними записями можно установить по полям LB и RB (левой и правой границе интервала родительской записи). В этом случае будут отобраны все дочерние записи (включая дочерние дочерних, дочерние дочерних дочерних и т.д.), относящиеся к данной родительской. С технической точки зрения отбор может быть выполнен двумя способами: в детальный запрос передаются непосредственно значения левой и правой границ родительской записи и в детальный запрос передается идентификатор родительской записи, которая через JOIN по условию вхождения в интервал объединяется с дочерними записями. Очевидно, что первый способ приведет к более быстрому запросу на извлечение дочерних записей (на

один JOIN меньше), зато второй способ гарантирует всегда правильное извлечение дочерних записей, даже если границы родительской записи изменились, например, вследствие добавления новой записи другим пользователем системы, а датасет не был обновлен.

Если опция включена, то будет использоваться связь по непосредственным значениям границ интервала родительской записи. В противном случае, связь будет устанавливаться по идентификатору родительской записи.

Опция устанавливается для всех пользователей системы.

##### Предупреждать о возможном несоответствии детальной записи главной записи

Если опция активна и в детальный объект добавляется запись, которая не соответствует текущей записи в главном объекте, то на экран будет выдано предупреждение о том, что добавленная запись не будет видна в списке («исчезнет» из него) после обновления, перечитывания данных.

Опция устанавливается только для текущего пользователя системы.

##### Всегда запрашивать параметры для текущего фильтра и предупреждать о фильтрации данных

Если данная опция установлена, то при открытии формы просмотра, данные которой отфильтрованы с помощью фильтра с параметрами, на экран будет выведено окно для ввода этих параметров. Если фильтр не параметризованный, то на экран будет выведено предупреждение о том, что список отфильтрован.

Если опция не активна, то пользователь не будет предупрежден о примененной фильтрации и параметрам, если они требуются, будут присвоены предыдущие, сохраненные значения.

Данная опция была введена в программу, когда разработчики столкнулись с большим количеством обращений от пользователей вроде:

«куда пропали мои данные?» или «я вчера ввел документ, а сегодня не вижу его в списке».

Напомним, что если к данным применена фильтрация, то пиктограммка

«воронка» на панели инструментов меняет свой цвет с серого на красный, и название фильтра выводится в строке состояния окна. Более подробная информация о примененном фильтре выводится во всплывающей подсказе, если навести указатель мыши на кнопку фильтрации на панели инструментов или на строку состояния.

Опция устанавливается только для текущего пользователя системы.

##### Предупреждать, если часть полей ввода в диалоговом окне скрыта

В сложных диалоговых окнах сверху может отображаться набор закладок с полями ввода, а под ними — таблица с данными. Размер верхней и нижней части можно изменять, перетягивая разделительную полосу вверх- вниз. Можно уменьшить размер верхней части так, что не все поля будут видны, что, в свою очередь, может привести к нежелательным ошибкам, при вводе данных пользователем. Если данная опция включена, то в таких случаях, при открытии диалогового окна, будет выдаваться предупреждение и предложение пользователю автоматически скорректировать размеры верхней и нижней частей окна с тем, чтобы все поля были видны на экране.

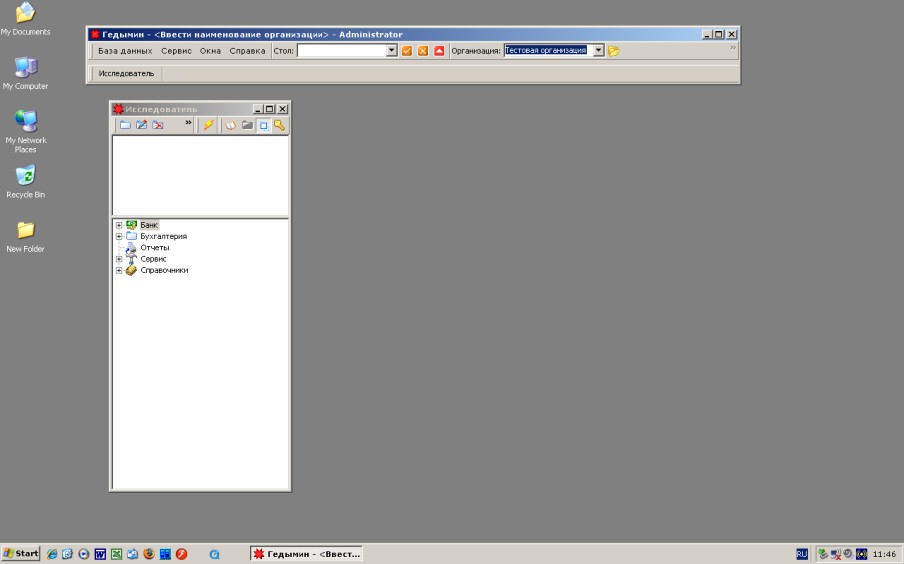
Опция устанавливается только для текущего пользователя системы.

# Глава II. Организация пользовательского

# интерфейса

В этой главе мы познакомимся с организацией пользовательского интерфейса платформы Гедымин.

Запустим Гедымин и подключимся к демонстрационной базе данных под учетной записью **Administrator**. Если это наш первый вход в программу, то мы увидим на экране следующую картину:



*Рис. 3. Вид экрана при первом запуске Гедымина*

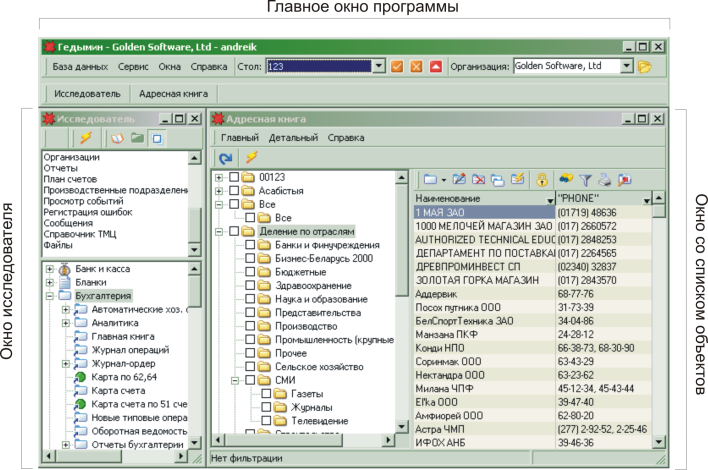
Два отображенных окна – это основные окна платформы: **Главное окно программы** и **Исследователь системы**.

Двойным щелчком по заголовку Главного окна можно заставить его расположиться в верхней части экрана, вытянувшись во всю его ширину. Аналогичные действия с окном Исследователя заставят его прижаться к левому краю экрана и растянуться на всю доступную высоту. Обратите внимание, что Исследователь не раскрылся на весь экран, а занял привычное для себя положение. Если мы начнем открывать другие окна, они займут свободное пространство ниже Главного окна и правее Исследователя, не перекрывая их. Такое поведение окон называется **Магическим перемещением**. Его можно отключить в опциях системы.

Совет: Если при включенном режиме **Магического перемещения** требуется открыть окно на весь экран, то в момент двойного щелчка по заголовку следует удерживать нажатой клавишу **Ctrl**.

Пользовательский интерфейс Гедымина организован в стиле SDI (Single Document Interface). Его прообразом послужила среда программирования Borland Delphi 5. Главное окно программы располагается вверху экрана, занимает всю его ширину и имеет небольшую высоту. Справочники, списки документов, отчеты и др. располагаются обособленно от главного окна программы, каждый в своем окне на рабочем столе Windows. Размер окон может быть изменен, при этом становится доступной часть рабочего стола или окна других программ, располагающиеся под Гедымином. Такое устройство многооконного интерфейса особенно удобно, если необходимо организовать одновременную работу с несколькими приложениями. Например, работать с Гедымином и иметь перед глазами данные, находящиеся в электронной таблице Excel.

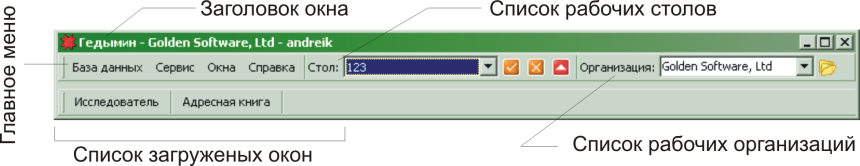
Обычно, при работе с Гедымином экран выглядит следующим образом:



*Рис. 4. Вид экрана при работе с Гедымином*

## Главное окно программы

Главное окно программы предназначено для вызова основных команд, отображения информации о текущей рабочей организации и текущем пользователе системы, выбора и управления рабочими столами, выбора или изменения текущей рабочей организации, а также для отображения списка загруженных в настоящий момент времени окон.



*Рис. 5. Главное окно программы*

Ниже приведено описание каждой из составляющих главного окна:

#### Заголовок окна

Заголовок главного окна содержит наименование программы, наименование текущей рабочей организации, а также наименование учетной записи пользователя.

#### Главное меню

Главное меню состоит из четырех пунктов: База данных, Сервис, Окна и Справка. Ниже приводится описание команд из каждого пункта:

##### База данных

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| Подключенные пользователи | Выводит на экран окно со списком подключенных к базе данных пользователей. Команда доступна только для пользователя,  входящего в группу **Администраторы**. |
| Подключиться к базе данных | Команда позволяет выполнить подключение к другой базе данных не перезапуская Гедымин. Перед подключением необходимо отключиться от текущей базы данных с помощью команды **Отключиться от базы**  **данных**. Команда доступна только для пользователя, входящего в группу  **Администраторы**. |
| Отключиться от базы данных | Закрывает текущее соединение с базой данных. Команда доступна только для пользователя, входящего в группу  **Администраторы**. |
| Подключиться в однопользовательском  режиме | Позволяет подключиться к базе данных в однопользовательском режиме. Перед  подключением необходимо отключиться от |

|  |  |
| --- | --- |
|  | текущей базы данных с помощью команды  **Отключться от базы данных**.  Внимание: подключиться к базе данных в однопользовательском режиме можно только под учетной записью **Administrator**. |
| Вернуть базу данных в нормальный режим | Команда позволяет вернуть базу данных в многопользовательский режим. Перед выполнением команды необходимо сначала  выполнить отключение от базы данных. |
| Архивное копирование | Открывает окно создания архивной копии базы данных. Доступна только для пользователя из групп **Администраторы** и  **Операторы архива**. |
| Восстановление базы данных | Открывает окно восстановления базы данных из архивной копии. Доступна только для пользователя из групп **Администраторы** и  **Операторы архива**. |
| Обновление статистики индексов | Обновляет статистику индексов3. Команда доступна только для пользователя,  входящего в группу **Администраторы**. |
| Выход | Выход из программы. Обратите внимание, что если запущен процесс архивирования или восстановления базы данных из архива, то выйти из программы нельзя. Необходимо  сначала дождаться завершения процесса. |

**Администраторы**. Такой учетной записи будут полностью доступны4 все

Обратите внимание, что любая учетная запись может входить в группу

3 Статистика индекса — это число в диапазоне от 0 до 1, которое показывает как распределены значения данных в колонке, по которой создан индекс. Статистика используется оптимизатором при создании плана выполнения запроса. Сервер Interbase не обновляет статистику индекса при изменении, добавлении или удалении данных, поэтому рекомендуется периодически выполнять команду Обновление статистики индексов или выполнять архивное копирование и последующее восстановление базы данных. Неверные значения статистики индексов могут привести к генерации неоптимальных планов и, соответственно, к снижению производительности работы системы.

4 Полный доступ подразумевает возможность создания, изменения, удаления объекта, печати его данных, а также назначение прав доступа.

объекты справочников, документы, проводки, а также большинство команд из меню. Но, такие действия, как операции над структурой базы данных, создание новых экранных форм, подключение в однопользовательском режиме доступны только под учетной записью **Administrator**. Последняя всегда входит в группу **Администраторы**.

**Сервис**

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| Редактор скрипт- объектов… | Открывает окно **Редактора скрипт объектов**.  Команда доступна только для пользователя, входящего в группу **Администраторы**. |
| Редактор форм… | Открывает окно **Редактора экранных форм**. Команда доступна только под учетной записью  **Administrator**. |
| SQL редактор… | Открывает окно **Редактора SQL запросов**. Команда доступна только для пользователя,  входящего в группу **Администраторы**. |
| Просмотр шаблона документов… | Открывает окно тестирования шаблонов импорта текстовых документов. Команда доступна только для пользователя, входящего в  группу **Администраторы**. |
| Установить пакеты настроек… | Открывает окно установки пакетов настроек. Команда доступна только для пользователя,  входящего в группу **Администраторы**. |
| Опции… | Открывает окно с опциями системы. |

**Окно**

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| Исследователь | Открывает окно **Исследователя системы**. Для команды назначено сочетание горячих клавиш  Ctrl+F11. |
| Список окон… | Открывает список загруженных в данный  момент времени окон. |
| Очистить память | Удаляет из памяти все окна просмотра, которые загружены в данный момент но не видимы на экране (т.е. находятся в закрытом  состоянии). |
| Сохранить стол… | Команда позволяет сохранить текущий  рабочий стол. |
| Удалить стол | Команда позволяет выбрать из списка и |

|  |  |
| --- | --- |
|  | удалить один из сохраненных ранее рабочих  столов. |

**Справка**

Как легко догадаться из названия, здесь собраны команды вызова электронной справки. На сегодняшний день доступна документация по платформе, языку программирования VBScript и системе построения отчетов FastReport. Пункт меню **Регистрация** отвечает за привязку копии программы к компьютеру. И, наконец, последний пункт – **О системе…** – не такой уж и бесполезный как это может показаться на первый взгляд. Диалоговое окно, которое вызывается данной командой, кроме удовлетворения известного тщеславия разработчиков содержит массу полезной информации: о версии и размещении исполняемого модуля, клиентской библиотеке и сервере базы данных, о версии файла базы данных и параметрах текущего подключения и т.п. Данная информация пригодится вам при обращении в службу технической поддержки для устранения возможных неполадок.

#### Список рабочих столов

Справа от главного меню располагается выпадающий список рабочих столов, а также кнопки сохранения текущего рабочего стола, удаления рабочего стола и очистки экрана. Последняя команда закрывает и удаляет из памяти все окна за исключением **Исследователя системы**.

##### Что такое рабочий стол?

Рабочим столом, в терминах платформы Гедымин, называется совокупность окон просмотра, расположенных определенным образом на экране. Рабочему столу можно присвоить имя и сохранить его в базе данных.

Например, разработчик регулярно использует Редактор скрипт- объектов, Редактор SQL и окно со списком таблиц. Для того, чтобы каждый раз при запуске программы не открывать вышеупомянутые окна вручную, можно, единожды открыв и расположив их удобным образом на экране, сохранить затем рабочий стол под именем «Разработка». При следующей загрузке Гедымина сохраненные окна будут открыты автоматически5. Если разработчик, к тому же, обращается и к бухгалтерским функциями

5 В опциях системы можно отключить параметр автоматической загрузки рабочего стола при входе в программу.

платформы, он может создать второй рабочий стол, назвав его

«Бухгалтерия», и включить в него такие окна, как «Журнал-ордер»,

«Журнал хозяйственных операций» и «План счетов». В любой момент времени можно перейти от одного рабочего стола к другому, выбрав его из выпадающего списка.

По умолчанию, рабочий стол сохраняется только при вызове соответствующей команды, однако установив соответствующий параметр в опциях системы можно добиться автоматического сохранения текущего рабочего при выходе их программы. Там же можно определить, как будет вести себя рабочий стол при смене рабочей организации: сохраняться автоматически, не сохраняться или необходимое действие будет запрашиваться у пользователя.

Обратите внимание, что список рабочих столов свой не только для каждого пользователя, но и для каждого разрешения экрана.

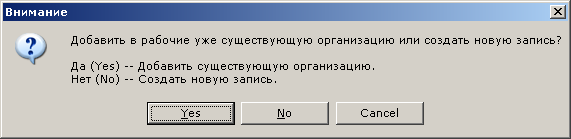
В ряде случаев требуется запретить пользователю самостоятельно изменять рабочий стол. Сделать это можно, применив соответствующую политику безопасности на закладке **Политики** в окне **Опции**, которое вызывается из пункта **Сервис** главного меню.

#### Список рабочих организаций

На платформе Гедымин можно вести учет по нескольким организациям в рамках одной базы данных. Организация, по которой ведется учет называется **Рабочей организацией**. Активная в данный момент времени рабочая организация называется **Текущей рабочей организацией**. Ее наименование отображается в поле ввода выпадающего списка, который служит для смены текущей рабочей организации.

##### Управление списком рабочих организаций

Справа от списка расположена кнопка, с помощью которой можно открыть окно просмотра рабочих организаций. Обратите внимание, что при добавлении записи, система спросит нас: хотим ли мы создать совершенно новую организацию или сделать рабочей одну из компаний из справочника клиентов.



*Рис. 6. Добавление рабочей организации*

#### Список загруженных окон

Нижнюю часть главного окна занимает список загруженых в настоящий момент времени окон. Загруженное в память окно не обязательно видно на экране, оно может быть скрыто (свернуто) или находиться под другими окнами. Скрытое окно можно отобразить, щелкнув мышью по его названию в списке.

##### Свернуть или закрыть?

Если щелкнуть правой кнопкой мыши на списке загруженных окон, то откроется контекстное меню с четырьмя командами: **Свернуть форму**, **Свернуть все**, **Закрыть форму**, **Закрыть все**.

Первые две команды позволяют свернуть, т.е. убрать с экрана, конкретную форму6 или все видимые, открытые формы. Для того, чтобы снова отобразить свернутую форму на экране необходимо кликнуть по ее наименованию в Списке загруженных окон. Команды **Закрыть форму** и **Закрыть все** не просто убирают форму с экрана, но и удаляют ее из памяти, высвобождая тем самым ресурсы компьютера. После закрытия формы, ее наименование уже не будет высвечиваться в Списке загруженных окон и для повторного открытия необходимо будет прибегнуть к **Исследователю**.

В отличие от других программ Windows, в Гедымине при щелчке по пиктограммке закрытия окна (крестик на заголовке, в правом верхнем его углу) окно не закрывается, а сворачивается, т.е. остается в оперативной памяти компьютера. Так сделано, чтобы уменьшить нагрузку на сервер баз данных. Ведь, при открытии окна может выполняться длительный запрос, а пользователь может часто открывать-закрывать окна, например, при работе с неколькими документами.

Если необходимо не свернуть, а именно закрыть окно (т.е. удалить его из памяти компьютера), то следует удерживать нажатой клавишу **Shift** в момент щелчка по пиктограммке закрытия.

## Исследователь системы

Через Исследователь осуществляется доступ ко всем объектам системы. Его содержимое зависит от загруженных на базу данных настроек. Окно Исследователя, как правило, располагается в левой части экрана (см. Рис. 4).

6 Вместо слова "окно" мы будем часто использовать словосочетание "экранная форма" или просто "форма".

Если оно скрыто, воспользуйтесь коомбинацией клавиш **Ctrl+F11** или командой из меню **Окна** главного окна программы.

В верхней части окна располагается панель инструментов, под ней – список последних команд и, наконец, большую часть окна занимает древовидный иерархический список бизнес-объектов и команд. Для выполнения команды или открытия формы просмотра бизнес-объекта необходимо щелкнуть по элементу списка двойным щелчком левой клавиши мыши или установить на него курсор и нажать на клавиатуре **Enter**.

Первое обращение к команде Исследователя создаст и откроет на экране сооветствующую форму просмотра. Последующие обращения к этой же команде приведут к эффекту, аналогичному нажатию на кнопку в списке загруженных окон. Т.е. активизируют уже существующую форму. Создать отдельный экземпляр формы можно, если при повторном обращении к команде Исследователя нажать и удерживать клавишу **Ctrl**.

##### Список команд

Список команд хранится в базе данных. Технически, команда Исследователя является обычным бизнес-объектом типа **TgdcExplorer**, а окно Исследователя – слегка модифицированной формой просмотра (предназначение и принципы работы с экранными формами просмотра подробно рассмотрены в следующем подразделе). Из этого следует, что пользователь способен самостоятельно изменять, добавлять или удалять команды исследователя7.

Команды Исследователя бывают четырех видов:

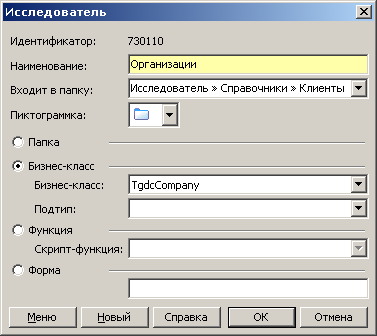
Папка используется для тематической группировки команд; Бизнес-класс открывает форму просмотра указанного бизнес-

класса;

Функция выполняет указанную скрипт-функцию; Форма открывает на экране окно указанного типа.

Ниже приведено диалоговое окно для редактирования пункта Исследователя и дано описание его полей:

7 Только пользователи из группы Администраторы имеют право выполнять указанные действия.



*Рис. 7. Диалоговое окно редактирования команды Исследователя*

###### Наименование Описание

Идентификатор Идентификатор записи в базе данных. Наименование Наименование команды Исследователя.

Входит в папку Папка в дереве Исследователя, куда входит команда.

Пиктограммка Графическое изображение, которое отображается слева от наименования пункта в дереве Исследователя.

Папка Если радио-кнопка выбрана, то редактируемая команда является Папкой.

Бизнес-класс Если радио-кнопка выбрана, то редактируемая команда предназначена для отображения окна просмотра бизнес-класса. Имя класса и, при необходимости, его подтип задаются в следующих двух полях.

Функция Если радио-кнопка выбрана, то редактируемая команда предназначена для вызова указанной скрипт- функции. Имя скрипт-функции задается в следующем поле.

Форма Если радио-кнопка выбрана, то редактируемая команда предназначена для отображения на экране указанной формы. Имя класса окна задается в следующем поле.

Еще одно важное предназначение Исследователя – это разграничение прав доступа к бизнес-классам. Более подробно мы поговорим об этом в главе "Безопасность".

Внимание! Следует быть осторожным с удалением команд из Исследователя, так как в этом случае невозможно будет настроить права доступа на соответсвующий объект. Никогда не удаляйте команды из эстетических побуждений, например, если вам кажется, что их слишком много на экране.

Некоторые из команд Исследователя не являются частью какой-либо настройки, а "зашиты" непосредственно в платформу. К таковым относятся:

Раздел **Банк**. Команды: **Выписки по р\с** и **Картотека по р\с**8. Раздел **Бухгалтерия**. Команды: **Автоматические хозяйственные операции**, **Анализ счета**, **Главная книга**, **Журнал хозяйственных операций**, **Журнал-ордер**, **Карта счета**, **Оборотная ведомость**, **План счетов**, **Типовые хозяйственные операции**.



 Команда **Отчеты**. Выводит на экран список всех отчетов, загруженных в базу данных. Следует заметить, что рядовой пользователь вызывает отчет из соответствующего меню на форме просмотра. Команда **Отчеты** используется в основном администратором или разработчиком, в тех редких случаях, когда надо разыскать затерявшийся отчет или выполнить операцию (обычно, назначение прав доступа) над группой отчетов.

 Раздел **Сервис**.

 Подраздел **Администратор**. Содержит команды для работы с учетными записями: **Пользователи** и **Группы пользователей**. Команда **Журнал событий** открывает журнал регистрации действий пользователя и событий системы. Подробно про учетные записи и аудит действий пользователя читайте в главе "Безопасность".

 Подраздел **Аттрибуты**. Содержит команды для работы с метаданными и настройками: **Домены**, **Исключения**, **Настройки**, **Представления**, **Процедуры**, **Таблицы**.

 Прочие команды из раздела **Сервис**: Исследователь, Константы, Прикрепления, Региональные установки, Типовые документы, Файлы, Фильтры, Хранилище.

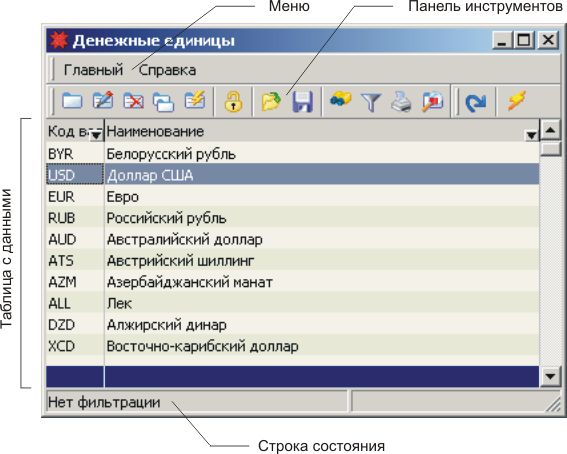
8 Рудименты доисторической эпохи, когда Гедымин еще не имел четкого разграничения на платформу и прикладной код (настройки).

 Раздел **Справочники**. Данный раздел позволяет обратиться к основным справочникам, реализованным на уровне платформы: Административно-территориальных единиц, Государственных праздников, Графиков рабочего времени, Должностей, Клиентов, Валют, ТМЦ, Типов банковских счетов и т.д.

## Форма просмотра

Форма просмотра предназначена для отображения на экране списка бизнес-объектов одного типа. В зависимости от структуры бизнес-объекта форма может быть простой или сложной. В первом случае форма содержит одну таблицу или дерево, а во втором -- два и более связанных между собой набора данных.

Начнем рассмотрение со случая, когда окно содержит единственную таблицу. Такие формы используют простейшие справочники, вроде справочника Валют, Единиц измерения, Назначений платежа и т.д. Откроем окно **Справочника валют** с помощью команды **Валюты** в разделе Справочники-Курсы валют Исследователя системы. Как показано на Рис. 8, форма состоит из: Меню, Панели инструментов, Таблицы с данными и Строки состояния.



*Рис. 8. Форма просмотра с единственной таблицей.*

Наименование типа объекта, с которым мы имеем дело, выводится в заголовке окна.

#### Меню

Меню содержит два пункта: Главный9 и Справка. В первом – сосредоточены команды для работы с бизнес-объектом. Название второго говорит само за себя.

Опишем каждую из команд меню **Главный**:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | **Команда** | **Быстрый**  **вызов** | **Описание** |
|  | | Добавить… | Ins | Открывает диалоговое окно ввода  новой записи. |
|  | | Изменить… | Ctrl+Enter, Enter | Открывает диалоговое окно для изменения данных выделенной  записи. |
|  | | Удалить | Ctrl+Del | Удаляет выделенную запись. |
|  | | Дубликат | Ctrl+D | Создает копию текущей записи и открывает ее в окне для  редактирования. |
|  | | Слияние… | Ctrl+R | Открывает окно слияния  (объединения) двух записей. |
|  | | Добавить в отмеченные |  | Добавляет выделенную запись в отмеченные. Используя команду **Только отмеченные** можно работать со списком отобранных  ранее записей. |
|  | | Удалить из  отмеченных |  | Убирает выделенную запись из  списка отмеченных. |
|  | | Загрузить из файла… |  | Открывает ранее сохраненный файл и загружает из него записи в базу  данных. |
|  | | Сохранить в  файл… |  | Сохраняет выделенные записи в  файле на диске. |
|  |  | Сохранить |  | При редактировании в таблице |

9 Название "Главный" выглядит несколько неуклюже и в данном контексте лишено смысла, ибо объект только один. Вторая часть – "Детальный" – появляется в более сложных формах, где пристутствует два связанных между собой набора данных.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | подтверждает изменения текущей  записи, сохраняет их в базе данных10. |
|  | Поиск | Ctrl+S | Открывает или закрывает панель  поиска. |
|  | Фильтр |  | Открывает меню фильтров. При акивном фильтре пиктограммка окрашена в красный цвет. Если подвести курсор мыши задержать его на пол-секунды, в окошке всплывающей подсказки будут показаны параметры установленного  фильтра. |
|  | Печать | Ctrl+P | Открывает меню отчетов формы. |
|  | Макросы | Ctrl+M | Открывает меню с макросами  формы. |
|  | Только отмеченные |  | Включает/выключает режим отображения только отмеченных  записей. |
|  | Прикрепить… |  | Позволяет прикрепить к выбранному объекту любой произвольный объект или файл. Ранее прикрепленные объекты  отображаются в выпадающем меню. |
|  | Добавить в  настройку… |  | Позволяет добавить выбранные  записи в указанную настройку. |
|  | Загрузить  настройки формы… |  | Позволяет скопировать визуальный  настройки формы из профиля другого пользователя системы. |
|  | Распространить настройки |  | Распространяет текущие визуальные  настройки формы на всех пользователей системы. |

10 Команда **Сохранить** имеет еще один смысл. Идеология Гедымина подразумевает, что по завершении редактирования данные записываются в базу и транзакция закрывается. Если разработчик допустил в коде ошибку и по выходу из диалогового окна или по завершении выполнения макроса транзакция осталась открытой, пользователь может вручную закрыть и подтвердить ее с помощью команды **Сохранить**.

#### Панель инструментов

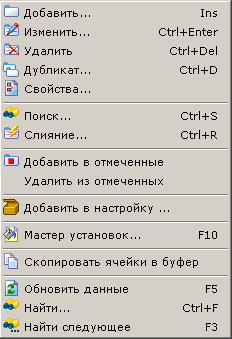
На панели инструментов представлены наиболее часто используемые команды. Кроме основной панели, существует еще и дополнительная, наполнение которой зависит от конкретного бизнес-объекта. Например, в окне со списком платежных поручений, на дополнительной панели выводится список банковских счетов текущей организации.

В дополнение к командам, описанным в предыдущем подразделе, на панели инструментов присутствует кнопка  переключения режима редактирования в таблице.

#### Таблица с данными

Пространство, ограниченное панелью инструментов сверху и строкой состояния снизу, называется **Рабочей областью**. Обычно здесь располагается таблица, хотя, при создании своих окон, разработчик волен разместить здесь любой управляющий элемент для отображения и редактирования данных.

По щелчку правой кнопки мыши над таблицей можно вызвать контекстное меню, которое содержит основные команды по работе с данными, а так же команды поиска в таблице и настройки ее внешнего вида.



*Рис. 9. Контекстное меню формы просмотра*

В дополнение к командам меню формы и панели инструментов контекстное меню содержит следующие команды:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | **Команда** | **Быстрый**  **вызов** | **Описание** |
|  | | Свойства… |  | Открывает диалоговое окно  **Свойства бизнес-объекта**. |
|  |  | Обновить | F5 | Перечистывает данные с сервера. |
|  | | Найти… | Ctrl+F | Открывает окно поиска в таблице. |
|  | | Найти следующее | F3 | Находит следующее значение,  удовлетворяющее критериям поиска. |

#### Строка состояния

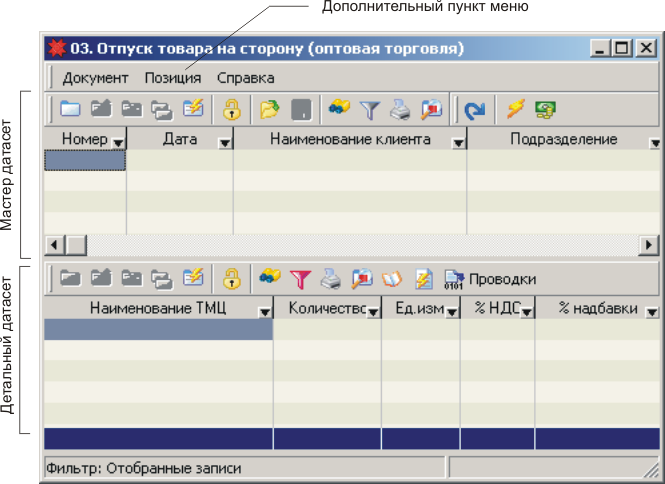
В зависимости от того, применен к данным фильтр или нет, в строке состояния выводится либо наименование примененного фильтра, либо надпись «Нет фильтрации». В первом случае, более подробную информацию о фильтре можно получить, наведя указатель мыши на строку состояния и обождав пока не появится всплывающая подсказка.

#### Сохранение визуальных настроек формы

Визуальные настройки формы сделанные пользователем автоматически сохраняются при ее закрытии. Иногда, возникает необходимость открыть форму в чистом, изначальном виде. Для этого необходимо в момент щелчка мышью по команде **Исследователя** удерживать клавишу **Shift** в нажатом положении.

## Форма просмотра мастер-дитэйл

Внешний вид форм просмотра мастер-дитэйл различается в зависимости от используемых визуальных компонентов (таблица или дерево) и их взаимного расположения. Возможные компоновки сложных форм просмотра приведены на рисунках ниже.



*Рис. 10. Окно просмотра мастер-дитэйл в горизонтальном исполнении*



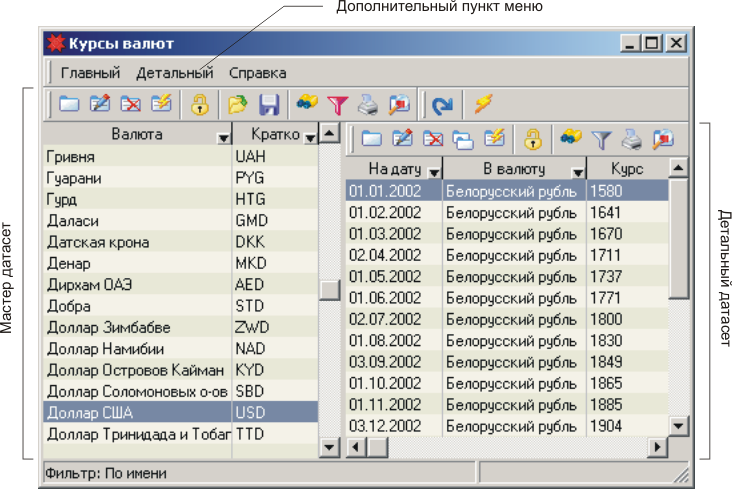
*Рис. 11. Окно просмотра мастер-дитэйл с деревом*

Обратите внимание, что в форме с деревом во избежание путаницы не выводится панель инструментов для главного набора данных. Для добавления, изменения или удаления элементов используйте меню **Главный** или контекстное меню по правой кнопке мыши.

Для формы с деревом главное меню содержит два дополнительных переключателя:

|  |  |
| --- | --- |
| Вложенные уровни | Если режим активирован, то в правой части окна выводятся как те записи, которые непосредственно относятся к выделенному пункту, так и ко всем вложенным его  подпунктам. |

|  |  |
| --- | --- |
| Разрешить перетаскивание | Если режим включен, то пользователь может с помощью мышки перетаскивать элементы дерева, менять их иерархию. Если перетаскивание запрещено, то переместить элемент можно, открыв его на редактирование и выбрав другую  родительскую запись. |



*Рис. 12. Окно просмотра мастер-дитэйл, вертикальная раскладка*

Общим для всех форм мастер-дитэйл является наличие двух связанных наборов данных (датасетов). Один из них является главным (мастером), а другой – подчиненным (детальным). Содержимое детального набора данных зависит от текущей записи в главном наборе. Например, в окне просмотра документа **Накладная на приход ТМЦ**, список накладных является главным датасетом, а позиции выбранной накладной – детальным. Для работы с детальным датасетом в меню окна предназначен отдельный пункт, который называется **Детальный** или **Позиция** (см. Рис. 10 или Рис. 11). Кроме этого, детальный датасет имеет свою панель управления и свое контекстное меню, которое вызывается по нажатию на правую кнопку мыши. Поскольку команды **Добавить**, **Изменить**, **Удалить** и т.д. ведут себя аналогично командам для главного датасета мы не будем

останавливаться здесь на их подробном описании.

##### Отображение только записей детального датасета

Посмотрим на форму справочника товаров (Рис. 11). Слева отображается древовидный список групп, а справа – содержимое выбранной группы. Но,

что, если мы захотим увидеть все товары, присутствующие в базе данных? Гедымин дает нам такую удобную возможность. Для этого достаточно передвинуть разделяющую полосу влево так, чтобы дерево вообще исчезло с экрана. Выполнив обратную операцию, отыскав у левой границы окна разделяющую полосу и передвинув ее вправо мы вернем форму в исходное состояние.

## Форма просмотра документа

Форма просмотра объектов типа документ отличается тем, что пункты меню называются не **Главный** и **Детальный**, а **Документ** и **Позиция**, и на панели инструментов добавлены следующие команды:

**Для шапки документа**

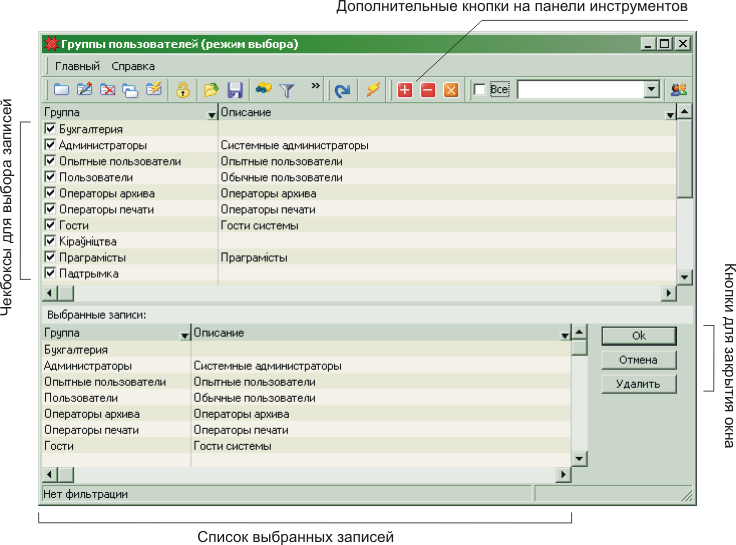
|  |  |  |
| --- | --- | --- |
|  | **Команда** | **Описание** |
|  | Провести проводки  по документу | Заново формирует проводки для текущей  и всех нижележащих записей. |
|  | Перейти на проводку | Открывает окно журнала хозяйственных операций и позиционирует курсор на  проводке для выбранного документа. |

**Для позиции документа**

|  |  |  |
| --- | --- | --- |
|  | **Команда** | **Описание** |
|  | Перейти на проводку | Открывает окно журнала хозяйственных операций и позиционирует курсор на проводке для выбранной позиции  документа. |
|  | Карточку ТМЦ | Команда доступна только в окне складского документа. Открывает на экране окно  карточки ТМЦ. |
|  | Показать проводки | Показывает в отдельной таблице внизу  окна проводки по текущему документу. |

## Форма в режиме выбора

Операция выбора нескольких записей достаточно часто встречается в повседневной работе. Например, оператор выбирает товары при заполнении счет-фактуры, системный администратор выбирает группы пользователей, при настройке политик безопасности и т.д. Для выбора нескольких записей используется форма просмотра, открытая в режиме выбора.



*Рис. 13. Форма в режиме выбора*

Обратимся к рисунку. Выбор (пометка) записей производится с помощью чекбоксов, которые выводятся в крайней левой колонке таблицы. Дополнительные кнопки на панели инструментов, которые появляются в режиме выбора, позволяют:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Команда** | **Быстрый**  **вызов** | **Описание** |
|  | Выделить все |  | Пометить все записи в таблице. |
|  | Убрать  выделение |  | Снять пометку с записей в таблице. |
|  | Убрать  выделение со всех |  | Снять пометку со всех выделенных записей. |

На первый взгляд может показаться, что команды **Убрать выделение** и **Убрать выделение со всех**. Выполняют одно и тоже действие. Но это не так. Рассмотрим, например, выбор товаров из окна **Справочник ТМЦ**. Пусть в дереве групп рисутствуют пункты **ТНП** и **Продтовары**. В группе **ТНП** два товара: Пылесос и Печь СВЧ, из которых один товар – Пылесос – выбран. В группе **Продтовары** три позиции: Мука, Сахар, Соль. Первые две выбраны. В конкретный времени в списке товаров отображается содержимое только одной группы, на которой установлен курсор. Пусть это будет товарная группа **Продтовары**. Команда **Убрать выделение** снимет выделение с позиций Мука и Сахар, так как они отображены в таблице. Пылесос останется помеченным, так как входит в другую группу. Если, же мы

выберем команду **Убрать выделение со всех**, то пометка снимется со всех трех товаров, не взирая на то, каким группам они принадлежат.

В нижней части окна находится панель на которой выводится список выбранных записей, а так же кнопки:

|  |  |
| --- | --- |
| Ок | Подтверждает выбор записей и закрывает окно. |
| Отмена | Отменяет выбор записей и закрывает окно. |
| Удалить | Удаляет запись из списка отобранных, т.е. снимает с нее  выделение. |

Форма в режиме выбора открывается всегда в модальном режиме. Т.е. для продолжения работы с программой ее обязательно необходимо закрыть с помощью кнопок «Ок» или «Отмена». Закрытие формы с помощью пиктограммки в виде крестика в правом верхнем углу равносильно нажатию кнопки «Отмена».

## Поиск данных

В базе данных могут находиться миллионы записей. Часто среди них надо отыскать одну-две, удовлетворяющие определенным условиям. Гедымин предоставляет пользователю пять способов поиска. Рассмотрим их более подробно, а также постараемся дать ответ на вопрос: Когда какой поиск лучше использовать?

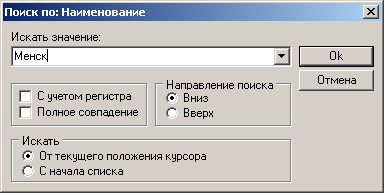
#### Поиск по первым введенным символам

Данный вид поиска доступен в таблице и дереве. Если вы хотите найти определенное значение, установите курсор в нужную колонку и начинайте набирать первые символы значения. Если, такая запись есть в базе данных,

— курсор перейдет на нее. Если несколько записей удовлетворяют введенным начальным символам — курсор перейдет на первую из них.

#### Поиск данных в колонке

Данный вид поиска доступен в таблице и дереве и позволяет найти запись по вхождению строки. Используйте сочетание клавиш **Ctrl+F** или конктекстное меню для его активации. В появившемся диалоговом окне (Рис. 14) следует ввести искомое значение, установить параметры и нажать кнопку **Ок**.



*Рис. 14.Диалоговое окно поиска*

Список параметров поиска приведен в таблице ниже:

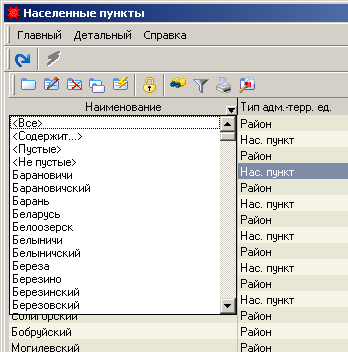
|  |  |
| --- | --- |
| С учетом регистра | Поиск будет чуствителен к регистру символов. |
| Полное совпадение | Значение поля записи должно полностью  совпадать с введенной строкой. |
| Вниз | Поиск будет идти вниз от текущего положения  курсора. |
| Вверх | Поиск будет идти вверх от текущего положения  курсора. |
| От текущего  положения курсора | Поиск начнется от текущего положения курсора. |
| С начала списка | Поиск начнется с первой записи в списке. |

Если запись найдена — курсор будет установлен на нее. В противном случае, на экране появится сообщение о том, что поиск окончился безуспешно. Для поиска следующей подходящей записи достаточно нажать клавишу **F3** или выбрать команду **Найти следующее** из контекстного меню. Заголовок окна содержит информацию о колонках, по которым идет поиск. Если в таблице установлено расширенное отображение, то поиск

будет осуществляться сразу по нескольким столбцам.

#### Автофильтр

Кто работал с электронными таблицами Microsoft Excel наверняка сразу поймет о чем идет речь. Автофильтр позволяет быстро отобрать данные на основании значений, уже имеющихся в колонке. Для установки автофильтра необходимо щелкнуть мышью по треугольничку в правом нижнем углу заголовка колонки.



*Рис. 15. Автофильтр*

На экране появится список значений из колонки. Выберите конкретное значение и на экране отобразятся только соответствующие ему записи.

Если автофильтр установлен, то текст в заголовках колонок таблицы выводится синим цветом. Кроме этого, синим цветом подсвечивается треугольничек в той колонке, по которой установлена автофильтрация.

Вверху списка располагаются четыре команды:

|  |  |
| --- | --- |
| <Все> | Снимает автофильтр и выводит все записи в таблице. |
| <Содержит> | Позволяет отобрать только те записи, которые  содержат введенную подстроку. |
| <Пустые> | Позволяет отобрать записи, которые не содержат  значений в данной колонке. |
| <Не пустые> | Позволяет отобрать записи, которые содержат какое  либо, не пустое, значение в данной колонке. |
| <Не 0 и не пустые> | Команда доступна только для числовых полей. |

В целях экономии памяти автофильтр выводит в списке только первые несколько десятков уникальных значений из колонки.

##### Регулярные выражения

В диалоговом окне ввода строки фильтрации (команда <Содержит> из меню автофильтра) присутствует флаг **Использовать регулярные выражения** (см. Рис. 16).



*Рис. 16. Ввод строки фильтрации*

Если он снят, то записи отбираются по вхождению введенной подстроки, если установлен -- каждая запись рассматривается на предмет соответствия введенному регулярному выражению.

Что же представляют из себя регулярные выражения? Читатель наверняка знаком с возможностью поиска файлов в операционной системе по маске, которая может содержать специальные символы: \* обозначает любое количество символов в имени файла, ? – один символ в указанной позиции. Например, маске Отчет\*.doc соответствуют все документы Microsoft Word, название которых начинается со слова Отчет, а \*.?db – это и файлы баз данных сервера Interbase (расширение GDB), и Firebird (расширение FDB).

Регулярные выражения похожи на маски файловой системы, но предоставляют более широкие возможности. Регулярное выражение состоит из литералов, метасимволов, квантификаторов и множеств. С помощью круглых скобок выражение может быть разбито на несколько подвыражений. Вертикальная черта | обозначает оператор выбора ИЛИ.

Литерал – это любой символ, наличие или отсутствие которого проверяется в строке. Например, регулярное выражение "банк" соответствует всем строкам, содержащим в себе слово "банк".

Метасимволы ^ (начало строки) и $ (конец строки) позволяют задать положение искомой подстроки. Например, выражение "^ОАО" найдет все строки, которые **начинаются** со слова "ОАО", а "банк$" – которые **оканчиваются** на слово "банк".

Вместо литерала можно использовать множество допустимых символов, заключив его в квадратные скобки. Например, найти все строки, которые начинаются с цифры можно следующим образом: "^[0123456789]". Задать множество мы можем следующими способами (в скобках приведены примеры):



Перечислив все допустимые символы ([0123456789]); Указав диапазон допустимых символов ([0-9]);

Используя один из метасимволов: w – любая буква и знак подчеркивания; W – все, кроме букв и знака подчеркивания; d – цифра; D – все, кроме цифр.

Метасимвол ^ в начале множества обозначает отрицание. Например, выражение "^[^АБВ]" соответствует всем строкам, которые **не начинаются** с букв А, Б или В.

Метасимвол "." обозначает любой возможный символ.

Квантификаторы задают допустимое количество повторений литерала, символа из множества или подвыражения и бывают следующими:

\* -- любое количество, включая 0.

+ -- любое количество, большее нуля.

{n} -- ровно n повторений.

{n,m}-- ровно n или m повторений.

{n-m}—количество повторений в диапазоне от n до m, включая границы.

Круглые скобки выделяют подвыражения. Например, следующее регулярное выражение найдет все строки в которых **один и только один** раз встречается слово "банк" – "(банк){1}".

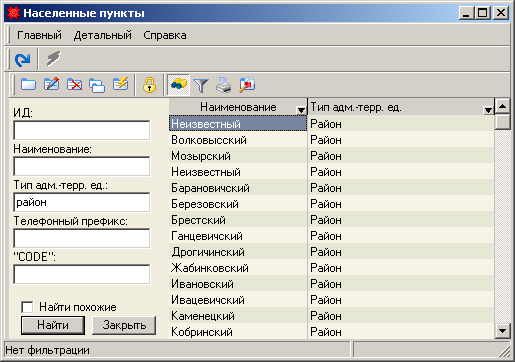
С помощью вертикальной черты – оператора ИЛИ – можно выбрать одно из двух выражений. Например, чтобы отобрать все строки, которые начинаются с аббревиатур "ОАО" или "ЗАО", следует воспользоваться таким выражением: "^(ОАО|ЗАО){1}". Эту же задачу можно решить немного другим способом: "^[ОЗ]{1}(АО){1}".

Символ обратной косой черты "\" перед метасимволом позволяет использовать его в качестве литерала. Например, все цены в долларах мы отыщем таким выражением: "\$[d]+".

#### Панель поиска

Панель поиска11 открывается с помощью кнопки **Поиск** на панели инструментов или с использованием команды **Поиск** в меню окна.

11 Данное название способно запутать пользователя. Под поиском обычно понимают установку курсора на нужную запись в общем списке, а под фильтрацией – отбор и отображение только записей, удовлетворяющих определенному критерию. Таким образом, Панель поиска правильнее было бы назвать Панелью фильтрации или отбора.



*Рис. 17. Панель поиска*

На ней отображается список полей и две кнопки: **Найти** — выполняет поиск данных согласно введенным критериям и **Закрыть** — закрывает панель поиска, возвращает датасет в исходное состояние.

В зависимости от типа поля поиск осуществляется:

Для строковых полей — по вхождению введенной подстроки.



Для числовых полей и полей типа дата и время — по точному соответствию введенного значения.

При вводе значений для поиска по строковым полям можно использовать шаблоны % и \_. Первый заменяет произвольное (вкдючая ноль) количество любых символов, а второй — один любой символ в данной позиции. Если необходимо найти символ процента или подчеркивания непосредственно, наберите их, предворив символом процента. Например, %% — найдет все записи, содержащие % в указанном поле, а %\_

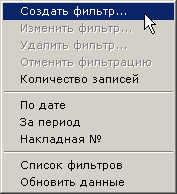
— все записи, содержащие символ подчеркивания.

#### Фильтрация данных

Из четырех описанных ранее способов поиска, только панель Поиска (и, при определенной настройке внешнего вида таблицы, поиск в колонке) позволяют искать записи по значениям нескольких полей. Но, и они могут использовать только условия равно или содержит. Если надо организовать сложный поиск данных мы прибегаем к механизму фильтрации.

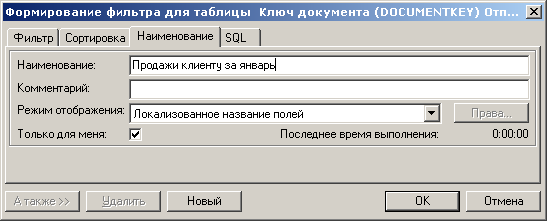
Пусть перед нами стоит задача найти все накладные на оптовую реализацию товара покупателю "Рога и Копыта" за январь 2007 года. Рассмотрим ее пошаговое решение:

1. На демонстрационной базе данных воспользуемся **Исследователем** и откроем окно **Торговля\03.Отпуск товара на сторону (оптовая торговля)**.
2. Найдем на панели инструментов в верхней части окна кнопку фильтрации  и выберем ее. На экране появится меню:



*Рис. 18. Меню команды Фильтрация*

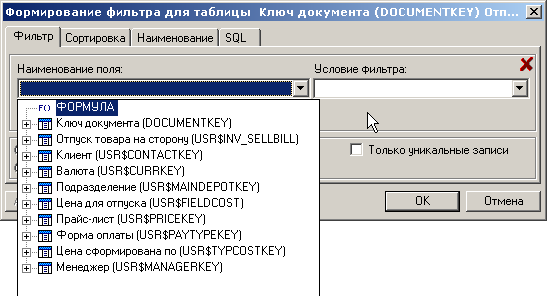
1. Из появившегося меню выберем команду **Создать фильтр…** На экране отобразится диалоговое окно создания фильтра:



*Рис. 19. Диалоговое окно создания фильтра. Вкладка Наименование*

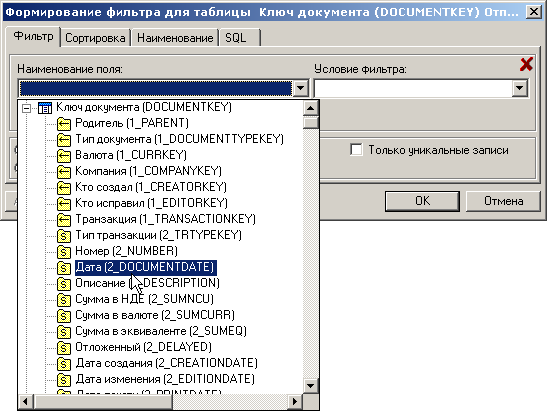
1. Введем в поля **Наименование** и **Режим отображения** значения, как показано на Рис. 19.
2. Перейдем на вкладку Фильтр и раскроем выпадающий список

###### Наименование поля:



*Рис. 20. Вкладка Фильтр*

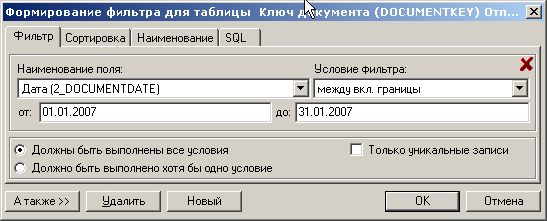
1. Как видно на Рис. 20, список содержит наименования всех таблиц, участвующих в запросе. Дата документа содержится в таблице GD\_DOCUMENT, названной в списке как **Ключ документа**12. Раскроем список полей этой таблицы. Для этого щелкнем мышью по изображению плюса, слева от наименования:



*Рис. 21. Выбор поля из таблицы документа*

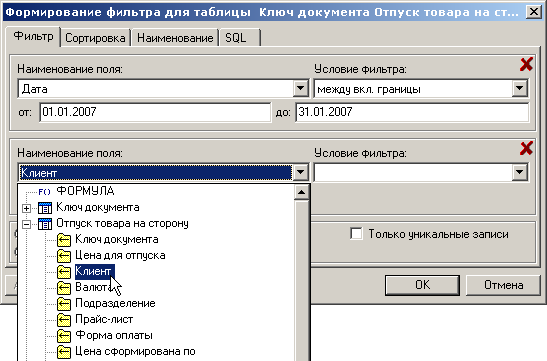
12 Так как одна и таже таблица может участвовать в запросе несколько раз, в качестве имени используется локализованное название поля-ссылки на эту таблицу. В данном случае таблица документа **Отпуск товара на сторону** (USR$INV\_SELLBILL) имеет поле DOCUMENTKEY, которое ссылает на таблицу GD\_DOCUMENT. **Ключ документа**, локализованное название этого поля, мы и видим в списке.

1. С помощью мыши выберем поле **Дата** (см. Рис. 21), из списка **Условие фильтра** выберем **Между вкл. границы**, и в поля **От** и **До** введем диапазон дат от первого до последнего дня января, как показано на Рис. 22 ниже.



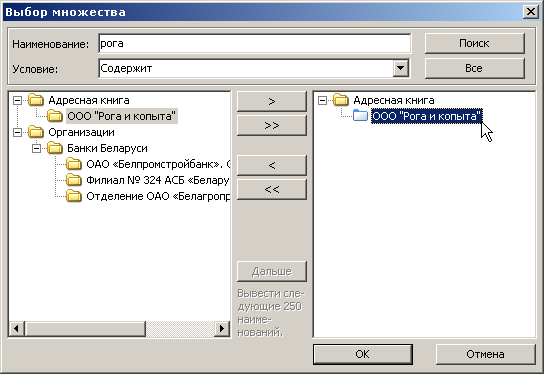
*Рис. 22. Ввод условия фильтрации*

1. Покончив с первым условием, приступим к вводу второго. Найдем и выберем в левом нижнем углу окна кнопку **А также >>**. В выпадающем списке **Наименование поля** найдем и раскроем содержимое таблицы **Отпуск товара на сторону**. Выберем поле **Клиент**, как показано на рисунке ниже:



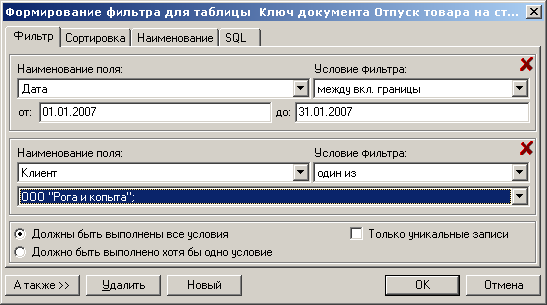
*Рис. 23. Создание второго условия*

1. Условие фильтра установим **Один из**. При этом в нижней части панели появится еще один выпадающий список для выбора значения. Попытаемся раскрыть его. На экране откроется окно поиска и выбора значения фильтрации:



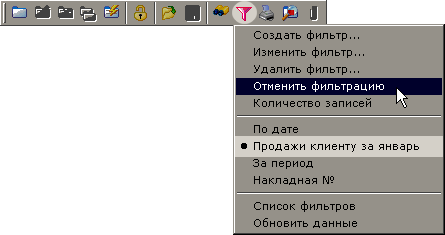
*Рис. 24. Окно поиска и выбора значения фильтрации*

1. Сначала, мы должны найти в базе данных нужные нам объекты. Так как нам известно наименование искомой компании, введем в поле **Наименование** строку "рога", а в поле **Условие** выберем "Содержит" и нажмем кнопку **Поиск**. В левом списке отобразятся объекты, в названии которых присутствует подстрока "рога". В данном случае нас интересует единственная запись -- ООО "Рога и копыта". Установим на нее курсор и с помощью кнопки  скопируем в правый список. Закроем окно с помощью кнопки **Ok**.
2. На этом формирование критерия фильтрации закончено. Окно фильтра примет следующий вид:



*Рис. 25. Второе условие фильтрации*

1. С помощью кнопки **Ок** в правом нижнем углу окна сохраним созданный фильтр и применим его к набору данных. Теперь в списке накладных отображаются только документы, выписанные в январе на ООО "Рога и копыта".
2. Обратите внимание, что пиктограммка фильтрации изменила свой цвет с серого на красный, а в меню появилось наименование нового фильтра:



*Рис. 26. Меню фильтрации после добавления нового фильтра*

1. Для возвращения к исходному набору данных следует воспользоваться командой **Отменить фильтрацию**.

На данном примере мы ознакомились с основной последовательностью действий при создании нового фильтра. Пусть вас не пугает количество действий, которые пришлось выполнить для отбора из общего списка накладных по конкретному клиенту за указанный период. Это вполне разумная плата за практически неограниченные возможности поиска и фильтрации информации, предоставляемые нам данным механизмом. К тому же, однажды созданный фильтр записывается в базу данных и его последующее применение требует не более усилий, чем выбор соответствующего пункта из меню.

Механизм фильтрации предназначен для опытного пользователя или разработчика. Как правило, на этапе создания прикладного решения, для каждого окна формируется свой список фильтров, которые могут быть полезны конечному пользователю.

##### Меню фильтрации

Пример меню фильтрации приведен на Рис. 18. Ниже дано описание всех его команд.

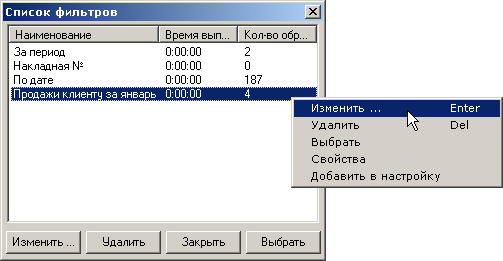
|  |  |
| --- | --- |
| **Команда** | **Описание** |
| Создать фильтр… | Вызывает на экран окно создания нового  фильтра. |
| Изменить фильтр… | Открывает окно для редактирования  активного фильтра. |
| Удалить фильтр | Удаляет активный фильтр. |
| Отменить фильтрацию | Отменяет действие активного фильтра и |

|  |  |
| --- | --- |
|  | возвращает набор данных в исходное  состояние. |
| Количество записей | Вычисляет количество записей в текущей  выборке13. |
| *<Список доступных фильтров>* | Список из не более чем десяти ранее созданных фильтров отображается в этой  части меню. |
| Список фильтров | Выводит на экран диалог со списком всех  фильтров для данного окна. |
| Обновить данные | Повторно выполняет запрос на сервере.  Аналогична действию клавиши **F5**. |

С помощью политик групповой безопасности можно настроить права пользователей на изменение и применение фильтров. По-умолчанию создавать, редактировать и удалять фильтры могут только члены групп **Администраторы** и **Опытные пользователи**.

##### Окно со списком фильтров

В окне отображается все фильтры, даже те, которые не помещаются в меню фильтрации.



*Рис. 27. Окно со списком фильтров*

Кнопки внизу окна и контекстное меню дают в совокупности шесть команд. Приводим их описание ниже:

13 При подсчете количества записей они не загружаются в оперативную память клиентского компьютера и, следовательно, не передаются по сети. Вместо этого создается новый запрос, где исходная SELECT часть заменяется на выражение SELECT COUNT(\*). Модифицированный запрос отправляется на сервер, который и подсчитывает количество записей. Команда достаточно интеллектуальна, чтобы не загружать лишний раз сервер, если все записи выборки уже загружены на клиентскую машину.

|  |  |
| --- | --- |
| **Команда** | **Описание** |
| Изменить… | Открывает диалоговое окно редактирования  выбранного фильтра. |
| Удалить | Удаляет выбранный фильтр. |
| Закрыть | Закрывает окно без изменения текущего фильтра. |
| Выбрать | Делает выбранный фильтр активным и закрывает  окно. |
| Свойства | Позволяет просмотреть свойства фильтра и настроить права доступа. С помощью прав доступа можно сделать фильтр невидимым для некоторых  групп пользователей. |
| Добавить в настройку | Позволяет добавить в настройку выбранный  фильтр. |

##### Окно создания или редактирования фильтра

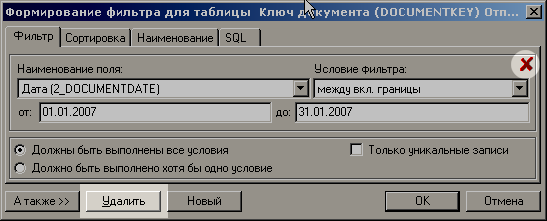
Диалоговое окно создания/изменения фильтра (см. Рис. 19) имеет четыре вкладки: **Фильтр**, **Сортировка**, **Наименование** и **SQL**. Как вы уже поняли, создание фильтра начинается с ввода его имени, которое должно быть уникальным. Тут же, на вкладке **Наименование**, можно указать необязательный комментарий. С помощью списка **Режим отображения** мы выбираем, какое наименование поля будет использоваться при определении критерия отбора: физическое имя колонки таблицы из базы данных, локализованное наименование платформы Гедымин или оба14. Если флажок **Только для меня включен**, то фильтр будет доступен только создавшему его пользователю, если выключен – имя фильтра появится в меню у всех пользователей системы.

На вкладке **Фильтр** (см. Рис. 20) мы указываем один или несколько критериев формирования выборки. Последовательность действий следующая:

1. Из списка **Наименование поля** выбрать поле.
2. Из списка **Условие фильтра** выбрать условие отбора записей.
3. В зависимости от типа поля и условия отбора ввести одно или несколько значений.
4. При необходимости, с помощью кнопки **А также>>** в нижней части окна, добавить следующий критерий отбора. Повторить шаги с 1 по 3.

14 В некоторых версиях платформы смена режима отображения начинает действовать только после перезагрузки Гедымина.

Критерий отбора можно удалить с помощью пиктограммки в правом верхнем углу панели (см. Рис. 28). Кнопка **Удалить** внизу окна удаляет последний критерий в списке.



*Рис. 28. Кнопки удаления критерия отбора*

Как видно на Рис. 21 поля в выпадающем списке обозначены разными пиктограммками. Попытаемся разобраться что значит каждая из них.

|  |  |
| --- | --- |
|  | Поле является ссылкой на запись в другой или этой же таблице. Справа от пиктограммки выводится имя поля. В зависимости от установленного Режима отображения это будет локализованное наименование поля, оригинальное или оба. В последнем случае наименование может выглядеть так: "Родитель (1\_PARENT)". Здесь, "Родитель" – локализованное имя поля, "PARENT" – оригинальное  название колонки, а префикс "1\_" указывает на тип поля – ссылка. |
|  | Простое поле, т.е. не ссылка и не атрибут типа множество. Справа от пиктограммки выводится имя поля. В зависимости от установленного Режима отображения это будет локализованное наименование поля, оригинальное или оба. В последнем случае наименование может выглядеть так: "Номер (2\_NUMBER)". Здесь, "Номер" – локализованное имя поля, "NUMBER" – оригинальное название колонки, а префикс "2\_" указывает что это просто поле с  данными. |
|  | Атрибут типа множество. Напомним, что для атрибута типа множество в базе данных создается отдельная таблица, которая  хранит связку, т.е. две ссылки, одну на таблицу, которой |

|  |  |
| --- | --- |
|  | принадлежит атрибут, и вторую – на таблицу-справочник с элементами множества15. После пиктограммки, в списке мы увидим строку следующего вида: "Получатель (4\_GD\_DOCUMENTLINK&DESTDOCKEY)" . Здесь, "Получатель" – это локализованное наименование поля ссылки в таблице-связке, "4\_" – префикс типа множество, "GD\_DOCUMENTLINK" – имя  таблицы-связки, "DESTDOCKEY" – имя поля в таблице-связке. |
|  | Ссылка из другой таблицы на данную таблицу. Справа от пиктограммки выводится строка следующего вида: "Согласно документу (5\_USR$REN\_DIVSTATEMLINE&DOCUMENTKEY)".  Здесь, "Согласно документу" – локализованное наименование поля, которое ссылается, "5\_" – признак ссылки, "USR$REN\_DIVSTATEMLINE" – имя таблицы, которая ссылается,  "DOCUMENTKEY" – имя поля, которое ссылается. |

#### Когда какой поиск использовать?

Разнообразие возможностей поиска может поставить в тупик даже опытного пользователя: какой способ следует предпочесть в той или иной ситуации?

Прежде чем попытаться сформулировать ответ, стоит пояснить какие механизмы использует каждый из описанных выше видов поиска. Поиск по первым введенным символам, поиск по колонке и автофильтр — отрабатывают на клиенте. Сначала весь набор данных перекачивается по сети и размещается в оперативной памяти рабочей станции, затем приложение пробегается от первой до последней записи сверяя каждую на соответствие введенному условию. Очевидно, что если датасет большой,

15 Рассмотрим товар (таблица GD\_DOOD), который имеет набор налогов. Справочник налогов располагается в таблице GD\_TAX. Связка, т.е. таблица множества, имеет имя GD\_GOODTAX. При фильтрации по товару в выпадающем списке будет одна позиция для атрибута множества: "Налог (4\_GD\_GOODTAX&TAXKEY)". Но, в качестве таблицы-справочника может выступать сама исходная таблица. Например, таблица множества GD\_GOODSET (Комплект товаров) имеет две ссылки на справочник ТМЦ. Первую – на комплект, и вторую – на комплектующий. В этом случае в выпадающем списке фильтрации для одного атрибута множества будет две позиции: "Товар (4\_GD\_GOODSET&GOODKEY)" и "Комплектующий (4\_GD\_GOODSET&SETKEY)".

содержит десятки колонок и десятки или сотни тысяч записей, то сканирование его приведет к:

1. загрузке сервера;
2. передаче большого объема данных по сети;
3. загрузке всех данных на клиентскую машину и, как следствие, потребление большого объема оперативной памяти для их размещения. Не говоря уже о том, что при попытке загрузить слишком большое количество данных оперативная память, доступная программе может исчерпаться и на экран будет выдано сообщение об ошибке;

Поиск с использованием фильтров или панели поиска приводит к добавлению условий отбора данных в SQL запрос и отсылке его на сервер. При этом минимизируется сетевой трафик и на клиентский компьютер пересылаются только те записи, которые отвечают заданным критериям отбора.

Очевидно, что выбор вида поиска в конкретной ситуации — это компромисс, между удобством и доступностью поиска по первым введенным символам или автофильтра и потреблением оперативной памяти и загрузкой сервера и сети. Если набор данных относительно не велик (на практике это означает не более нескольких тысяч записей) то можно смело прибегать к первым трем видам поиска. Если записей в датасете десятки тысяч и более, то следует использовать панель поиска или создать фильтр. Можно также порекомендовать использовать коомбинацию двух видов поиска. Например, на большой таблице настроить фильтр, который бы отбирал относительно небольшую часть записей и затем искать в них необходимые данные с помощью поиска по колонке, поиска по первым введенным символам или автофильтра.

|  |  |  |
| --- | --- | --- |
| Вид поиска | Преимущества | Недостатки |
| По первым символам |  | По мере поиска подходящей записи данные загружаются в оперативную память  клиента. |
| Поиск в таблице |  | По мере поиска подходящей записи данные загружаются в оперативную память  клиента. |
| Автофильтрация | Возможно использование  регулярных выражений. | Для активизации  автофильтра весь набор |

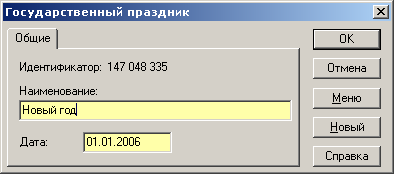
|  |  |  |
| --- | --- | --- |
|  |  | данных должен быть передан на клиента.  Невозможно сохранить параметры автофильтра между запусками  программы. |
| Панель поиска | Поиск выполняется на сервере. | Критерии поиска жестко заданы: вхождение подстроки для текстовых данных и точное соответствие для чисел и дат.  Значения поиска не сохраняются между запусками программы. |
| Фильтр | Поиск выполняется на  сервере. |  |

## Диалоговое окно

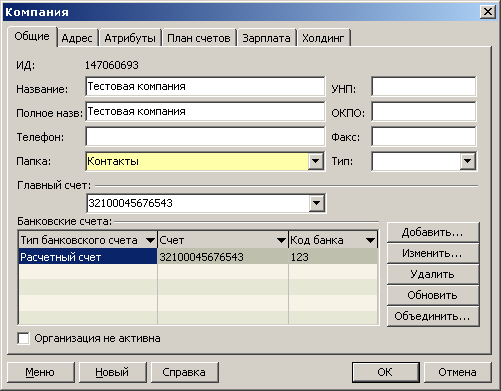
Формы просмотра различного вида, с которыми мы познакомились выше, используются для отображения списка объектов. Просмотр данных выбранного объекта и его редактирование осуществляется в отдельном модальном16 диалоговом окне.

В качестве примеров, ниже приведены диалоговые окна редактирования объектов Государственный праздник (Рис. 29) и Клиент (Рис. 30). Соответствущие справочники открываются из раздела Справочники Исследователя системы.

16 Модальным называется окно, которое пользователь должен закрыть для продолжения работы с программой. Как правило, модальное диалоговое окно имеет кнопки Ок и Отмена. Первая – сохраняет сделанные изменения и закрывает окно, а вторая – позволяет отказаться от сделанных правок.



*Рис. 29. Диалоговое окно редактирования государственного праздника*

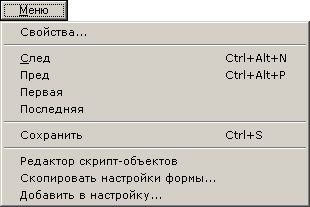


*Рис. 30. Диалоговое окно редактирования компании*

Легко понять, что поля для ввода данных располагаются на одной или нескольких вкладках в зависимости от сложности объекта. Неизменным для диалогового окна любого типа объектов является наличие пяти кнопок:

|  |  |
| --- | --- |
| Ок | Закрывает диалоговое окно и сохраняет внесенные  изменения. |
| Отмена | Закрывает диалоговое окно с отменой сделанных  изменений. |
| Меню | Открывает меню с дополнительными командами. |
| Новый | Сохраняет изменений в текущем объекте, создает новый объект и переходит к его редактированию  без закрытия окна. |
| Справка | Показывает справку по текущему объекту. |

При нажатии на кнопку Меню на экране откроется меню следующего содержания:



*Рис. 31. Меню с дополнительными командами в диалоговом окне*

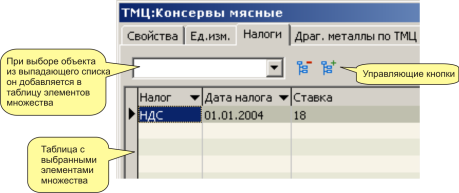
|  |  |
| --- | --- |
| Свойства… | Открывает диалоговое окно со свойствами  текущего объекта. Более подробно см. главу Бизнес- объект. |
| След | Сохраняет текущую запись и переходит на следующую в списке без закрытия диалогового  окна. |
| Пред | Сохраняет текущую запись и переходит на предыдущую в списке без закрытия диалогового  окна. |
| Первая | Сохраняет текущую запись и переходит на первую  в списке без закрытия диалогового окна. |
| Последняя | Сохраняет текущую запись и переходит на последнюю в списке без закрытия диалогового  окна. |
| Сохранить | Сохраняет внесенные изменения без закрытия  диалогового окна. |
| Редактор скрипт- объектов | Открывает редактор скрипт-объектови переходит на модуль текущей формы. Более подробно про редактор скрипт-объектов см. главу Среда  разработки. |
| Скопировать настройки формы… | Позволяет выбрать учетную запись пользователя системы и скопировать от нее визуальные  настройки формы. |
| Добавить в  настройку… | Добавляет форму и все связанные с ней скрипт-  объекты в выбранную настройку. |

Обязательные для заполнения поля ввода выделяются желтым цветом.

#### Вкладка множество

Редактирование атрибута типа множество рассмотрим на примере объекта Товар. Раскроем ветку Справочники в Исследователе системы и дважды щелкнем по позиции Справочник ТМЦ. Выберем из списка любой

товар и откроем его на редактирование. Перейдем на вкладку Налоги (см. Рис. 32).



*Рис. 32. Вкладка множество*

Для одного товара может быть одновременно задано несколько налогов. Например, НДС, налог с продаж, местный сбор и т.п. Это и есть множество. Как видно, для работы с одним атрибутом выделяется целая вкладка, основную площадь которой занимает таблица со списком отобранных записей. Данные можно исправлять непосредственно в таблице. Для удаления записи из таблицы, т.е. для исключения элемента из множества, необходимо выделить ее и нажать кнопку  в верхней части вкладки. Добавить запись, т.е. включить элемент во множество, можно либо выбрав его из выпадающего списка, либо с помощью кнопки . В последнем случае на экране появится форма в режиме выбора, где можно галочками отметить сразу несколько записей.

В реляционной базе данных множество хранится в промежуточной таблице, где каждая запись имеет две ссылки: на объект, которому принадлежит атрибут, и на справочник с возможными его элементами. Очевидно, что добавить запись в таблицу связки невозможно, пока не сохранена запись самого объекта. При переходе на вкладку атрибута типа множество, Гедымин попытается сохранить текущий объект и, если это невозможно (например, из-за того, что не заполнены обязательные поля), на экран будет выдано предупреждающее сообщение.

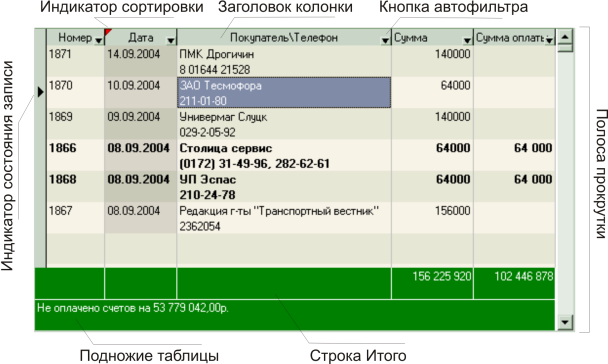
## Управляющие элементы

Для отображения и редактирования данных используются как стандартные управляющие элементы операционной системы Windows: поле ввода, список, древовидный список, выпадающий список, радио- кнопка, чек бокс и т.д., так и элементы управления, свойственные только платформе Гедымин. Их небольшое количество позволяет унифицировать интерфейс разнородных прикладных настроек и снизить затраты на обучение, а богатые функциональные возможности существенно облегчают

труд пользователя. Ниже мы познакомимся с таблицей, выпадающим списком, выпадающим списком для ввода множества, полем ввода даты и калькулятором для вычисления и ввода числовых значений.

#### Таблица

Бесспорно, центральный элемент пользовательского интерфейса любого прикладного решения на платформе Гедымин. Таблица17 служит для отображения списка записей, их сортировки, фильтрации и редактирования. Используется в формах просмотра и диалоговых окнах сложных документов. На Рис. 33 показаны основные элементы таблицы.



*Рис. 33. Составляющие элементы таблицы*

|  |  |
| --- | --- |
| Индикатор состояния записи | Опциональный компонент. Как правило, присутствует в момент редактирования данных непосредственно в таблице. Состояние записи отображается ввиде соответствующей пиктограммки:  Текущая запись. |

17 Не смотря на то, что таблица на экране в большинстве случаев отображает данные из нескольких таблиц базы, объединенных в SQL запросе, мы отдаем предпочтение именно этому термину перед такими наименованиями, как "грид" – калька от английского наименования компонента grid, и его переводом – словом "сетка". В тех случаях, когда возможно двусмысленное толкование сказанного, мы будем непосредственно пояснять: идет речь об экранном компоненте или объекте базы данных.



|  |  |
| --- | --- |
|  | Запись выделена.  Запись выделена и является текущей. Запись в состоянии редактирования. |
| Индикатор сортировки | Присутствие небольшого красного треугольника в левом верхнем углу заголовка сигнализирует о том, что записи в таблице упорядочены по убыванию значений в данной колонке. Красный треугольник в правом верхнем углу – сортировка по  возрастанию. |
| Заголовок колонки | Кроме отображения наименования поля (или полей), показываемого в колонке, выполняет следующие функции:  Перетаскивая заголовок колонки с помощью мыши можно изменить ее положение в таблице;  Перетаскивая разделительную линию между заголовками двух соседних колонок можно изменять их ширину;  Двойной щелчек по разделительной линии между заголовками колонок приведет к выравниванию размера колонки слева по находящимся в ней данным;  Одиночный щелчек по заголовку колонки упорядочит данные в таблице по возрастанию, повторный – по убыванию18;  Правая кнопка мыши на заголовке колонки вызывает контекстное меню. |
| Кнопка автофильтра | Открывает меню автофильтрации. См. раздел  **Поиск данных** выше по тексту. |
| Полоса прокрутки | Вертикальная и горизонтальная полосы прокрутки  используются для навигации по записям таблицы с помощью мыши. |
| Строка Итого | Для числовых колонок показывает суммарное значение или надпись **Рассчитать…**, если не все  записи загружены в буфер. |

18 Поля типа Дата первым щелчком сортируются по убыванию, а повторным – по возрастанию. Для отмены сортировки необходимо перечитать данные запроса, т.е. нажать клавишу F5 или выбрать команду Обновить из контекстного меню.

|  |  |
| --- | --- |
| Подножие таблицы | Показывает количество записей в таблице, переданных с сервера на клиентскую машину, количество выделенных записей и номер записи на которой находится курсор. Определив обработчик соответствующего события, разработчик может использовать подножие таблицы для вывода  вспомогательной информации. |

##### Навигация и выделение записей

Для навигации по таблице используются следующие клавиши:

←, Shift-Tab Перемещение на одну колонку влево.

→, Tab Перемещение на одну колонку вправо. Home, Ctrl - ← Перемещение на первую колонку в таблице.

End, Ctrl - → Перемещение на последнюю колонку в таблице.

↑ Перемещение на предыдущую запись.

↓ Перемещение на следующую запись.

PgUp Перемещение на один экран вверх.

PgDn Перемещение на один экран вниз.

Ctrl - ↑ Перемещение на первую запись в таблице. Ctrl - ↓ Перемещение на последнюю запись в таблице.

Ctrl-Tab Перемещение фокуса на следующий управляющий элемент окна в установленном порядке табуляции.

Для перемещения по записям таблицы с помощью мыши следует использовать полосы прокрутки, либо колесо мыши.

Выделение непрерывной группы записей осуществляется перемещением курсора при нажатой клавише Shift. Отдельные записи можно пометить, если удерживать клавишу Ctrl и щелкать по ним левой кнопкой мыши.

Если количество выделеных записей составляет две и более, то значения в строке Итого будут рассчитываться для группы выделенных записей, а не для всей таблицы.

##### Буфер записей

Видимые на экране записи отображаются из внутреннего буфера таблицы, который размещается в оперативной памяти. Начальное количество записей, загружаемых в буфер при открытии окна, зависит от параметров дисплея и настроек таблицы, и составляет, как правило, несколько десятков. По мере перемещения вниз за пределы экрана, новые записи подгружаются с сервера и размещаются в буфере.

Следующие действия приведут к считыванию **всех** записей с сервера и размещению их в оперативной памяти компьютера:



Перемещение на последнюю запись в таблице.

Нажатие на кнопку автофильтрации в заголовке колонки. Сортировка данных по колонке (щелчек левой кнопкой мыши по заголовку колонки).

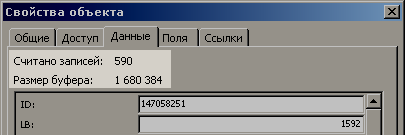
 Щелчек левой кнопкой мыши по надписи **Рассчитать…** в строке Итого.

 Поиск по первым введенным символам или команде **Поиск…**

несуществующего значения.

Следует быть осторожным при работе с большими наборами данных такими, как журнал хозяйственных операций, список документов или справочник клиентов. Считывание всех их записей может привести к израсходованию оперативной памяти, свопингу на жесткий диск и замедлению работы компьютера. В таких случаях рекомендуется сначала настроить фильтр для отображения небольшого среза данных (например, проводок за ограниченный период времени или клиентов из одного населенного пункта) и только потом работать с ними в таблице.

Количество считанных записей и размер буфера19 в байтах можно узнать, если из контекстного меню таблицы вызвать команду **Свойства…** и перейти на вкладку **Данные** (см. Рис. 34).



*Рис. 34. Размер буфера в окне Свойства объекта*

Для освобождения выделенной под буфер памяти достаточно выполнить команду **Обновить** (быстрая клавиша **F5**) или закрыть форму, т.е. щелкнуть левой кнопкой мыши по крестику в правом верхнем углу окна, одновременно удерживая нажатой клавишу **Shift**.

##### Редактирование в таблице

Платформа предоставляет нам две возможности редактирования записи: в отдельном диалоговом окне и непосредственно в таблице. По умолчанию,

19 Память под буфер выделяется блоками, т.е. размер занимаемый одной записью в общем случае **не равен** размеру буфера поделенному на количество считанных записей.

таблицы в формах просмотра находятся в режиме только для чтения. Нажатие клавиши **Enter** открывает на экране диалоговое окно с текущей записью, а попытка набирать текст с клавиатуры активизирует поиск по первым символам.

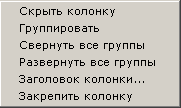
Переход в режим редактирования в таблице осуществляется с помощью нажатия кнопки  на панели инструментов. При этом, в позиции самой левой колонки, на экране появится индикатор состояния записи (см. Рис. 33). Пиктограммка  указывает на то, что таблица находится в **режиме просмотра**. Нажатие клавиши **Enter** переведет текущее поле в состояние редактирования. Того же эффекта можно достичь, если просто начать набирать текст на клавиатуре. Разница лишь в том, что во втором случае введенные символы заменят существующее содержимое поля. В зависимости от типа данных будет задействован соответствующий редактор колонки – строка ввода, выпадающий список и т.п. Для перехода на следующее поле следует закрыть редактор нажатием клавишь **Enter** или **Esc**. В последнем случае, все внесенные изменения будут отменены и поле примет свое прежнее значение. Индикатор  сигнализирует о том, что таблица находится в **состоянии редактирования**. Установив значения всех полей мы должны сохранить изменения в базе данных. Для этого достаточно переместить курсор на другую запись, например, с помощью клавиши **↑**. Даже если курсор установлен на первой позиции в списке или в нем присутствует всего одна запись, нажатие стрелки вверх **↑** сохранит сделанные изменения и переведет таблицу в состояние просмотра. Для отмены всех сделанных изменений достаточно нажать клавишу **Esc**.

В большинстве случаев, перемещение фокуса на другой элемент управления или закрытие окна с таблицей в состоянии редактирования приведет к автоматической отмене сделанных изменений.

Так как клавиша **Enter** в режиме редактирования в таблице открывает редактор для текущего поля, диалоговое окно вызывается коомбинацией клавишь **Ctrl-Enter**. Если таблица находится в режиме редактирования, вызов диалогового окна сохранит все сделанные изменения.

##### Контекстное меню

В зависимости от того, где нажать правую кнопку мыши: на заголовке таблицы или в области данных, контекстное меню будет содержать разный набор команд.

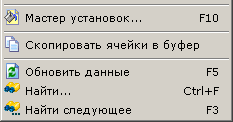


*Рис. 35. Контекстное меню заголовка колонки*

Как показано на Рис. 35 контекстное меню заголовка таблицы содержит следующие команды:

|  |  |
| --- | --- |
| Скрыть колонку | Делает невидимой текущую колонку. Вернуть  видимость колонки можно через Мастер установок. |
| Заголовок колонки… | Изменить наименование колонки. |
| Закрепить колонку | Сделать текущую колонку и все колонки слева от нее несдвигаемыми при прокрутке вправо за пределы экрана. Закрепленая колонка (или колонки) визуально отделяется от остальной  таблицы жирной вертикальной чертой. |

Контекстное меню над данными таблицы дублирует основные команды из меню формы просмотра и панели инструментов, а также содержит следующие пункты:



*Рис. 36. Контекстное меню таблицы*

|  |  |
| --- | --- |
| Мастер установок | Открывает окно с параметрами текущей таблицы. |
| Скопировать ячейки в буфер | Помещает значения текущих записей в буфер обмена в формате CSV. Команда широко используется для экспорта данных таблицы в  Microsoft Excel. |
| Обновить данные | Перечитывает (переоткрывает) набор данных,  связанный с таблицей на экране. Освобождает память, выделенную под буфер записей. |
| Найти… | Открывает окно поиска в таблице. |
| Найти следующее | Повторяет поиск для последнего введенного  значения. |

##### Мастер установок

Название Мастер установок является анахронизмом. Возможно, когда-то процесс настройки внешнего вида таблицы и требовал пошагового выполнения действий, но сейчас – это просто диалоговое окно с пятью вкладками. Ниже мы рассмотрим предназначение каждой из них.



*Рис. 37. Мастер установок. Вкладка Таблица*

Представленная на Рис. 37 вкладка Таблица содержит следующие разделы:

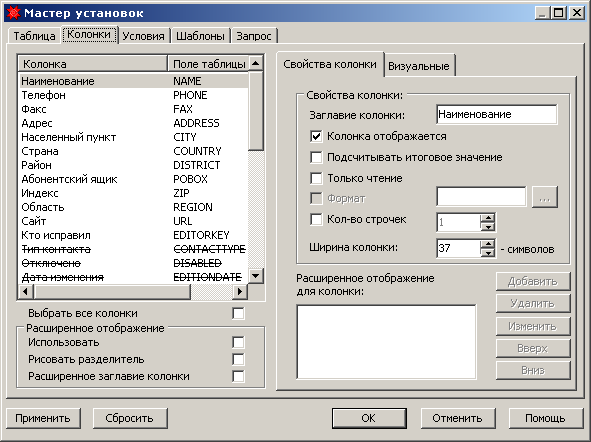
|  |  |
| --- | --- |
| Шрифт и цвет | Шесть кнопок для настройки шрифта и цвета фоновой заливки для записи таблицы, выделенной  записи и заголовка соответственно. |
| Режим отображения полос | Режим, при котором четные и нечетные строки таблицы выводятся с разными цветами фона. Флаг **Использовать** позволяет включать/выключать режим, а две одноименные кнопки **Цвет** служат для выбора фона нечетной и четной полосы  соответственно. |
| Внешний вид таблицы | Пример того, как будет выглядеть таблица с  примененными параметрами визуального форматирования. |

Кроме этого, на вкладке располагаются четыре флажка:

|  |  |
| --- | --- |
| Горизонтальные линии таблицы | Рисовать разделительные линии между строками  таблицы. Как правило, применяется при отключенном режиме отображения полос. |
| Растягивать колонки  по ширине таблицы | Все колонки будут выведены в пределах экрана без  необходимости прокрутки вправо. |
| Показывать строку | Отображать строку с итоговыми значениями в |

|  |  |
| --- | --- |
| Итого | нижней части таблицы. |
| Показывать  подножие таблицы | Отображать строку с вспомогательной  информацией в нижней части таблицы. |

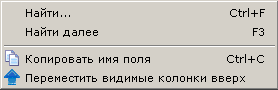
Если вкладка Таблица служит для настройки внешнего вида таблицы в целом, то на вкладке Колонки (см. Рис. 38) мы имеем возможность управлять параметрами каждого поля в отдельности.



*Рис. 38. Мастер установок. Вкладка Колонки*

В левой части вкладки находится список полей из отображаемого набора данных. Список имеет два столбца: локализованное наименование поля (его мы видим в заголовке колонки) и физическое наименование поля в SQL запросе. Наименования скрытых полей перечеркнуты. Перетаскивая строки списка с помощью мыши можно изменять порядок следования колонок в таблице.

Сложные наборы данных могут иметь по несколько сотен полей, среди которых не всегда легко отыскать нужное. По правой кнопке мыши над списком колонок вызывается контекстное меню.



*Рис. 39. Контекстное меню списка колонок*

Как показано на Рис. 39, меню содержит следующие команды:

|  |  |
| --- | --- |
| Найти… | Найти колонку по ее локализованному или  физическому наименованию. |
| Найти далее | Повторить поиск для последнего введенного  значения. |
| Копировать имя поля | Скопировать физическое имя текущего поля в |

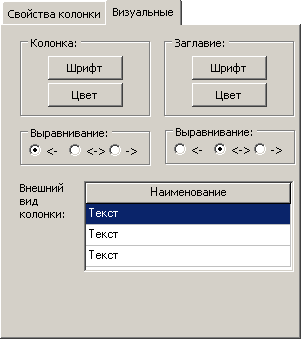
|  |  |
| --- | --- |
|  | буфер. |
| Переместить видимые колонки вверх | Переместить видимые колонки в верхнюю часть списка. Данная команда выполняется автоматически при завершении работы с мастером  установок. |

Настройка внешнего вида колонки осуществляется следующим образом: сначала выбираем одну или несколько колонок из списка, затем настраиваем их параметры в правой части окна, на вкладках **Свойства колонки** (см. Рис. 38) и **Визуальные**. С помощью флага **Выбрать все колонки** можно быстро выделить все позиции в списке. Ниже дано подробное описание параметров отображения колонки.

|  |  |
| --- | --- |
| Заглавие колонки | Наименование поля, которое выводится в заголовке колонки. При первом отображении таблицы и после сброса всех настроек в качестве заголовка колонки используется краткое локализованное  наименование поля в базе данных20. |
| Колонка  отображается | Данный флаг управляет видимостью колонки. |
| Подсчитывать итоговое значение | Имеет смысл для числовых полей. Если установлен,  сумма по колонке будет рассчитана и выведена в строке Итого. |
| Только чтение | Если флаг установлен, то при редактировании в таблице нельзя будет изменить значение поля. Следует понимать, что действие данного флага распространяется только на колонку таблицы на экране, а не на связанное с этой колонкой поле базы данных. Т.е. даже при установленном флаге,  поле можно будет изменить в диалоговом окне. |
| Формат | С помощью данного флага можно управлять форматированием значений типа число или дата при выводе на экран. Формат задается с помощью маски, которую необходимо ввести в поле или  выбрать из списка. Подробнее о масках см. ниже. |
| Кол-во строчек | Определяет высоту клетки таблицы в строках. |

20 Более подробно о кратком и полном локализованных наименованиях поля см. главу Структура базы данных.

|  |  |
| --- | --- |
| Ширина колонки | Ширина колонки в символах. Практически не имеет смысла при установленом режиме растягивания колонок и в таблицах, где пользователь может растягивать или сжимать  колонки с помощью мыши. |



*Рис. 40. Настройка цвета, шрифта и выравнивания колонки*

На вкладке Визуальные (см. Рис. 40) определяется шрифт, фоновый цвет и способ выравнивания данных и заголовка колонки.

##### Форматирование числовых значений

Маска, задающая формат вывода числовых значений может содержать следующие символы:

0 Если форматируемое значение имеет цифру в той позиции, в которой в строке маски находится «0», то эта цифра копируется в результирующую строку. В противном случае символ «0» будет помещен в результирующую строку.

# Если форматируемое значение имеет цифру в той позиции, в которой маска содержит «#», то эта цифра копируется в результирующую строку. В противном случае результирующая строка не меняется.

. Десятичная точка. Первый символ «.» в строке маски определяет положение десятичной точки в результирующей строке. Все остальные символы «.», если они есть, игнорируются. То, какой именно символ будет использован в результирующей строке в качестве десятичного разделителя, определяется настройками операционной системы просмотреть и/или изменить которые можно в Панели управления Windows.

, Разделитель тысяч. Если маска содержит один или несколько символов «,», то результирующая строка будет содержать разделители тысяч между каждыми тремя цифрами слева от десятичной точки. Конкретное расположение символов «,» в маске и их количество не влияет на положение разделителей тысяч в результирующей строке. Важен сам факт их наличия. То, какой именно символ будет использован в результирующей строке в качестве разделителя тысяч, определяется настройками операционной системы просмотреть и/или изменить которые можно в Панели управления Windows.

Е+ Инженерная нотация. Если любая последовательность из «Е+»,

«е+», «Е-», «е-» содержится в маске то чсило будет отформатировано согласно инженерной нотации. Группа в количестве до 4 символов «0» следующая сразе за последовательностью символов «Е+», «е+», «Е-» или «е-» определяет минимальное количество цифр в экспоненте. Последовательности «Е+», «е+» указывают на то, что знак «+» будет выводиться для положительных экспонент и знак «-» — для отрицательных. Последовательности «Е-», «е-» указывают на то, что знак экспоненты будет указываться только для отрицательных значений.

«хх»/’xx’ Символы, заключенные в двойные или одинарные кавычки, копируются напрямую в результирующую строку из маски и не влияют на форматирование.

; Точка с запятой разделяет секции, содержащие строки форматирования для положительных, отрицательных и нулевых значений.

Положение самого левого символа «0» слева от десятичного знака и самого правого символа «0» справа от десятичного знака определяет количество цифр, которые всегда присутствуют в результирующей строке.

Форматируемое число всегда округляется до того количество десятичных знаков, которое задано с помощью символов «#» или «0» справа от десятичной точки. Если маска не содержит десятичной точки, то форматируемое число округляется до ближайшего целого числа.

Строка формата может содержать от одной до трех секций, разделенных символом «;», задающих формат положительных, отрицательных и нулевых значений. Если секция отрицательных чисел или секция нулевых значений пропущены (т.е. содержат пустую строку), то используется строка форматирования, заданная для положительных значений.

Ниже приводятся примеры форматирования числовых значений:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Строка**  **форматирования** | **Значение** | | | |
| **1234** | **-1234** | **0.5** | **0** |
|  | 1234 | -1234 | 0.5 | 0 |
| **0** | 1234 | -1234 | 1 | 0 |
| **0.00** | 1234.00 | -1234.00 | 0.50 | 0.00 |
| **#.##** | 1234 | -1234 | .5 |  |
| **#,##0.00** | 1,234.00 | -1,234.00 | 0.50 | 0.00 |
| **#,##0.00;(#,##0.00)** | 1,234.00 | (1,234.00) | 0.50 | 0.00 |
| **#,##0.00;;Zero** | 1,234.00 | -1,234.00 | 0.50 | Zero |
| **0.000E+00** | 1.234E+03 | -1.234E+03 | 5.000E-01 | 0.000E+00 |
| **#.###E-0** | 1.234E3 | -1.234E3 | 5E-1 | 0E0 |

##### Форматирование значений типа дата и время

Маска, задающая форматирования значений типа дата и время может содержать следующие латинские символы:

1. Отображает дату в соответствии с коротким форматом даты, установленным в операционной системе, и время в соответствии с длинным форматом времени.
2. Отображает день ввиде числа без начального нуля (1-31). dd Отображает день ввиде числа с начальным нулем (01-31). ddd Отображает день ввиде аббривиатуры (Пн-Вс).

dddd Отображает полное наименование дня недели.

ddddd Отображает дату в соответствии с коротким форматом даты, установленным в операционной системе.

dddddd Отображает дату в соответствии с длинным форматом даты, установленным в операционной системе.

1. Отображает месяц ввиде числа без начального нуля (1-12). mm Отображает месяц ввиде числа с начальным нулем (01-12). mmm Отображает месяц ввиде аббривиатуры (Янв-Дек). mmmm Отображает наименование месяца (Январь-Декабрь).

yy Отображает год, как двухзначное число (00-99).

yyyy Отображает год, как четырех значное число (0000-9999). h Отображает часы как число без начального нуля (0-23). hh Отображает часы как число с начальным нулем (00-23).

1. Отображает минуты как число без начального нуля (0-59). nn Отображает минуты как число с начальным нулем (00-59). s Отображает секунды как число без начального нуля (0-59). ss Отображает секунды как число с начальным нулем (00-59).

z Отображает тысячные доли секунды как число без начальных нулей (0-999).

zzz Отображает тысячные доли секунды как число с начальными нулями (000-999).

t Отображает время согласно короткому формату отображения времени, установленному в операционной системе.

tt Отображает время согласно длинному формату отображения времени, установленному в операционной системе.

/ Отображает символ-разделитель частей даты установленный в операционной системе.

: Отображает символ-разделитель часов, минут, секунд, установленный в операционной системе.

«хх»/’xx’символы, заключенные в двойные или одинарные кавычки, копируются напрямую в результирующую строку из маски и не влияют на форматирование.

Символы, задающие формат могут быть заданы как в верхнем, так и внижнем регистре.

Если задана пустая строка в качестве маски, то форматирование производится как будто был задан символ «с».

##### Расширенное отображение данных в таблице

Сравните таблицы на Рис. 8 и Рис. 33. В чем разница? В первом случае каждая запись занимает ровно одну строку и в каждой клетке выводится значение только одного поля. Во втором – два разных поля (наименование клиента и его телефон) выводятся в одной ячейке и каждая запись занимает более одной строки по высоте. Такой режим называется расширенным отображением данных.

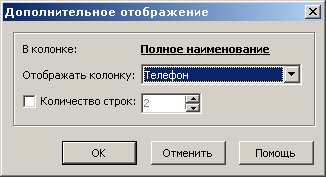
Обратите внимание на левый нижний угол вкладки **Колонки** (см. Рис.

38). Три флага управляют расширенным отображением:

|  |  |
| --- | --- |
| Использовать | Включает/выключает режим расширенного  отображения данных. |
| Рисовать разделитель | Определяет необходимость рисования  горизонтальной разделительной черты между полями в пределах одной клетки. |
| Расширенное  заглавие колонки | Определяет высоту заглавия колонки. Одна строка,  если флаг снят. Две и более – если установлен. |

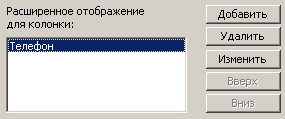
Рассмотрим пошаговую процедуру настройки расширенного отображения на примере справочника клиентов. Добавим в колонку Наименование сведения о телефонном номере клиента:

1. Откроем окно справочника клиентов (Исследователь-Справочники- Клиенты).
2. Откроем Мастер установок (F10 или команда из контекстного меню).
3. Перейдем на вкладку Колонки.
4. Установим флаг **Использовать** расширенное отображение.
5. В списке колонок найдем позицию Наименование-NAME и установим на нее курсор.
6. В правом нижнем углу вкладки **Свойства колонки** найдем список полей расширенного отображения и выберем кнопку Добавить. На экране появится окно, как показано на Рис. 41 ниже. Выберем из выпадающего списка поле **Телефон** и нажмем **Ок**.



*Рис. 41. Добавление поля в расширенное отображение*

1. Список полей расширенного отображения примет следующий вид:

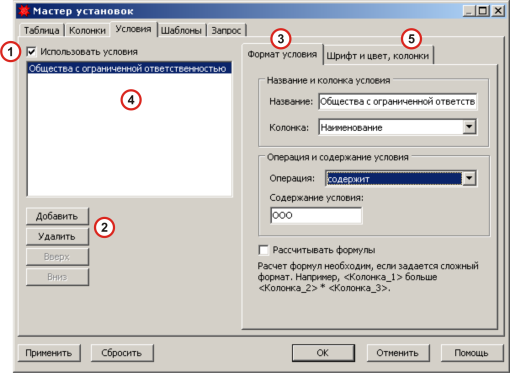


*Рис. 42. Список полей расширенного отображения*

1. С помощью кнопок справа от списка (см. Рис. 42) можно добавлять, изменять или удалять поля из расширенного отображения, а также управлять порядком их следования.
2. Закроем Мастер установок с помощью кнопки Ок.

##### Условное форматирование

Условное форматирование позволяет устанавливать внешний вид записи в зависимости от находящихся в ней данных. Например, неоплаченные документы в списке счетов-фактур можно выделить жирным шрифтом, а сторнировочные проводки в журнале хозяйственных операций вывести красным цветом. Настройка параметров условного форматирования осуществляется на вкладке Условия, как показано на Рис. 43 ниже.



*Рис. 43. Мастер установок. Вкладка Условия*

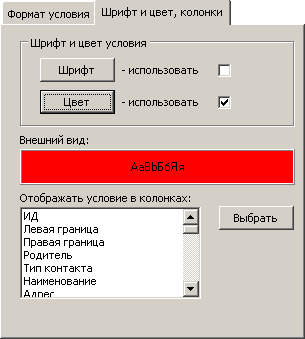
Поставим перед собой задачу вывести в справочнике клиентов наименования компаний, которые являются обществами с ограниченной ответственностью, на оранжевом фоне. Пошагово рассмотрим алгоритм создания нового условия:

1. Откроем окно справочника клиентов (Исследователь-Справочники- Клиенты).
2. Откроем Мастер установок (F10 или команда из контекстного меню).
3. Перейдем на вкладку Условия.
4. Установим флаг  -- **Использовать условия** (Здесь и далее цифрами в кружочке обозначены управляющие элементы вкладки Условия на Рис. 43).
5. Ниже списка условий находится группа кнопок . Найдем и выберем в ней кнопку **Добавить**.
6. Перейдем на вкладку  Формат условия и заполним поля следующим образом:

Название Общества с ограниченной ответственностью Колонка Наименование

Операция Содержит Содержание условия ООО

1. Обратите внимание, что наименование условия появилось в списке .
2. Перейдем на вкладку  -- Шрифт и цвет, колонки (см. Рис. 44).

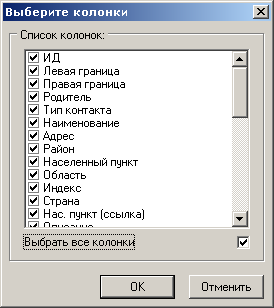


*Рис. 44. Визуальные параметры условного форматирования*

1. В группе **Шрифт и цвет условия** установим флаг напротив кнопки

###### Цвет.

1. С помощью кнопки **Цвет** выберем оранжевый фон для отображения записи в таблице.
2. Остается указать список колонок, к которым будет применено условное форматирование. Воспользуемся кнопкой **Выбрать**, справа от списка **Отображать условия в колонках**, и установим флаг **Выбрать все колонки**, как показано на Рис. 45.



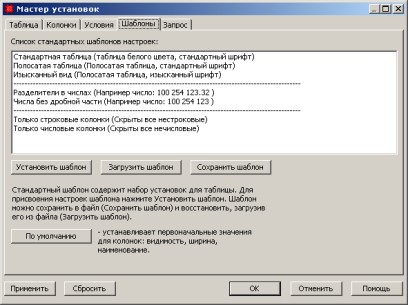
*Рис. 45. Выбор колонок для отображения условного форматирования*

1. Закроем окно выбора колонки с помощью кнопки Ок.
2. Закроем Мастер установок с помощью кнопки Ок.
3. Убедимся, что в справочнике клиентов все организации, наименование которых содержит аббревиатуру "ООО" выводятся на оранжевом фоне.

Если в списке задано несколько условий, то они применяются в порядке сверху вниз. Очередность их следования можно изменять с помощью кнопок  (см. Рис. 43).

##### Шаблоны внешнего вида

Вкладка Шаблоны (см. Рис. 46) предназначена для быстрой настройки внешнего вида таблицы на основе одного из шаблонов, а также для сохранения или загрузки настроек из дискового файла.

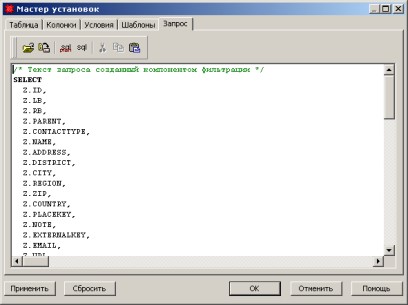


*Рис. 46. Мастер установок. Вкладка Шаблоны*

|  |  |
| --- | --- |
| Установить шаблон | Для применения одного из предопределенных шаблонов следует установить на нем курсор в списке, который занимает большую часть вкладки,  и нажать кнопку **Установить шаблон**. |
| Загрузить шаблон | Загружает визуальные настройки таблицы из  дискового файла. |
| Сохранить шаблон | Сохраняет текущие настройки таблицы в дисковом  файле. По умолчанию, файл с настройками имеет расширение DBG. |

##### Запрос

Разработчик достаточно часто прибегает к услугам данной вкладки, так как по тексту SQL запроса достаточно просто определить данные каких таблиц отображаются на экране.



*Рис. 47.Мастер Установок. Вкладка Запрос*

Как показано на Рис. 47 вкладка содержит панель с кнопками, предназначение которых описано в следующей таблице.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Команда** | **Быстрый**  **вызов** | **Описание** | |
|  | Загрузить из  файла |  | Заменяет текущий запрос  содержимым выбранного файла. | |
|  | Сохранить в  файл |  | Сохраняет текст текущего запроса в  файле. | |
|  | Применить с планом |  | Присваивает текущий запрос вместе с планом связанному компоненту (набору данных) и переоткрывает  его. | |
|  | Применить |  |  | Присваивает текущий запрос без плана связанному компоненту (набору данных) и переоткрывает  его. |
|  | Вырезать |  | Вырезает выделенный текст и  помещает его в буфер обмена. | |
|  | Копировать |  | Копирует выделенный текст в буфер  обмена. | |
|  | Вставить |  | Вставляет текст из буфера обмена. | |

Следует отметить, что возможность внести изменения в текст SQL запроса здесь предусмотрена исключительно для отладочных целей. Данные изменения нигде не сохраняются и будут утеряны после закрытия окна с таблицей.

##### Сброс настроек

Обратите внимание на две кнопки в левом нижнем углу Мастера Установок (см. Рис. 48).



*Рис. 48. Кнопки Применить и Сбросить*

Первая из них позволяет применить сделанные настройки без закрытия окна мастера, а вторая – возвращает внешний вид таблицы в первоначальное состояние. При этом, локализованные наименования колонок, их видимость, ширина, формат вывода данных и т.п. берутся из визуальных настроек поля, заданных при создании таблицы, о чем подробно будет рассказано ниже, в главе Структура базы данных.

##### Ограничение доступа к настройке таблицы

Настройка большой таблицы представляет собой достаточно трудоемкое занятие. На практике часто требуется защитить таблицу от неквалифицированного пользователя, не дать ему возможности изменять ее облик. Ограничить доступ пользователей можно через политики групповой безопасности. Соответствующая политика называется **Изменение визуальных настроек**.

##### Копирование настроек таблицы

Если с программой работают десятки пользователей, то нет никакого смысла настраивать таблицу для каждого из них заново. Достаточно настроить ее под одной учетной записью и затем распространить на другие учетные записи. В Гедымине существует четыре способа копирования визуальных настроек от одного пользователя другому:

###### Через сохранение настроек в файл

1. Загрузиться под логином пользователя, настройки которого будут скопированы;
2. Открыть окно, содержащее нужную таблицу;
3. Перейти в таблицу и открыть «Мастер установок»;
4. Перейти на закладку «Шаблоны»;
5. Нажать кнопку «Сохранить шаблон» и ввести имя файла;
6. Выйти из Гедымина и зайти под учетной записью пользователя которому необходимо скопировать настройки;
7. Открыть окно, содержащее нужную таблицу;
8. Перейти в таблицу и открыть «Мастер установок»;
9. Перейти на закладку «Шаблоны»;
10. Нажать кнопку «Загрузить шаблон» и выбрать ранее сохраненный файл.

###### Скопировать настройки формы от одного пользователя другому

В том числе, скопируются и настройки всех таблиц, находящихся на форме.

1. Загрузиться под учетной записью пользователя которому необходимо скопировать настройки;
2. Открыть нужную форму;
3. В пункте «Главный» меню формы выбрать команду «Загрузить настройки формы»;
4. Выбрать учетную запись, настройки которой будут скопированы;

###### Скопировать все визуальные настройки от одного пользователя другому

1. Открыть раздел «Сервис»→«Администратор» → «Пользователи» в

«Исследователе»;

1. Открыть учетную запись пользователя которому будем копировать настройки;
2. Выбрать кнопку «Профиль» в правой части диалогового окна;
3. Выбрать из выпадающего списка учтеную запись, все визуальные настройки которой будут скопированы данному пользователю;
4. Закрыть окно изменения учтеной записи.

###### Скопировать данные из хранилища

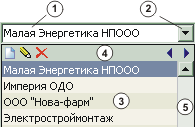
Скопировать данные непосредственно из хранилища одного пользователя в хранилище другого.

#### Выпадающий список

Выпадающий список предназначен для подстановки некоторого объекта в поле записи, путем выбора его из списка или посика в базе данных по наименованию.

Выпадающий список состоит из следующих частей (см. Рис. 49):

1. Поле ввода текста;
2. Копка вызова/скрытия списка;
3. Список объектов;
4. Панель инструментов;
5. Полоса прокрутки списка.



*Рис. 49. Выпадающий список*

##### Поле ввода текста

Поле ввода текста предназначено для ввода и отображения наименования объекта. При выборе объекта из базы данных путем поиска по наименованию в это поле необходимо ввести все наименование объекта или его часть. После этого необходимо нажать клавишу F3 или щелкнуть мышкой по кнопке вызова/скрытия списка. Будет осуществлен поиск объекта. Если в базе данных существует единственный объект, удовлетворяющий введенному наименованию, то он будет выбран и его наименование будет подставлено в поле ввода.

Если в базе данных существует несколько объектов, наименования которых удовлетворяют введенной строке, то на экране будет отображен список этих объектов. Если среди них находится искомый объект, то его необходимо выбрать с помощью мыши или используя клавиши управления курсором выделить его и нажать Enter.

Если список слишком велик — можно воспользоваться полосой прокрутки.

##### Кнопка вызова/скрытия списка

Если поле ввода пустое, то при нажатии на кнопку вызова списка на экран будет выведен список всех доступных объектов. Если в поле ввода введено наименование объекта или его часть, то при нажатии на кнопку вызова списка будет произведен поиск в базе данных и на экран будет выведен список найденных объектов. Если найден только один объект, то его наименование будет подставлено в поле ввода.

Если при открытом списке объектов нажать на кнопку вызова/скрытия списка, то он будет закрыт.

##### Список объектов

Список объектов выводится на экран когда пользователь нажимает на кнопку вызова списка или если был задан поиск объекта по наименованию

и для введенной строки в базе данных существует несколько подходящих объектов.

##### Полоса прокрутки списка

Если список содержит большее количество объектов чем помещается на экране, то можно воспользоваться полосой прокрутки для пролистывания списка вверх или вниз.

##### Панель инструментов

Находясь в списке объектов можно вызвать команды: Создания нового объекта,



Изменения выбранного объекта, Удаления выбранного объекта, Копирования выбранного объекта.

Соответствующие пиктограммки расположены на панели инструментов в верхней части выпадающего списка.

В правой части панели инструментов расположены стрелки изменения ширины выпадающего списка.

##### Горячие клавиши

Обычно, при работе с выпадающим списком (когда курсор находится в поле ввода) доступны горячие клавиши, приведенные ниже. Мы говорим обычно, потому что список доступных команд и, следовательно, активных горячих клавишь зависит как прав текущего пользователя, так и от конкретной настройки выпадающего списка, произведенной настройщиком.

F1 — вызов справки;



F2 — создание нового объекта; F3 — поиск;

F4 — изменение выбранного объекта; Ctrl-R — объединение двух записей; F7 — точный поиск;

F8 — удаление выбранного объекта; F9 — форма объекта;

F11 — свойства объекта;

F12 — переключение раскладки клавиатуры.

###### Вызов справки

Вызов справки по использованию выпадающего списка.

###### Создание нового объекта

При нажатии клавиши Ф2 на экран будет выведено диалоговое окно ввода нового объекта. После заполнения необходимых полей необходимо нажать Ок, после чего объект будет сохранен в базе, а его наименование подставлено в поле ввода.

В некоторых случаях, когда выпадающий список позволяет выбрать объекты разных типов, например: Компанию, Контакт, Банк и т.п. Перед окном ввода параметров объекта на экран будет выведен список доступных для создания типов объектов.

###### Поиск

При нажатии клавиши Ф3 будет выполнен поиск в базе данных объекта, наименование которого содержит введенную строку. Можно задать достаточно сложное условие поиска, если использовать специальные символы: "\_" и "%". Символ "\_" (подчеркивание) означает обязательное наличие любого символа в данной позиции. Символ "%" означает произвольную последовательность символов (включая пустую строку). Эти два символа аналогичны символам "?" и "\*", которые можно использовать при поиске файлов в операционной системе.

Например, введя в строке ввода: "технолог%1" мы найдем в базе данных и "ООО Технология, филиал №1" и "Технологическое управление №1".

###### Изменение выбранного объекта

Если выбран существующий в базе объект, то по нажатию на клавишу Ф4 можно отредактировать его в отдельном окне.

###### Текущий ключ

Просмотреть идентификатор выбранного объекта можно воспользовавшись клавишей Ф5.

###### Объединение двух записей

По коомбинации клавиш Ctrl-R можно вызвать окно объединения двух записей.

###### Точный поиск

По клавише Ф7 можно выполнить точный поиск. Точный поиск выполняется гораздо быстрее, особенно если по полю, по которому осуществляется поиск объекта в базе данных создан индекс. Однако, при этом необходимо набрать полное имя объекта. Если будет введена только часть наименования объект не будет найден.

###### Удаление выбранного объекта

Выбранный объект можно удалить из базы данных нажав клавишу Ф8.

###### Форма объекта

Если необходимо просмотреть список объектов или осуществить поиск по сложному критерию, то необходимо открыть Форму объекта по нажатию клавиши Ф9. Выбрать/найти там нужный объект и закрыть форму.

###### Свойства объекта

По нажатию клавиши Ф11 можно открыть стандартное окно с просмотром свойств выбранного объекта. В частности, из этого окна можно узнать уникальный идентификатор текущей записи.

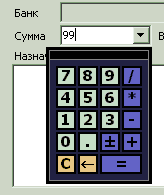
###### Переключение раскладки клавиатуры

Чрезвычайно удобная функция для тех, кто привык набирать текст не отрывая глаз от клавиатуры. Если вы набрали строку на белорусском языке, когда ваша клавиатура была переключена на английскую раскладку, вам не придется печатать все заново. Нажатие на F12 не только переключает раскладку, но и конвертирует набранный вами текст из одной раскладки в другую.

#### Калькулятор

Для ввода числовых данных применяется поле ввода с калькулятором.

Данный управляющий элемент состоит из следующих частей:



*Рис. 50. Управляющий элемент калькулятор*

|  |  |
| --- | --- |
| Поле ввода | Поле ввода позволяет набрать числовое значение. В простейшем случае его можно использовать как обычное поле ввода, т.е. ввести числовое значение и  по клавише табуляции перейти на следующее поле |

|  |  |
| --- | --- |
|  | в окне. Если необходимо выполнить вычисление, то следует набрать значение первого операнда. Затем нажать требуемую арифметическую операцию, после чего на экране появится панель калькулятора. Затем следует ввести значение второго операнда и нажать клавишу Ввод или Равно. Панель калькулятора исчезнет с экрана, результат арифметической операции будет  вычислен и подставлен в поле ввода. |
| Кнопка вызова калькулятора | Кнопка вызова калькулятора позволяет вывести на экран панель калькулятора с кнопками. После чего, можно с помощью мыши вводить числовые значения и выполнять над ними арифметические операции. Для того, чтобы закрыть панель  калькулятора необходимо нажать кнопку Равно. |
| Панель с кнопками | Панель с кнопками (панель калькулятора) предназначена для ввода цифр и выбора арифметических операций с помощью мыши. Панель калькулятора отображается на экране либо по нажатию на кнопку вызова калькулятора, либо при нажатии на клавишу Ввод (Enter) когда курсор находится в поле ввода, либо при нажатии на кнопки арифметических операций, когда курсор  находится в поле ввода. |

Следует обратить внимание, что поле ввода с калькулятором ведет себя несколько по иному, чем обычное поле ввода. Так, если находясь в обычном поле ввода нажать клавишу Ввод, то сработает кнопка Ок (или Готово) диалогового окна. Если же курсор находится в поле ввода с калькулятором, то по нажатию на клавишу Ввод на экране откроется панель с кнопками. Для того, чтобы закрыть диалоговое окно необходимо с помощью клавиши табуляции переместить курсор на следующее поле и нажать клавишу Ввод.

# Глава III. Архитектура платформы

## Платформа Гедымин

Платформа Гедымин23 написана на Делфи 5, на языке Object Pascal. В нулевых версиях мы еще не сильно задумывались о гибкости, настраиваемости и расширяемости. Планировалось, что прикладные решения (или, как их тогда называли, подсистемы) будут дописываться на Делфи, включаться в общий проект и компилироваться в единый выполняемый файл. Максимум, на что мог рассчитывать настройщик – это возможность изменить формулы в жестко заданных расчетных алгоритмах и добавить поля в существующие таблицы. Поскольку такие поля расширяли имеющиеся сущности – их прозвали атрибутами. Второе свое название – пользовательские поля – они получили, так как создавались в угоду конкретным пользователям системы. Мало добавить поле-атрибут в базу данных, его надо вывести на экран. Для этого в диалоговые окна программист добавлял специальную панель – контейнер атрибутов, где система автоматически помещала для каждого пользовательского поля свой элемент управления. Как правило, такие элементы управления располагались в столбец, один под другим. В те далекие времена, даже возможность изменить порядок следования управляющих элементов в контейнере воспринималась как революционное и мощное средство настройки пользовательского интерфейса программы.

Как это часто бывает в жизни, действительность оказалась жестокой и немилосердной. Покинув тепличные лабораторные условия нулевой Гедымин сразу же напоролся на ее рифы. Практически каждый второй клиент требовал обязательной реализации специфических для него функций, которые были не нужны или просто несовместимы с требованиями других пользователей. И, если мы еще могли отделить

23 Гедымин – рабочее имя проекта. Для рабочих версий программ серии Анжелика мы использовали названия городов Беларуси. До сих пор, выполняемый модуль программы Средства Анжелики называется brest.exe. Позже, мы начали использовать имена белорусских князей. Как и в случае с Анжеликой, в момент подготовки к первой презентации, у нас опять не было подходящего имени. В этот раз, мы не стали испытывать судьбу и оставили в силе наименование рабочего проекта. Тем более, что разработчики уже успели привыкнуть к нему.

специфическую функциональность на уровне программного кода с помощью условной компиляции, то разделить структуры в базе данных было гораздо сложнее. Любое минимальное изменение требовало перекомпиляции проекта. Клиент ожидал от нескольких дней до двух-трех недель, прежде чем ему поставляли новую версию. Так как разработка Гедымина при этом не прекращалась ни на минуту, новый выполняемый модуль мог содержать и новые ошибки, на исправление которых опять требовалась перекомпиляция. Это был замкнутый порочный круг.

#### Гедымин 1.х

Фактически, был сделан большой шаг вперед относительно семейства программа Анжелика. Мы перешли на 32-х битный код, клиент-серверную СУБД, единую для всех подсистем базу данных, унифицировали интерфейс пользователя и, самое главное, изобрели бизнес объект, но этого было недостаточно. Исходя из опыта пилотных внедрений были сформулированы следующие требования:

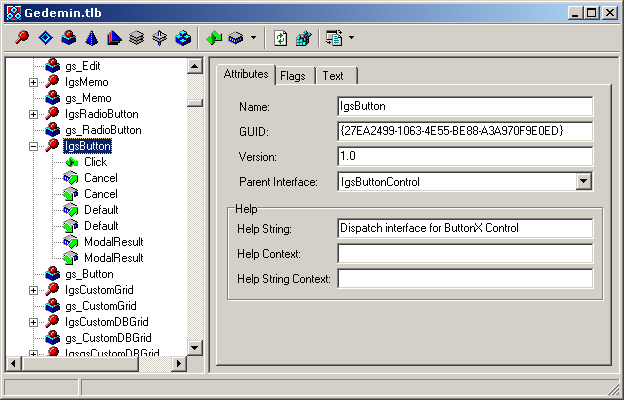
1. Прикладное решение должно быть отделено от платформы. Сама платформа должна стать средством разработки, средой выполнения и включать в себя только наиболее общие, базовые объекты.
2. Настройщик должен иметь возможность создавать метаданные (домены, таблицы, триггеры, хранимые процедуры и т.д.).
3. Настройщик должен иметь возможность писать программный код, изменять поведение существующих объектов и создавать свои объекты, наследуя их от существующих.
4. Настройщик и опытный пользователь должны иметь возможность изменять условия выборки данных и настраивать внешний вид таблиц на экране, а также создавать свои печатные формы.
5. Средства разграничения прав доступа, предоставляемые стандартом SQL, должны быть расширены. Необходимо реализовать разграничение доступа на уровне записи в таблице, политики системной безопасности и аудит действий пользователя.

После анализа доступных альтернатив для выполнения программного кода был выбран Windows Script Host24 (WSH). Из двух доступных

24 В русскоязычной документации ОС Windows применяется название Сервер Сценариев.

скриптовых25 языков, JScript и VBScript, мы остановились на последнем, как на более простом в освоении. Взаимодействие между Гедымином и WSH осуществляется через объект Windows Script Control. Начиная с Windows XP указанные компоненты являются неотъемлимыми частями операционной системы. Для более ранних версий доступны установочные пакеты от Microsoft.

VBScript является языком с поздним связыванием. Обращаться к объекту из скрипта можно, только если он поддерживает диспетчеризацию. Так как стандартные объекты Делфи не поддерживают интерфейс IDispatch, нам пришлось создать для каждого из них интерфейс в библиотеке типов и класс-обертку, поддерживающий данный интерфейс. На Рис. 53 показан фрагмент содержимого библиотеки типов gedemin.tlb с объявлением интерфейса IgsButton, который используется для обращения к стандартной кнопке – TButton.



*Рис. 53. Определение интерфейса IgsButton в библиотеке типов*

Ниже приведен фрагмент кода с объявлением данного интерфейса и реализацией класса-обертки TwrpButton.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Interface: IgsButton

// Flags: (4416) Dual OleAutomation Dispatchable

// GUID: {27EA2499-1063-4E55-BE88-A3A970F9E0ED}

25 Английское слово script дословно переводится на русский язык как сценарий. Тем не менее, среди разработчиков платформы закрепилось использование именно кальки с английского.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* IgsButton = interface(IgsButtonControl)

['{27EA2499-1063-4E55-BE88-A3A970F9E0ED}']

procedure Click; safecall;

function Get\_Cancel: WordBool; safecall; procedure Set\_Cancel(Value: WordBool); safecall; function Get\_Default: WordBool; safecall; procedure Set\_Default(Value: WordBool); safecall; function Get\_ModalResult: Integer; safecall;

procedure Set\_ModalResult(Value: Integer); safecall; property Cancel: WordBool read Get\_Cancel

write Set\_Cancel;

property Default: WordBool read Get\_Default write Set\_Default;

property ModalResult: Integer read Get\_ModalResult write Set\_ModalResult;

end;

...

TwrpButton = class(TwrpButtonControl, IgsButton) private

function GetButton: TButton; protected

procedure Click; safecall;

function Get\_Cancel: WordBool; safecall; procedure Set\_Cancel(Value: WordBool); safecall; function Get\_Default: WordBool; safecall; procedure Set\_Default(Value: WordBool); safecall; function Get\_ModalResult: Integer; safecall;

procedure Set\_ModalResult(Value: Integer); safecall; end;

...

function TwrpButton.GetButton: TButton; begin

Result := GetObject as TButton; end;

procedure TwrpButton.Click; begin

GetButton.Click; end;

...

Создание объектов-оберток происходит незаметно для программиста при обращении к Делфи объекту с помощью методов GetComponent,

FindComponent или CreateObject. Удаляются они также автоматически по завершении использования.

Интерфейс класса-обертки содержит все методы и свойства исходного класса. Типы данных Делфи, которым нет прямого соответствия в VBScript, передаются следующим образом:

###### Перечисление

Приводится к целочисленному типу. Например, если тип TAlignment в Делфи задан следующим образом:

type TAlignment = (taLeftJustify, taRightJustify, taCenter);

То, элементам перечисления taLeftJustify, taRightJustify и taCenter в скрипте будут сооветствовать целочисленные значения 0, 1 и 2, соответственно. Следующие две строки кода идентичны.

Delphi:

Label.Alignment := taCenter;

VBScript:

Label.Alignment = 2

Для улучшения читабельности кода рекомендуется объявлять и использовать константы.

###### Множество

Наименования элементов множества объединяются в строку. В качестве символа разделителя используется пробел. Например, тип TFontStyles объявлен следующим образом:

type

TFontStyle = (fsBold, fsItalic, fsUnderline, fsStrikeOut); TFontStyles = set of TFontStyle;

Тогда следующие две строки кода идентичны. Delphi:

Font.Style := [fsBold, fsItalic];

VBScript:

Font.Style = "fsBold fsItalic"

В преобразовании типов нет никакой магии. Оно осуществляется на уровне класса-обертки. Вот фрагмент исходного кода платформы Гедымин для метода TwrpFont.Get\_Style:

function TwrpFont.Get\_Style: WideString; begin

Result := ' ';

if fsBold in GetFont.Style then Result := Result + 'fsBold ';

if fsItalic in GetFont.Style then Result := Result + 'fsItalic ';

if fsUnderline in GetFont.Style then Result := Result + 'fsUnderline '; if fsStrikeOut in GetFont.Style then Result := Result + 'fsStrikeOut ';

end;

#### Перекрытие методов

**Подсистема печати**

Оба решения, использовавшихся нами для вывода на печать, обладали своими сильными и слабыми сторонами. A-Post Script прекрасно справлялся с регламентированными печатными формами, где требовалось строго выдерживать размер каждой графы, но был бессилен при построении многостраничных отчетов. С этим хорошо справлялся RTF, но он требовал наличия внешней программы для изменения шаблона документа. К тому же, мы столкнулись с проблемой несовместимости кодировки знаков в файлах, подготовленных в разных версиях Microsoft Word. Рассмотрев доступные альтернативы, мы остановились на библиотеке Fast Report, как на наиболее оптимальной подсистеме печати для Гедымина. Это бэнд ориентированный генератор отчетов с богатыми возможностями по визуальному оформлению, кросс-таблицами и встроенным интерпретатором языка Паскаль для обработки информации перед выводом на печать. Наличие исходных кодов прельщало нас возможностью

самомстоятельно исправлять ошибки и дорабатывать библиотеку под наши нужды26.

## Структура программы на Гедымине

Программа на платформе Гедымин состоит из скрипт-объектов. Скрипт- объект – это собирательное название для всех объектов, содержащих выполняемый код. Ниже приводится их полный список с необходимыми пояснениями.

#### Скрипт-функция

Является минимальной единицей кода, которая хранится в базе данных. как единое целое. Может содержать одну или несколько процедур. Имя скрипт-функции должно быть уникально в пределах модуля и совпадать с именем одной из процедур, которая называется главной процедурой. При выполнении скрипт-функции ее текст целиком загружается в Windows Script Control и управление передается главной процедуре.

#### Макрос

Макрос по сути является скрипт-функцией, которую можно вызвать из меню макросов формы просмотра. В имени макроса можно использовать пробелы, кавычки, скобки, буквы не латинского алфавита и т.п. Для быстрого вызова можно назначить последовательность горячих клавиш.

#### Отчет

Обязательными элементами отчета являются: скрипт-функция для подготовки данных и шаблон, в который эти данные подставляются. Дополнительно могут быть определены еще две скрипт-функции: параметров и обработки событий. Отчет имеет имя и вызывается из меню отчетов на форме просмотра.

#### VB класс

Скрипт-функция, которая содержит определение класса VBScript.

26 Стоит заметить, что Fast Report – это единственный коммерческий компонент в составе платформы. Сторонний разработчик, который получил исходный код Гедымина и хочет заниматься его дальнейшим развитием, обязан самостоятельно приобрести лицензию на использование Fast Report.

#### Константы и переменные

Особый вид скрипт-функции, которая содержит объявление глобальных констант и/или переменных, доступных по всех модулях программы.

#### Метод

Содержит процедуру, перекрывающую виртуальный метод Делфи класса. Из кода процедуры возможен вызов оригинального метода.

#### Событие

Скрипт-функция, которая выполняется, когда в системе происходит определенное событие.

#### Глобальный VB объект

Скрипт-функция глобального объекта содержит три обязательных элемента: объявление глобальной переменной, процедуру инициализации (вызывается после подключения к базе данных) и процедуру деинициализации (вызывается перед отключением от базы или выходом из программы). Как правило, используется для создания глобальных экземпляров классов VBScript.

Ниже приводится пример скрипт-функции глобального VB объекта:

Option Explicit

' объявление глобальной переменной

Public BNOptions

' создание глобального объекта

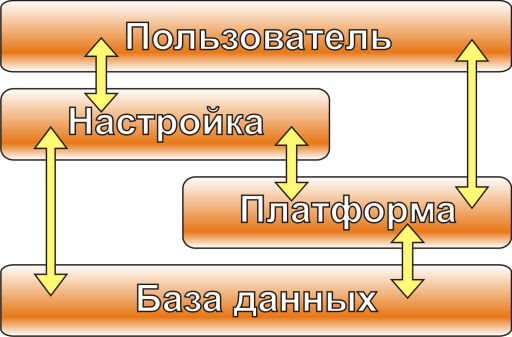
Sub BNOptions\_Initialize

Set BNOptions = New TVBBN\_Options End Sub

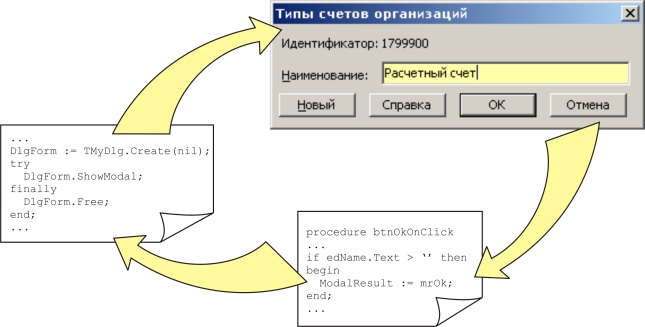
' уничтожение глобального объекта

Sub BNOptions\_Terminate Set BNOptions = Nothing

End Sub



*Рис. 54. Структура платформы Гедымин*



*Рис. 55. Выполнение кода в обычном приложении*



*Рис. 56. Выполнение кода на платформе Гедымин*

## Настройки

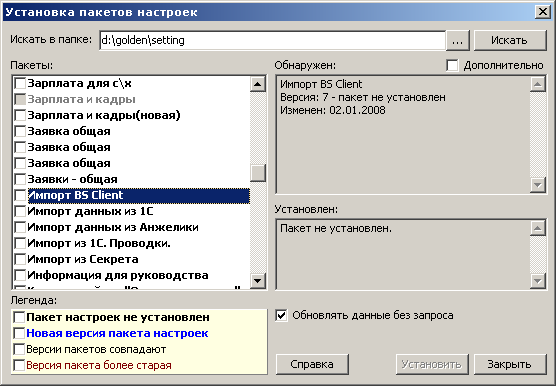
Под настройкой мы понимаем совокупность метаданных, данных, объектов хранилища и скрипт-объектов, предназначенных для решение определенной прикладной задачи. Например, настройка Платежные документы может включать в себя следующие объекты: таблицы для хранения платежных поручений и требований, специфических справочников (метаданные), экранные формы, скрипт-функции, отчеты (скрипт-объекты), наполнение специфических справочников (данные). Настройка может быть сохранена в файле и затем загружена на другую базу данных.

##### Пакет настроек

На практике бывает удобно разделить объекты, относящиеся к одной прикладной задаче, по нескольким настройкам. Например, в одну

настройку поместить метаданные, в другую – скрипт-объекты, в третью – данные справочников и т.п.

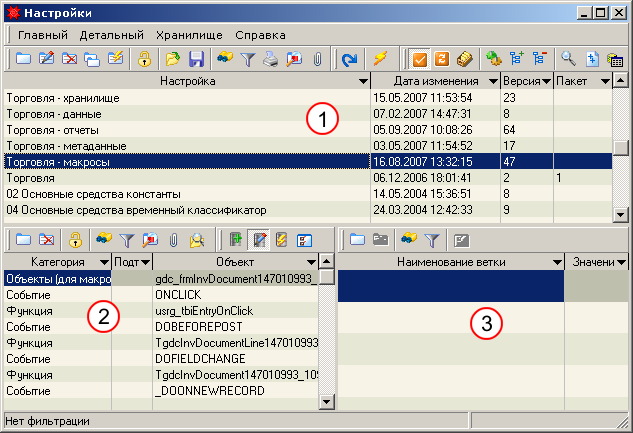
##### Загрузка пакета настроек



*Рис. 57. Окно загрузки пакетов настроек*

##### Создание настройки Окно настроек

Окно Настройки вызывается из подраздела Атрибуты раздела Сервис Исследователя системы, и имеет следующий вид:



*Рис. 58. Окно Настройки*

Как видно на рисунке, окно разделено на три списка. Первый – это настройки в базе данных. Для выбранной настройки во втором списке

отображаются входящие в нее объекты, а в третьем – объекты хранилища. Специфические команды представлены на панели инструментов каждого из списка. Ознакомимся с ними.

|  |  |  |  |
| --- | --- | --- | --- |
|  | | **Команда** | **Описание** |
|  | | Активировать/деактивировать  настройку |  |
|  |  | Переактивировать настройку |  |
|  | | Сформировать настройку |  |
|  | | Установить порядок позиций |  |
|  | | Выбрать настройки, от которых зависит данная  настройка |  |
|  | | Очистить зависимости |  |
|  | | Просмотр настройки |  |
|  | | Добавить пропущенные  позиции |  |
|  | | Выгрузить в текстовый файл |  |

**Список объектов**

|  |  |  |
| --- | --- | --- |
|  | **Команда** | **Описание** |
|  | Сохранять с детальными  объектами |  |
|  | Перезаписывать данными из  потока |  |
|  | Установить флаг "Обновлять  данными из потока" |  |
|  | Проверка целостности  настройки |  |

**Список объектов хранилища**

|  |  |  |
| --- | --- | --- |
|  | **Команда** | **Описание** |
|  | Добавить форму пользователя |  |
|  | Удалить |  |
|  | Проверка целостности  настройки хранилища |  |

# Глава IV. Hello world!

Со времен ветхого завета в мире программирования принято начинать знакомство с новой системой написанием простейшей программы вывода на экран строки "Hello World!". Будем придерживаться канонов. Читатель, уже знакомый с тем, как на Гедымине пишется программный код и создаются экранные формы, без малейших для себя последствий может пропустить эту главу. Мы же выполним следующие действия:

1. Запустим Гедымин и подключимся к демо базе под учетной записью

###### Administrator.

1. В главном окне программы раскроем меню **Сервис** и вызовем команду

###### Редактор скрипт-объектов.

1. Найдем окно **Проводник**, в дереве выделим пункт **Глобальные макросы**27 и нажмем правую кнопку мыши.
2. Из появившегося меню выберем команду **Добавить макрос,** как показано на Рис. 59.

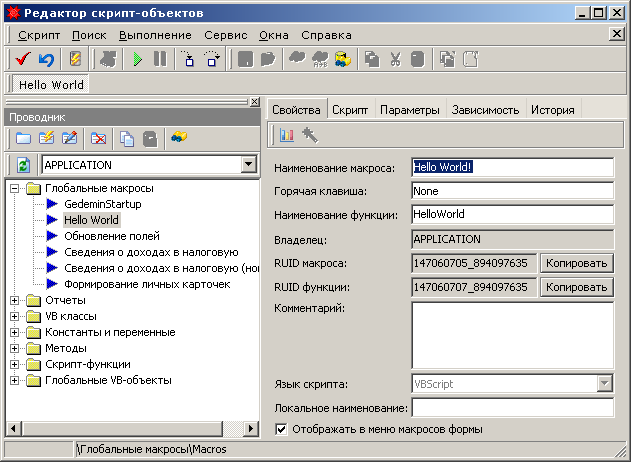


*Рис. 59. Вызов команды Добавить макрос*

27 Обратите внимание, что ветвь Глобальные макросы присутствует только в модуле APPLICATION. Выбор текущего модуля осуществляется из выпадающего списка над деревом объектов Проводника.

Совет: При первом обращении Редактор скрипт-объектов может содержать вспомогательные окна, такие как: Список переменных, Стек вызовов, Сообщения и др. Для экономии экранного пространства их можно закрыть. Если вы случайно закрыли окно Проводника, вернуть на экран его можно с помощью комбинации горячих клавиш Ctrl-Alt-T или воспользовавшись меню Окна Редактора скрипт-объектов (не путайте с меню Окна главного окна Гедымина).

По выполнении вышеуказанных действий на рабочем поле Редактора скрипт-объектов откроется окно редактирования макроса.



*Рис. 60. Окно создания нового макроса*

Пусть вас не смущает обилие полей ввода. В следующих главах мы подробно ознакомимся с предназначением каждого из них. А пока, наберем в поле **Наименование макроса** строку "Hello World!", а в поле **Наименование функции** – "HelloWorld" (Естественно, кавычки не являются частью строки и вводить их не следует. Они используются в тексте исключительно для выделения. Пробелы в имени функции недопустимы).

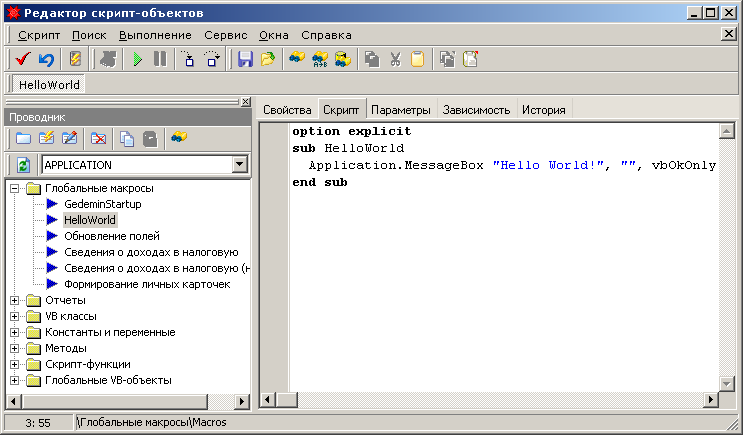
Далее, перейдем на вкладку **Скрипт** и наберем такой текст процедуры:

Option Explicit Sub HelloWorld

Application.MessageBox "Hello World!", "", vbOkOnly

End Sub

Результат наших усилий приведен на Рис. 61.



*Рис. 61. Ввод текста процедуры*

По окончании ввода нажмем клавишу **F9** для выполнения нашей программы. Если текст набран без синтаксических ошибок, то на экране появится следующее диалоговое окно:



*Рис. 62. Hello World!*

Ура! Наше первое приложение успешно работает.

## Обращаемся к базе данных

Пусть мы и достигли поставленной цели, но очевидно, что однострочная процедура вряд ли позволит нам составить даже самое поверхностное впечатление о возможностях платформы. Давайте усложним задачу и заставим нашу программу приветствовать не только весь мир, но и лично пользователя. Персональную информацию, имя учетной записи, возьмем

непосредственно из базы данных. Код второй версии процедуры HelloWorld выглядит следующим образом28:

1. Option Explicit
2. Sub HelloWorld

03

04

05

06

07

08

09

10

11

12

13

14

Dim gdcUser

Set gdcUser = Designer.CreateObject(Application,\_ "TgdcUser", "")

gdcUser.SubSet = "ByID" gdcUser.ID = IBLogin.UserKey gdcUser.Open

Application.MessageBox "Hello World!" & vbCrLf &\_ "Hello " & gdcUser.ObjectName & "!", \_

"", vbOkOnly

Designer.DestroyObject(gdcUser)

15 End Sub

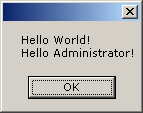
Прежде чем привести нашу процедуру в действие поясним некоторые нюансы.

Мы можем обращаться к функциям и параметрам Гедымина через набор глобальных объектов. В данном случае используются три таковых: глобальный объект **Designer** предназначен для динамического создания и уничтожения объектов в процессе работы программы, **IBLogin** – содержит параметры текущего подключения к базе данных, а **Application** – среди прочих – обладает функцией вывода на экран окна с сообщением.

В четвертой строке представленного кода, вызовом метода **Designer.CreateObject**, создается бизнес-объект типа **TgdcUser** и, для дальнейшего обращения, сохраняется в объявленной строкой выше переменной **gdcUser**. В шестой строке мы указываем, что будем работать только с одной записью, которую отыщем по идентификатору. В следующей строке мы передаем бизнес-объекту этот идентификатор, взяв его из свойства **IBLogin.UserKey**. В восьмой строке мы открываем набор данных методом **gdcUser.Open** и используем считанное из базы имя при формировании текста приветствия и выводе его на экран (строки с десятой по двенадцатую). И, наконец, закончив работу с бизнес-объектом, в четырнадцатой строке мы удаляем его с помощью метода **Designer.DestroyObject**.

28 При наборе текста процедуры, номера строк следует опустить.

Теперь, когда код процедуры уже не представляет для нас загадки, можем смело нажимать клавишу **F9** и созерцать на экране результат наших трудов:

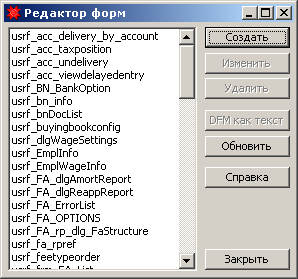


*Рис. 63. Персональное приветствие*

## Создаем экранную форму

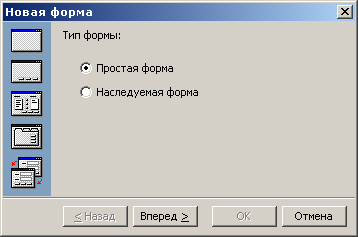
Любая современная среда быстрой разработки должна содержать инструменты визуального проектирования экранных форм. Гедымин в этом плане не является исключением. В предыдущих примерах мы использовали стандартное окно для вывода текстовых сообщений, посланное нам свыше создателями операционной системы. Попробуем повторить Творца и самостоятельно создать экранную форму.

Для этого обратимся к **Редактору форм**. Соответствующий пункт находится в меню **Сервис** главного окна программы.



*Рис. 64. Редактор форм*

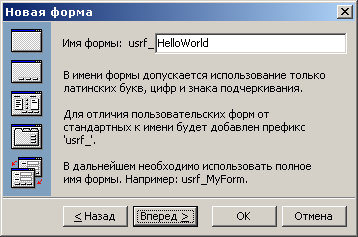
Как видно на Рис. 64 окно редактора содержит список форм, находящихся в данный момент в базе данных, и набор кнопок для осуществления действий над ними. Нас интересует команда **Создать**. Выберем соответствующую кнопку. На экране откроется мастер создания экранной формы:



*Рис. 65. Мастер создания экранной формы*

Оставим ответ на вопрос «Чем отличается простая форма от наследуемой?» для последующих глав. Не будем менять выбор по- умолчанию – **Простая форма** – и нажмем кнопку **Вперед**.

Следующая страница мастера предлагает нам ввести имя типа создаваемой формы.

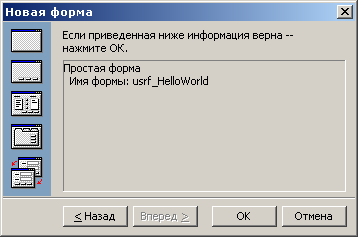


*Рис. 66. Ввод имени типа формы*

Внимательно ознакомимся с правилами, которым должно соответствовать имя и введем в поле ввода строку ―HelloWorld‖. Обратите внимание, что каждое имя типа формы автоматически получает префикс

―usrf\_‖, т.е. в тексте программы мы будем обращаться к созданному нами типу, как ―usrf\_HelloWorld‖.

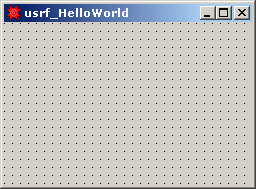
После того как имя введено, нажимаем **Вперед** и попадаем на финальную страницу мастера:



*Рис. 67. Контрольная страница мастера*

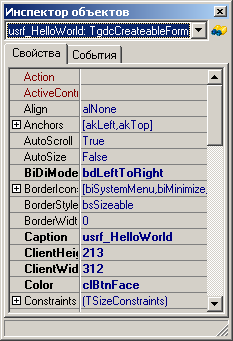
Убедимся, что мы не допустили ошибок в наименовании типа и нажмем **Ок** (если ошибки все-таки есть, то с помощью кнопки **Назад** можно вернуться и исправить их).

С этого момента начинается самое интересное. Гедымин переключится в режим **Дизайнера форм**. На экране появятся три окна. Проектируемая нами форма в исходном, девственно чистом состоянии:



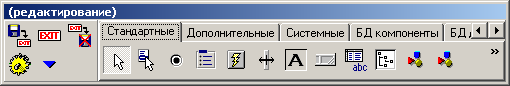
*Рис. 68. Форма в режиме дизайнера*

###### Инспектор объектов:



*Рис. 69. Инспектор объектов*

###### Палитра компонентов:



*Рис. 70. Палитра компонентов*

#### Принципы визуального проектирования

Для читателя, не знакомого с такими средами разработки, как Borland Delphi29 или Microsoft Visual Studio, поясним основные принципы. **Палитра компонентов** содержит доступные нам визуальные и невизуальные компоненты, например, кнопки, списки, переключатели, флажки, поля ввода и т.д. Поскольку компонентов много, во избежании путаницы они тематически сгруппированы и размещены на нескольких вкладках. Для выбора компонента мы открываем нужную вкладку и щелкаем мышью по его пиктограммке. Затем щелкаем по тому месту на форме, где мы хотели бы этот компонент разместить.

Изменить свойства компонента на форме можно следующим образом:

1. Находим компонент и выделяем одинарным щелчком мыши. Для изменения свойств самой формы, необходимо щелкнуть мышью по свободному пространству на ней;
2. переходим в окно **Инспектора объектов**;
3. находим в списке нужное свойство и устанавливаем его значение.

Совет: Для быстрого перехода в Инспектор объектов используйте клавишу F11, а для активизации Палитры компонентов – F10.

Перейдем от теории к практике. Для настройки нашей формы сделаем следующее:

1. Перейдем в Инспектор объектов и установим следющие свойства:

BorderStyle = bsDialog Caption = Hello World! Color = clBlue Position = poScreenCenter

1. На вкладке Стандартные палитры компонентов найдем и выделим компонент TLabel (ее пиктограммка имеед вид прописной буквы А).

29 С 2006 года разрабатывается поставляется компанией CodeGear под названием CodeGear Delphi.

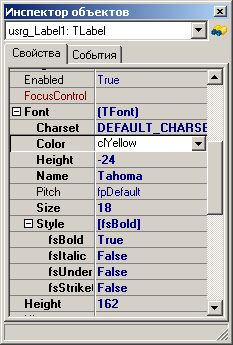
1. Поместим ее на форму и установим следующие свойства:

Align

= alClient

Alignment = taCenter Layout = tlCenter

1. Найдем в **Инспекторе объектов** свойство **Font**. Обратите внимание на изображение плюса слева от наименования свойства. Щелкнув по нему мы раскроем список свойств шрифта:



*Рис. 71. Свойство Font компонента TLabel*

1. Установим следющие значения, как показано на Рис. 71:

Font.Color Font.Name Font.Size

= clYellow

= Tahoma

= 18

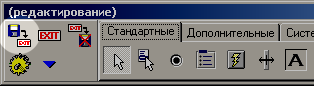
Font.Style.Bold = True

1. После проделанных манипуляций форма должна принять вид, как на Рис. 72:



*Рис. 72. Отредактированная форма*

1. Запомним название компонента TLabel – usrg\_Label1. Сохраним форму и выйдем из режима дизайнера. Пиктограммка команды **Сохранить и выйти** находится на **Палитре компонентов** (см. Рис. 73). Можно также воспользоваться горячей клавишей **F2**.



*Рис. 73. Команда Сохранить и выйти*

1. Закрываем окно **Редактора форм** с помощью кнопки **Закрыть**.

#### Создаем форму из макроса

Все готово к написанию третей версии процедуры HelloWorld, в которой мы не только прочитаем из базы данных наименование текущей учетной записи, но и выведем на экран приветствие в сконструированном нами окне.

1. Option Explicit
2. Sub HelloWorld

03

04

05

06

07

08

09

10

Dim gdcUser, F

Set gdcUser = Designer.CreateObject(Application,\_ "TgdcUser", "")

gdcUser.SubSet = "ByID" gdcUser.ID = IBLogin.UserKey gdcUser.Open

Set F = Designer.CreateObject(Application, \_

11

12

13

14

15

16

17

"usrf\_HelloWorld", "")

F.GetComponent("usrg\_Label1").Caption = \_ "Hello " & gdcUser.ObjectName & "!"

F.ShowModal

Designer.DestroyObject(F) Designer.DestroyObject(gdcUser)

18 End Sub

Рассмотрим приведенный листинг. Строки с первой по девятую нам уже знакомы. Здесь мы создаем бизнес-объект и считываем из базы данных наименование текущей учетной записи. В десятой строке мы создаем экземпляр спроектированной нами формы, указав имя ее типа – "usrf\_HelloWorld". В двенадцатой строке мы обращаемся к компоненту TLabel по имени "usrg\_Label1" и присваиваем свойству Caption текст приветствия, который формируем в тринадцатой строке. Метод ShowModal в следующей строке процедуры выводит форму на экран и останавливает выполнение макроса пока пользователь не закроет окно.

Проверяем правильно ли введен текст макроса и запускаем его на выполнение – F9. Результат приведен на Рис. 74.



*Рис. 74. Hello World 3*

## Заключение

Только что мы ознакомились с азами программирования на платформе Гедымин. Мы узнали где и как писать код, обратились к некоторым глобальным системным объектам, создали экземпляр бизнес-класса и с помощью него прочитали из базы данных наименование учетной записи, а также спроектировали и вывели на экран диалоговое окно.

# Глава V. Среда разработки

Среда разработки платформы Гедымин состоит из шести окон, вызываемых из меню **Сервис**:

 **Редактор скрипт-объектов** предназначен для написания, отладки программного кода и создания отчетов;



**Редактор форм** отвечает за создание и изменение экранных форм;

**Редактор SQL** позволяет выполнять запросы к базе данных;

**SQL монитор** является инструментом профилирования и оптимизации взаимодействия программы с сервером баз данных;

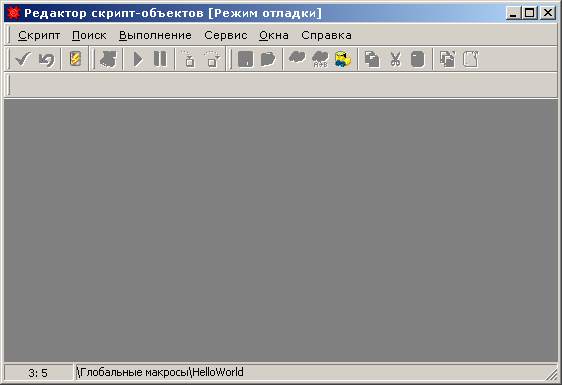
 Окно **Просмотр шаблона документов** позволяет тестировать шаблоны импорта данных из текстовых файлов;

 И, наконец, окно **Сравнение баз данных** предназначено для переноса и объединения настроек из разных баз.

В предыдущих главах нам уже приходилось работать с некоторыми из этих окон. Ниже мы в деталях познакомимся с каждым из них.

## Редактор скрипт-объектов

**Редактор скрипт-объектов** вызывается из меню **Сервис** главного окна программы. В чистом виде редактор содержит строку меню, панель инструментов, панель со списком открытых объектов (пустую) и строку состояния. Все остальное пространство занимает рабочая область (см Рис. 75).



*Рис. 75. Окно Редактора скрипт-объектов*

|  |
| --- |
| Совет: при первом обращении к Редактору скрипт-объектов, откроются |
| вспомогательные окна: Сообщения, Стек вызовов, Список переменных и т.п. |
| Чтобы не загромождать экран, закройте их. В последствии, вы всегда можете |

|  |
| --- |
| открыть нужное окно с помощью последовательности горячих клавиш или  команды из меню Окна. |

#### Меню Редактора скрипт-объектов

Ознакомимся с командами Редактора скрипт-объектов.

##### Пункт меню Скрипт

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Команда** | **Быстрый**  **вызов** | **Описание** |
|  | Сохранить | Ctrl+S | Сохраняет текущую скрипт-функцию. |
|  | Сохранить  все |  | Сохраняет все открытые скрипт-  функции. |
|  | Откат изменений |  | Отменяет все изменения с момента последнего сохранения скрипт-  функции30. |
|  | Сохранить скрипт в  файл… |  | Сохраняет текст текущей скрипт- функции в файле на диске. |
|  |
|  |
|  | Загрузить  скрипт из файла… |  | Заменяет текст текущей скрипт-  функции содержимым выбранного файла. |
|  | Копировать |  | Копирует выделенный фрагмент  текста в буфер обмена. |
|  | Вырезать |  | Вырезает выделенный фрагмент  текста и помещает его в буфер обмена. |
|  | Вставить |  | Вставляет текст из буфера обмена  начиная с текущей позиции курсора. |
|  | Копировать SQL |  | Извлекает из выделенного фрагмента кода на VBScript текст SQL запроса и  помещает его в буфер обмена31. |

30 С помощью вкладки История в окне редактирования скрипт-функции можно вернуться к любой предыдущей ее версии.

31 В тексте программы мы обычно присваиваем SQL запрос следующим образом:

q.SQL.Text = "SELECT id FROM gd\_contact " &\_ "WHERE name = :Name "

|  |  |  |  |
| --- | --- | --- | --- |
|  | Вставить SQL |  | Преобразует текст SQL запроса из буфера обмена в фрагмент кода на VBScript и вставляет его начиная с  текущего положения курсора. |
|  | Закрыть | Ctrl+Alt+F4 | Закрывает текущую скрипт-функцию. |
|  | Закрыть все |  | Закрывает все открытые скрипт-  функции. |
|  | Открыть |  | Содержит список ранее  редактировавшихся скрипт-функций. |

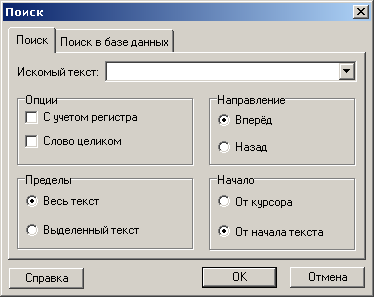
##### Пункт меню Поиск

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Команда** | **Быстрый**  **вызов** | **Описание** |
|  | Поиск… |  | Открывает диалоговое окно поиска в  тексте текущей скрипт-функции. |
|  | Поиск в базе данных… |  | Открывает диалоговое окно поиска  среди всех скрипт-функций в базе данных. |
|  | Замена… |  | Открывает диалоговое окно поиска и замены в тексте текущей скрипт-  функции. |
|  | Поиск неиспользуемых  функций… |  | Открывает окно поиска в базе данных нигде не используемых  скрипт-функций. |
|  | Перейти на  строку… |  | Вызывает команду перехода на  строку по номеру. |

На Рис. 76 показан вид окна при вызове команды **Поиск**. Аналогично окну поиска в таблице, которое мы рассматривали в разделе "Организация пользовательского интерфейса", здесь пользователь имеет возможность выбрать вид поиска (с учетом регистра или без, слово целиком или по вхождению заданной последовательности символов), область поиска (по всему тексту или в выделенном фрагменте), начало (от текущего положения

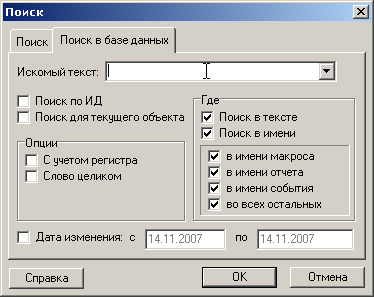
Выделив данный фрагмент и применив к нему команду **Копировать SQL** мы поместим в буфер обмена только текст SQL запроса, без кавычек, операторов объединения строк и т.п. И наоборот, скопировав из окна **Редактор SQL** запрос, с помощью команды **Вставить SQL** мы можем получить уже готовый код на языке VBScript.

курсора или от начала текста) и направление поиска. Поиск осуществляется в тексте текущей скрипт-функции.



*Рис. 76. Диалоговое окно поиска в тексте скрипт-функции*

Команда **Поиск в базе данных** открывает диалоговое окно, показанное на Рис. 77.



*Рис. 77. Диалоговое окно поиска в базе данных*

Его поля имеют следующее предназначение:

|  |  |
| --- | --- |
| Искомый текст | Здесь вводится текст для поиска. Выпадающий  список позволяет обратиться к одному из ранее использовавшихся значений. |
| Поиск по ИД | Поиск скрипт-функции с заданным  идентификатором. |
| Поиск для текущего | Поиск будет осуществляться только среди |

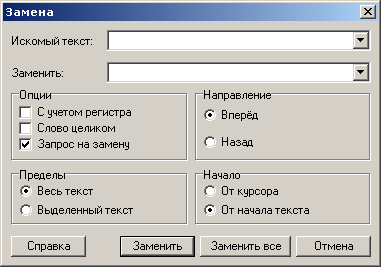
|  |  |
| --- | --- |
| объекта | скрипт-объектов текущего модуля, выбранного в  Проводнике. |
| С учетом регистра | Поиск будет чувствителен к регистру символов. |
| Слово целиком | Искомый текст должен быть отдельным словом  (фразой) в тексте скрипт-функции. |
| Поиск в тексте | Задает область поиска: в тексте скрипт-функции. |
| Поиск в имени | Задает область поиска: в имени скрипт-объекта. Дополнительно можно указать, среди каких скрипт-объектов искать: макросов, отчетов,  обработчиков событий, во всех прочих объектах. |
| Дата изменения | Ограничить область поиска только скрипт-  объектами, измененными в указанном периоде. |

Результат поиска отображается в окне **Сообщения** (см. Рис. 78). Двойным щелчком мыши можно открыть соответствующую скрипт-функцию на редактирование.



*Рис. 78. Результаты поиска в базе данных*

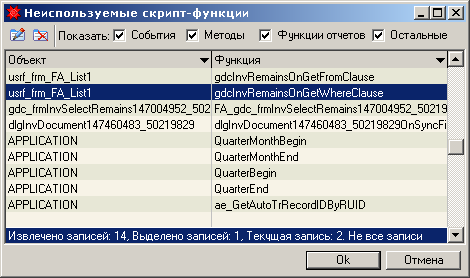
Диалоговое окно, показанное на Рис. 79 откроется на экране при обращении к команде **Замена**.



*Рис. 79. Диалоговое окно поиска и замены в тексте скрипт-функции*

Назначения переключателей аналогичны окну Поиска. Замена осуществляется в тексте текущей скрипт-функции.

Скрипт-функции, которые нигде не вызываются, обработчики событий удаленных компонентов, функции построения данных для изъятых из базы отчетов, перекрытые методы несуществующих классов, могут быть найдены с помощью окна, показанного на Рис. 80.



*Рис. 80. Поиск неиспользуемых скрипт-функций*

Выделив в списке скрипт-функцию ее можно открыть для просмотра/изменения или удалить.

##### Пункт меню Выполнение

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Команда** | **Быстрый**  **вызов** | **Описание** |
|  | Компилировать проект |  | По новой загружает все скрипт- функции в Windows Script Host. Команда применяется для поиска синтаксических ошибок и ситуаций, когда в коде присутствует вызов внешней функции, которая не указана в  директиве #include. |
|  | Построить отчет | Ctrl+Alt+F9 | Команда доступна если текущим объектом является отчет.  Запускает его на выполнение и  отображает результат на экране. |
|  | Запустить | F9 | Запускает текущий скрипт на  выполнение. |
|  | Проверка синтаксиса | Ctrl+F9 | Проверяет текущую скрипт- функцию на наличие  синтаксических ошибок. |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Пауза |  | Останавливает выполнение скрипт-функции, устанавливает  курсор на текущую команду. |
|  | Шаг в | F7 | При пошаговой отладке, если курсор установлен на строке с вызовом функции или процедуры,  – переходит в нее. В ином случае –  выполняет одну команду. |
|  | Шаг через | F8 | При пошаговой отладке  выполняет одну команду. |
|  | Перейти к курсору | F4 | Выполняет программу от текущей инструкции до строки, в которой  находится курсор. |
|  | Перейти на выполняемую строку |  | При отладке программы открывает скрипт-функцию и устанавливает курсор на инструкцию, которая будет  выполнена следующим шагом. |
|  | Сброс программы | Ctrl+F2 | Прерывает выполнение  программы. |
|  | Установить/снять  точку останова | F5 | Устанавливает или снимает точку  останова в текущей строке. |
|  | Вычислить… | Ctrl+F7 | Позволяет просмотреть значение переменной или вычислить  выражение. |
|  | Добавить в просмотр  переменных | Ctrl+F5 | Добавляет имя переменной или выражение в список просмотра  переменных. |
|  | Информация о типе |  | Если курсор установлен на имени объекта автоматизации, открывает окно для просмотра его  интерфейса. |

Команды отладки приложения доступны только при включенной опции **Использовать отладочную информацию** (см. команду **Опции…** в меню **Сервис**).

Последовательности клавиш быстрого вызова настраиваются в окне **Установки редактора**, вызываемого из меню **Сервис**. В таблице выше сочетания клавиш приведены для раскладки по умолчанию.

##### Пункт меню Сервис32

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Команда** | **Быстрый**  **вызов** | **Описание** |
|  | Опции… |  | Открывает окно с опциями среды  разработки. |
|  | Установки  редактора… |  | Открывает окно с настройками  редактора. |
|  | Шаблоны  кода… |  | Открывает окно со списком шаблонов  кода. |
|  | SQL Редактор |  | Вызывает окно SQL редактора. |

Как показано на Рис. 81, диалоговое окно с опциями среды разработки содержит пять вкладок: Общие, Отладка, Ошибки, Фильтр скрипт-функций и Фильтр.



*Рис. 81. Диалоговое окно Опции среды разработки*

Ниже приводится расшифровка всех доступных в данном окне флагов, переключателей и полей ввода.

|  |  |
| --- | --- |
| **Вкладка Общие** | |
| **Наименование поля** | **Значение** |
| Автосохранение изменений при  закрытии редактора | Если флаг установлен, все несохраненные скрипт-функции будут записаны в базу данных  при закрытии окна редактора скрипт-объектов. |
| Автосохранение  изменений перед | Если флаг установлен, все несохраненные  скрипт-функции будут записаны в базу данных |

32 Главное окно программы и Окно Редактора скрипт-объектов имеют в меню одноименные команды: Сервис, Окна и Справка. Будьте внимательны и не путайте их.

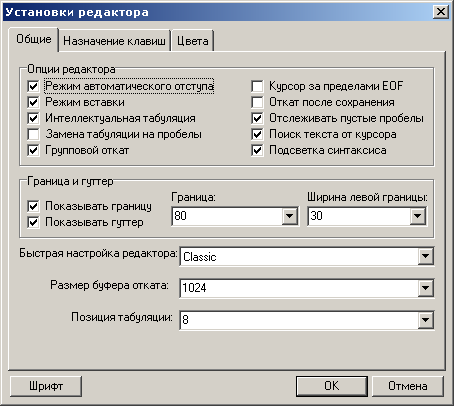
|  |  |
| --- | --- |
| запуском | перед запуском программы на выполнение. |
| Сохранять положение курсора и закладки | Если флаг установлен, будут сохранены и при следующем запуске восстановлены, все открытые скрипт-функции. Данная опция способна замедлить загрузку редактора скрипт-объектов, если количество открытых скрипт-функций  велико. |
| Выводить имя класса + подтип | Флаг управляет отображением информации в подразделе Методы дерева Проводника. Если установлен, то для подтипов класса будет использоваться полное наименование, например, TgdcAttrUserDefinedUSR\_ACC\_DELIVERY. Если флаг снят, -- будет отображаться только подтип --  USR\_ACC\_DELIVERY. |
| Предупреждение об обновлении дерева | Обновление дерева Проводника (клавиша F5) закроет все открытые на момент операции скрипт-функции. Если флаг установлен,  пользователь будут предупрежден об этом. |
| Восстанавливать рабочий стол | Имеется ввиду не рабочий стол Гедымина, о котором речь шла в разделе "Организация пользовательского интерфейса", а конфигурация открытых окон в Редакторе скрипт-объектов.  Установленный флаг приведет к автоматическому сохранению конфигурации при выходе и ее загрузке при следующем входе в  Редактор скрипт-объектов. |
| **Вкладка Отладка** | |
| Использовать  отладочную информацию | Опция должна быть включена для пошаговой  отладки. Однако, ее применение замедляет выполнение программы33. |
| Сохранять время | Установленный флаг приведет к тому, что время |

33 Гедымин в версиях 1.х и 2.х применяет свою оригинальную систему отладки и не обращается к соответствующим интерфейсам Windows Script Host. В режиме дебагера каждая строка програмы обрамляется двумя строками, которые передают системе информацию о состоянии переменных и позволяют выполнять останов при пошаговом выполнении. Таким образом, объем кода увеличивается втрое, что неизбежно сказывается на производительности. Данный механизм работает совершенно прозрачно для разработчика, который продолжает видеть только свой, исходный код без дополнительной отладочной информации.

|  |  |
| --- | --- |
| выполнения СФ в файл ScriptRuntime.log | выполнения каждой скрипт-функции будет заноситься в журнал. Файл журнала располагается в одной папке с исполняемым  модулем gedemin.exe. |
| **Вкладка Ошибки** | |
| Останавливать при возникновении ошибок | Показывает как будет вести себя программа при возникновении ошибки или исключительной ситуации. Подробное объяснение смотрите ниже,  в подразделе Обработка ошибок. |
| Останавливать при возникновении внутренних ошибок | Показывает как будет вести себя программа при возникновении ошибки или исключительной ситуации. Подробное объяснение смотрите ниже,  в подразделе Обработка ошибок. |
| Сохранять информацию об  ошибках в файл | Если опция активна, в папке файла gedemin.exe будет создан журнал с информацией о  возникших в процессе работы ошибках. |
| Имя файла в каталоге  Гедымина | Позволяет задать имя файла журнала. |
| Ограничивать количество строк | Во избежание переполнения диска, размер  журнала можно ограничить указанным количеством строк. |
| **Вкладка Фильтр скрипт-функций** | |
| Выводить скрипт- функции созданные  пользователем | Выводить или нет скрипт-функции созданные пользователем в подразделе Скрипт-функции  дерева проводника. |
| Выводить скрипт-  функции VB классов | Выводить или нет скрипт-функции VB классов в  подразделе Скрипт-функции дерева проводника. |
| Выводить скрипт-  функции макросов | Выводить или нет скрипт-функции макросов в  подразделе Скрипт-функции дерева проводника. |
| Выводить скрипт-  функции отчетов | Выводить или нет скрипт-функции отчетов в  подразделе Скрипт-функции дерева проводника. |
| Выводить скрипт- функции методов | Выводить или нет скрипт-функции перекрытых  методов в подразделе Скрипт-функции дерева проводника. |
| Выводить скрипт- функции событий | Выводить или нет скрипт-функции событий  объектов в подразделе Скрипт-функции дерева проводника. |
| Выводить скрипт-  функции проводок | Выводить или нет скрипт-функции  формирования проводок в подразделе Скрипт- |

|  |  |
| --- | --- |
|  | функции дерева проводника. |
| **Вкладка Фильтр** | |
| Наименование класса | Данная опция позволяет установить фильтр на наименования классов в разделе Методы дерева  Проводника. |
| Наименование метода | Данная опция позволяет установить фильтр на  наименование метода в разделе Методы дерева Проводника. |
| Наименование объекта | Данная опция позволяет установить фильтр на наименование объекта (компонента) в разделе  События34 дерева Проводника. |
| Наименование события | Данная опция позволяет установить фильтр на наименование события в разделе События дерева  Проводника. |
| Выводить только переопределенные  события и методы | При установленном флаге в разделах Методы и События дерева Проводника будут отображаться  только переопределенные методы и события. |
| Выводить только  отключенные события и методы | Действует только, если предыдущий флаг  установлен. Ограничивает вывод только отключенными методами и событиями. |

Внешний вид окна Установки редактора показан на Рис. 82.



*Рис. 82. Опции редактора*

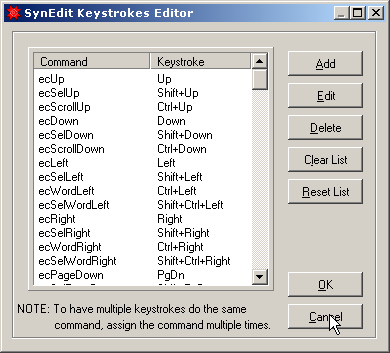
34 Раздел События не доступен в глобальном модуле Application.

Приведем описание доступных флагов, списков выбора и полей ввода.

|  |  |
| --- | --- |
| **Вкладка Общие** | |
| **Наименование поля,**  **флага, кнопки** | **Значение, команда** |
| Режим автоматического отступа | При переходе на новую строку по нажатию клавиши Enter устанавливать курсор в позицию первого непробельного символа из предыдущей  непустой строки. |
| Режим вставки | Открывать редактор в режиме вставки. |
| Интеллектуальная табуляция | Клавиша табуляции перемещает курсор в позицию первого непробельного символа из  предыдущей строки. |
| Замена табуляции на  пробелы | Использовать пробелы вместо символов  табуляции. |
| Групповой откат | По вызову команды Откат изменений отменять последнее действие и все действия такого же  типа. |
| Курсор за пределами  EOF | Разрешать перемещать курсор за последнюю  строку файла. |
| Откат после  сохранения | Разрешать откат изменений, сделанных до  сохранения файла. |
| Отслеживать пустые  пробелы | Сохранять пробелы в конце строки. |
| Поиск текста от  курсора | Вести поиск текста от текущего положения  курсора. |
| Подсветка синтаксиса | Включить подсветку синтаксиса. Настройка  цветовой схемы производится на вкладке Цвета. |
| Показывать границу | Показывать правую границу текста. |
| Показывать гуттер | Показывать гуттер – поле слева от текста, где  размещаются служебные пиктограммки. |
| Граница | Положение правой границы, в символах. |
| Ширина левой  границы | Ширина гуттера, в символах. |
| Быстрая настройка редактора | Выбор раскладки горячих клавиш для команд редактора. Просмотреть список горячих клавиш и выполнить индивидуальную настройку можно  на вкладке Назначение клавиш. |
| Размер буфера отката | Количество команд, сохраняемых в буфере  отката. |

|  |  |
| --- | --- |
| Позиция табуляции | Размер табуляции, в символах. |
| **Вкладка Назначение клавиш** | |
| Раскладки клавиатуры | Список предустановленных раскладок  клавиатуры. |
| Сбросить по  умолчанию | Вернуться к изначальному назначению клавиш  для выбранной раскладки. |
| Редактировать | Открыть окно со списком команд для назначения  горячих клавиш. Более подробно см. ниже. |
| **Вкладка Цвета** | |
| Вкладка предназначена для настройки цветовой схемы подсветки  синтаксиса исходного кода. | |

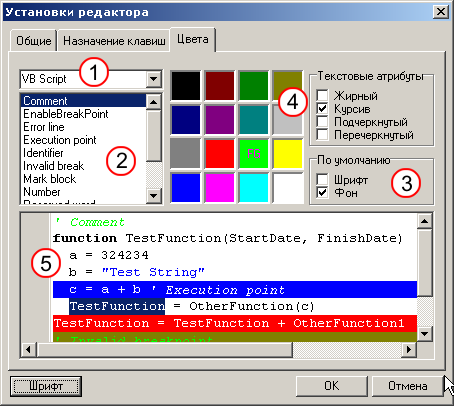
Гедымин поставляется с двумя раскладками клавиатуры: Default (следует соглашениям, принятым в Microsoft Visual Studio) и Classis (соответствует раскладке из ранних версий продуктов компании Borland). На Рис. 83 показано окно, где пользователь может самостоятельно назначить клавиши вызова командам редактора.



*Рис. 83. Окно настройки горячих клавиш редактора*

Настройка цветовой схемы подсветки синтаксиса происходит на вкладке **Цвета,** как показано на Рис. 84. Из выпадающего списка (1) выбирается тип исходного кода (VBScript или SQL). Для каждого структурного элемента из списка (2) можно в области (4) установить цвет текста (левая кнопка мыши) и/или цвет фона (правая кнопка), а также задать параметры начертания шрифта. С помощью флажков (3) можно задать использование шрифта и/или цвета фона по умолчанию. И, наконец, в нижней половине вкладки

(5) приводится пример того, как будет выглядеть код программы в окне редактора.



*Рис. 84. Настройка цветовой схемы подсветки синтаксиса*

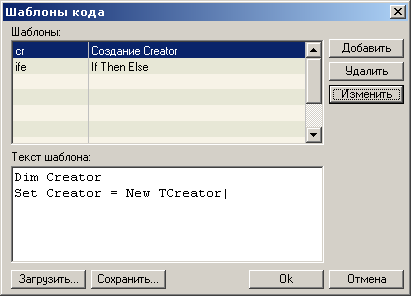
###### Шаблоны кода

Шаблоны кода – это удобный инструмент, который позволяет избежать повторного ввода повторяющихся конструкций. Например, программисту часто приходится создавать экземпляр класса TCreator. Типовой код выглядит следующим образом:

Dim Creator

Set Creator = New TCreator

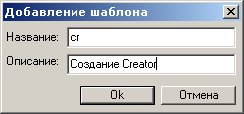
Автоматизируем эту задачу. Откроем окно Шаблоны кода (см. Рис. 85) и выберем кнопку Добавить.



*Рис. 85. Шаблоны кода*

В появившемся окне (см. Рис. 86), в поле **Название** введем два символа --

**cr**, а в поле **Описание** – Создание Creator. Нажмем кнопку Ок.



*Рис. 86. Создание нового шаблона*

В поле Текст шаблона введем следующий текст:

Dim Creator

Set Creator = New TCreator|

Обратите внимание на символ вертикальной черты в конце последней строки. Он показывает, где будет находиться курсор после вставки шаблона в код программы.

Закрываем окно кнопкой Ок.

Чтобы вставить в код программы созданный нами шаблон необходимо в нужном месте набрать последовательность символов **cr** и нажать комбинацию клавиш **Ctrl+J**.

**Пункт меню Окна**

|  |  |  |
| --- | --- | --- |
| **Команда** | **Быстрый**  **вызов** | **Описание** |
| Проводник | Ctrl+Alt+T | Открывает окно Проводника. |
| Время выполнения | Ctrl+Alt+R | Открывает окно регистрации  времени выполнения скрипт- |

|  |  |  |
| --- | --- | --- |
|  |  | функций. |
| Инспектор классов | Ctrl+Alt+I | Открывает окно Инспектора  классов. |
| Сообщения | Ctrl+Alt+M | Открывает окно Сообщений. |
| Стек вызовов | Ctrl+Alt+S | Открывает окно стека вызовов  скрипт-функций. |
| Список переменных | Ctrl+Alt+V | Открывает окно просмотра  значений переменных. |
| Точки остановки | Ctrl+Alt+B | Открывает окно со списком точек  останова. |

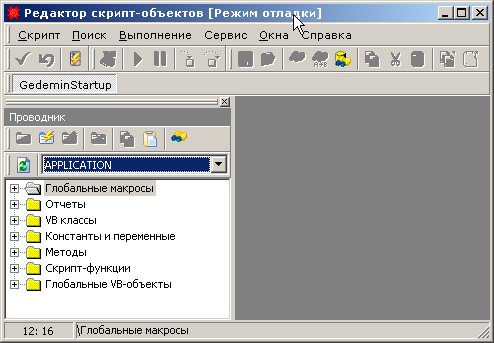
**Пункт меню Справка**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | | **Команда** | **Быстрый**  **вызов** | **Описание** |
|  | | Справка по  VBScript |  |  |
|  | | Справка по  FastReport |  |  |
|  | | Руководство  программиста |  |  |
|  |  | Справка |  |  |

#### Проводник

Роль, аналогичную Исследователю системы, в Редакторе скрипт- объектов выполняет Проводник. Если он скрыт, воспользуемся комбинацией клавиш Ctrl-Alt-T или соответствующей командой из пункта меню Окна35. Как и Исследователь, Проводник любит располагаться в левой части экрана (см. Рис. 87).

35 Не путайте с одноименным пунктом меню главного окна программы!



*Рис. 87. Окно проводника в редакторе скрипт-объектов*

Команды для работы с объектами Проводника доступны на панели инструментов или через контекстное меню.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | | **Команда** | **Быстрый**  **вызов** | **Описание** | |
|  | | Добавить  папку |  | Команда доступна только для  макросов и отчетов. | |
|  | | Добавить  объект |  | Создает новый объект. | |
|  | | Редактировать | Enter или  двойной щелчок | Открывает выбранный объект на редактирование. | |
|  |  | Удалить | Ctrl-Del | Удаляет выбранный объект. | |
|  | | Переименовать | F2 | Переименовать выделенный объект. Команда доступна только для вложенных папок в разделах  Макросы и Отчеты. | |
|  | | Копировать в буфер | Ctrl-C |  | Копировать выбранный объект в буфер обмена. С помощью данной команды можно переносить объекты  между модулями системы. |
|  | | Вставить из  буфера | Ctrl-V | Вставляет объект из буфера обмена. | |
|  | | Поиск | Ctrl-F | Поиск объекта среди элементов  дерева. | |



|  |  |  |  |
| --- | --- | --- | --- |
|  | Обновить | F5 | Перечитывает дерево объектов из  базы данных. |
|  | Информация о классе |  | Команда доступна только для классов из раздела Методы. Выводит на экран  информационное диалоговое окно. |
|  | Отключить |  | Команда доступна только в том случае, если текущим объектом является обработчик события.  Позволяет отключить его. |
|  | Добавить в настройку |  | Открывает диалоговое окно  добавления выбранного объекта в настройку. |

В дереве проводника отображается содержимое текущего модуля, выбор которого осуществляется из выпадающего списка на панели инструментов. Существует три типа модулей: глобальный модуль APPLICATION, модуль формы бизнес-объекта и модуль простой формы. В зависимости от типа дерево Проводника будет содержать те или иные корневые элементы. Ниже приводится их полный список.

|  |  |  |
| --- | --- | --- |
| **Наименование** | **Модуль** | **Описание** |
| Глобальные макросы | APPLICATION | Макросы, доступные из меню на  любой форме проекта. Данная ветвь может содержать вложенные папки. |
| Отчеты | APPLICATION | Все отчеты в базе данных. |
| VB классы | Все | Список VB классов модуля. |
| Константы и  переменные | APPLICATION | Глобальные константы и переменные. |
| Методы | APPLICATION | Иерархические списки классов, наследников TgdcBase – бизнес-классы, и  TgdcCreateableForm – экранные формы |
| Скрипт-  функции | Все |  |
| Глобальные VB  объекты | APPLICATION |  |
| Локальные макросы | Любая форма | Макросы, доступные из меню на  данной форме. Ветвь может содержать вложенные папки. |
| Отчеты <Имя | Форма бизнес- |  |

|  |  |  |
| --- | --- | --- |
| бизнес-класса> | объекта |  |
| Отчеты формы | Любая форма |  |
| События | Любая форма |  |

# Глава VI. Язык VBScript

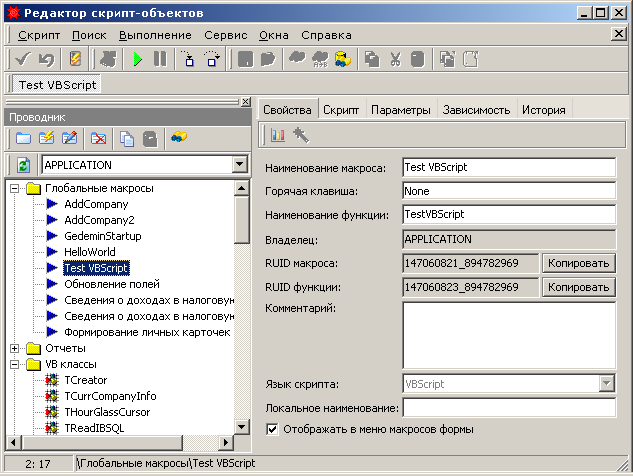
Наверное нет языка программирования проще, чем VBScript – Visual Basic Scripting Edition. Даже начинающий программист способен освоить его за два-три дня, а человек знакомый с Visual Basic или Visual Basic for Applications (VBA) практически сразу может приступать к работе. Ниже приведены основные сведения по синтаксису языка, дано описание операторов и встроенных функций. К сожалению, обработка ошибок в VBScript реализована на очень примитивном уровне. Разработчики не предусмотрели конструкций для обработки исключительных ситуаций, таких как try-except и try-finally в Delphi. В разделах "!!!" и "!!!" мы рассмотрим способы преодоления этих ограничений.

VBScript является интерпретируемым языком. За выполнение программы отвечает Windows Script Host, который поставляется в составе операционной системы начиная с версии Windows 98. На момент написания данной книги, последняя версия VBScript имела номер 5.7 и поставлялась в составе Windows Vista. Обновления для предыдущих версий операционной системы можно скачать с сайта компании Microsoft.

#### Подготовительные мероприятия

Учебные примеры мы будем выполнять в Гедымине, для чего создадим глобальный макрос согласно приведенным ниже инструкциям:

1. Запустим Гедымин;
2. Подключимся к демонстрационной базе данных под учетной записью Administrator;
3. Откроем окно Редактора скрипт-объектов;
4. Создадим новый глобальный макрос под названием "Test VBScript";
5. В поле Наименование функции введем TestVBScript.



*Рис. 88. Создание глобального макроса*

## Основы VBScript

#### Типы данных

В языке VBScript используется единственный тип данных – **Variant** (Вариант), который позволяет хранить в переменной число, строку, дату, булевское значение, ссылку на объект и другую информацию. Определить тип содержимого переменной можно с помощью набора функций: VarType, TypeName, IsArray, IsDate, IsEmpty, IsNull, IsNumeric, IsObject, которые будут рассмотрены ниже. Тип содержащейся информации еще называется подтипом варианта. Полный список подтипов приведен в следующей таблице:

|  |  |
| --- | --- |
| **Подтип** | **Описание** |
| Empty | Переменной не присвоено значение. При использовании неинициализированной переменной в числовых выражениях, будет подставляться 0, а в строковых – пустая  строка. |
| Null | Переменная не содержит данных. |
| Boolean | Булевская переменная может принимать значения **True**,  или **False**. |
| Byte | Целое число в диапазоне от 0 до 255. |
| Integer | Целое число в диапазоне от -32 768 до 32 767. |

|  |  |
| --- | --- |
| Currency | Число с фиксированной точкой в диапазоне от  -922 337 203 685 477.5808 до 922 337 203 685 477.5807. |
| Long | Целое число в диапазоне от -2 147 483 648 до 2 147 483 647. |
| Single | Число с плавающей точкой одинарной точности. Для отрицательных значений допустимый диапазон  от -3.402823E38 до -1.401298E-45. Для положительных – от  1.401298E-45 до 3.402823E38. |
| Double | Число с плавающей точкой двойной точности. Для отрицательных значений допустимый диапазон от  -79769313486232E308 до -4.94065645841247E-324. Для  положительных -- от 4.94065645841247E-324 до 1.79769313486232E308. |
| Date (Time) | Содержит число, представляющее дату в диапазоне от 1-го  января 100 года, до 31 декабря 9999 года. |
| String | Последовательность символов. Максимальная длина в  районе 2-х миллиардов знаков. |
| Object | Объект. |
| Error | Номер ошибки. |

В зависимости от выражения, в котором участвует переменная, ее содержимое будет автоматически приведено к нужному типу. Рассмотрим такой пример:

Option Explicit Sub TestVBScript

Dim A, B A = 5

B = "12"

Application.MessageBox A + B, "", vbOkOnly

End Sub

Так как в выражении участвует числовая переменная **A**, интерпретатор преобразует значение переменной **B** из строки "12" в число и просуммирует их:



*Рис. 89. Результат сложения*

Изменим макрос так, чтобы переменная **А** тоже содержала строку:

Option Explicit Sub TestVBScript

Dim A, B A = "5"

B = "12"

Application.MessageBox A + B, "", vbOkOnly

End Sub

Запустим его на выполнение. Теперь на экране появится результат слияния (конкатенции) двух строк, а не сумма их числовых представлений:



*Рис. 90. Результат слияния строк*

Во избежании путаницы с автоматическим приведением типов, рекомендуется использовать функции конверсии: CBool, CByte, CCur, CDate, CDbl, CInt, CLng, CSng, CStr.

Если результатом выражения должно быть именно слияние строк, а не сумма их числовых представлений, то следует использовать оператор & вместо +.

#### Переменные

Переменная – это удобное символьное обозначение области памяти, где приложение хранит некоторые данные. В процессе выполнения

приложения значение переменной может изменяться. Перед использованием переменную следует объявить с помощью оператора **Dim**.

Dim A

С помощью одного оператора можно объявить сразу несколько переменных, если перечислить их имена через запятую:

Dim Left, Right, Top, Bottom

При объявлении нет необходимости в указании типа данных, так как все переменные имеют тип **Variant**.

Если в первой строке текста скрипта не указано **Option Explicit**, то использовать переменные можно без объявления. Но, такой путь может привести к трудно выявимым ошибкам. Достаточно один раз ошибиться в написании имени переменной в тексте программы, чтобы получить непредсказуемый результат. Мы рекомендуем всегда указывать **Option Explicit** и объявлять переменные.

Имя переменной должно соответствовать следующим требованиям:

1. Начинаться с символа латинского алфавита;
2. Состоять только из символов латинского алфавита или из символов латинского алфавита и цифр;
3. Не превышать 255 символов в длину;
4. Быть уникальным в пределах своей области видимости.

##### Область видимости и время жизни

Область видимости переменной определяется тем, где она была объявлена. Если внутри тела процедуры36, то такая переменная называется локальной и доступна только в пределах этой процедуры. Если переменная объявлена в тексте скрипта, то она будет видима для всех процедур или функций определенных в этом скрипте. Локальные переменные могут иметь одинакове имена, если объявлены в разных процедурах.

В дереве **Проводника** окна **Редактора скрипт-объектов** присутствует специальный раздел – **Константы и переменные** – для объявления глобальных переменных, видимых для всех скрипт-функций проекта.

36 Очевидно, что сказанное в раной мере применимо и к функциям.

Интерпретатор выделяет память для локальных переменных в момент их объявления и высвобождает по выходу из процедуры. Глобальные переменные существуют с момента их объявления и пока скрипт не закончит свое выполнение. Применительно к Гедымину это означает, что глобальные переменные существуют на протяжении всего времени выполнения программы.

##### Присваивание значения переменной

Значение объявленной переменной присваивается с помощью оператора

=. Имя переменной указывается слева от оператора, новое значение – справа. Например:

A = 200

B = "Наименование"

##### Скалярные переменные и массивы

Переменная содержащая единственное значение называется скалярной. Иногда, возникает необходимость хранить несколько значений в одной переменной. В этом случае следует объявить массив. Синтаксис объявления идентичен объявлению скалярной переменной за тем исключением, что после имени в круглых скобках мы задаем размерность массива. Следующее объявление создаст массив из 12 элементов:

Dim Monthes(11)

В языке VBScript левая граница индекса массива всегда 0. Таким образом размер массива вычисляется, как число указанное в скобках плюс один. При присваивании значения элементу массива следует указать его индекс в круглых скобках:

Monthes(0) = "Январь" Monthes(1) = "Февраль" Monthes(2) = "Март"

...

Monthes(10) = "Ноябрь" Monthes(11) = "Декабрь"

Аналогично, при обращении к значению элемента мы используем его индекс:

MonthName = Monthes(5)

Массив не обязательно должен быть одномерным. VBScript позволяет нам задать до 60 размерностей при объявлении массива. Например, следующий оператор создаст двумерный массив из 12 строк и двух колонок37:

Dim MonthDays(11, 1)

При обращении к элементам многомерного массива следует указывать все индексы:

MonthDays(0, 0) = "Январь" MonthDays(0, 1) = 31 MonthDays(1, 0) = "Февраль" MonthDays(1, 1) = 28

...

Выше мы объявляли массивы, размер которых не меняется в процессе работы программы. Если заранее не известно сколько элементов понадобится, то можно объявить динамический массив:

Dim A()

Перед использованием следует установить размер динамического массива с помощью оператора ReDim:

ReDim A(25)

В процессе выполнения можно вызывать оператор ReDim многократно, каждый раз изменяя размер массива. Опция Preserve сохраняет значения элементов массива при изменении размера. Например, следующий код увеличит объявленный выше массив на пять элементов, оставив существующие нетронутыми:

ReDim Preserve A(30)

Помните, что при уменьшении размера массива, значения удаленных элементов будут безвозвратно утеряны.

С помощью оператора Erase можно очистить элементы фиксированного массива или освободить память, занимаемую динамическим массивом.

Dim A

37 При объявлении двумерного массива первое число всегда задает количество строк, а второе – колонок.

ReDim A(25)

...

Erase A

#### Константы

Правилом хорошего тона является объявление констант для многократно используемых в тексте программы значений. Грамотно присвоенное имя константы улучшает читабельность, а само использование

-- упрощает процесс внесения изменений в код. В отличие от переменных, значение константы нельзя изменить в процессе выполнения программы. Создание константы происходит с помощью оператора **Const**:

Const CountryName = "Belarus" Const CountryCode = 375

Несколько констант могут быть объявлены в рамках одного оператора, через запятую. Как и переменная, константа обладает своей областью видимости в зависимости от того, где (в процедуре или за ее пределами) и как (Public или Private) она была объявлена. Константы, созданные оператором Const без указания Public или Private являются общедоступными по-умолчанию.

В дереве **Проводника** окна **Редактора скрипт-объектов** присутствует специальный раздел – **Константы и переменные** – для объявления глобальных констант, видимых для всех скрипт-функций проекта.

Значения строковых констант заключаются в двойные кавычки. Значения типа Дата следует обрамлять символами решетки (#) и использовать американский формат: месяц/день/год. Например:

Const Public IndependenceDay = #03/25/1918#

Во избежание путаницы между константами и переменными рекомендуется использовать единый префикс для всех констант, например "con", или набирать имя константы в верхнем регистре.

Для облегчения труда программиста VBScript содержит набор предопределенных констант. Приведем их ниже.

**Константы для обозначения цветов**

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Описание** |
| vbBlack | &h00 | Черный. |
| vbRed | &hFF | Красный. |

|  |  |  |
| --- | --- | --- |
| vbGreen | &hFF00 | Зеленый. |
| vbYellow | &hFFFF | Желтый. |
| vbBlue | &hFF0000 | Синий. |
| vbMagenta | &hFF00FF | Малиновый. |
| vbCyan | &hFFFF00 | Голубой. |
| vbWhite | &hFFFFFF | Белый. |

**Константы для функций сравнения**

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Описание** |
| vbBinaryCompare | 0 | Двоичное сравнение. |
| vbTextCompare | 1 | Текстовое сравнение. |

**Константы для работы с датой и временем**

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Описание** |
| vbSunday | 1 | Воскресение. |
| vbMonday | 2 | Понедельник. |
| vbTuesday | 3 | Вторник. |
| vbWednesday | 4 | Среда. |
| vbThursday | 5 | Четверг. |
| vbFriday | 6 | Пятница. |
| vbSaturday | 7 | Суббота. |
| vbUseSystem | 0 | Использовать формат даты и времени, заданный в настройках  операционной системы. |
| vbUseSystemDayOfWeek | 0 | Использовать день недели, заданный  в настройках операционной системы как первый день недели. |
| vbFirstJan1 | 1 | Использовать неделю, на которую  приходится 1-е января. |
| vbFirstFourDays | 2 | Использовать неделю хотябы четыре  дня из которой приходятмся на новый год. |
| vbFirstFullWeek | 3 | Использовать первую полную  неделю в году. |

**Константы форматирования даты**

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Описание** |
| vbGeneralDate | 0 | Отображать дату и/или время. Для  действительных чисел отображать дату и |

|  |  |  |
| --- | --- | --- |
|  |  | время. Если присутствует только целая часть числа, отображать только дату. Если только дробная – время. Формат отображения определяется настройками операционной  системы. |
| vbLongDate | 1 | Отображать дату используя полный формат  для даты из настроек операционной системы. |
| vbShortDate | 2 | Отображать дату используя короткий  формат для даты из настроек операционной системы. |
| vbLongTime | 3 | Отображать время используя полный формат для времени из настроек  операционной системы. |
| vbShortTime | 4 | Отображать время используя короткий  формат для времени из настроек операционной системы. |

**Константы для окна Application.MessageBox**

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Описание** |
| vbOkOnly | 0 | Показывать только кнопку Ок. |
| vbOkCancel | 1 | Показывать кнопки Ок и Отмена. |
| vbAbortRetryIgnore | 2 | Показывать кнопки Прервать,  Повторить, Игнорировать. |
| vbYesNoCancel | 3 | Показывать кнопки Да, Нет, Отмена. |
| vbYesNo | 4 | Показывать кнопки Да, Нет. |
| vbRetryCancel | 5 | Показывать кнопки Повторить, Отмена. |
| vbCritical | 16 | Показывать пиктограммку критической  ошибки. |
| vbQuestion | 32 | Показывать пиктограммку вопроса. |
| vbExclamation | 48 | Показывать пиктограммку  восклицания. |
| vbInformation | 64 | Показывать информационную  пиктограммку. |
| vbDefaultButton1 | 0 | Первая кнопка в окне является  выделенной по-умолчанию. |
| vbDefaultButton2 | 1 | Вторая кнопка в окне является  выделенной по-умолчанию. |
| vbDefaultButton3 | 2 | Третья кнопка в окне является  выделенной по-умолчанию. |

|  |  |  |
| --- | --- | --- |
| vbDefaultButton4 | 3 | Четвертая кнопка в окне является  выделенной по-умолчанию. |
| vbApplicationModal | 0 | Пользователь должен закрыть окно перед тем как продолжить работу с  приложением. |
| vbSystemModal | 4096 | Окно будет расположено поверх всех выполняющихся приложений.  Пользователь должен закрыть окно  перед тем как продолжить работу. |

Следующие константы возвращаются функцией:

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Описание** |
| vbOk | 1 | Была выбрана кнопка Ок. |
| vbCancel | 2 | Была выбрана кнопка Отмена. |
| vbAbort | 3 | Была выбрана кнопка Прервать. |
| vbRetry | 4 | Была выбрана кнопка Повторить. |
| vbIgnore | 5 | Была выбрана кнопка Игнорировать. |
| vbYes | 6 | Была выбрана кнопка Да. |
| vbNo | 7 | Была выбрана кнопка Нет. |

**Строковые константы**

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Описание** |
| vbCr | Chr(13) | Возврат каретки. |
| vbCrLf | Chr(13)&Chr(10) | Возврат каретки, перевод строки. |
| vbFormFeed | Chr(12) | Пропуск страницы (не применяется в  Microsoft Windows). |
| vbLf | Chr(10) | Перевод строки. |
| vbNewLine | Chr(13)&Chr(10)  или Chr(10) | Переход на начало новой строки. В  зависимости от платформы. |
| vbNullChar | Chr(0) | Символ с кодом 0. |
| vbNullString |  | Пустая строка. Не тоже самое, что строка  нулевой длины "". |
| vbTab | Chr(9) | Символ табуляции. |
| vbVerticalTab | Chr(11) | Вертикальная табуляция (не  используется в Microsoft Windows). |

**Константы тройного состояния (tristate constants)**

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Описание** |
| vbUseDefault | -2 | Использовать значение по-умолчанию из  настроек операционной системы. |

|  |  |  |
| --- | --- | --- |
| vbTrue | -1 | Истина. |
| vbFalse | 0 | Ложь. |

**Подтипы варианта**

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Описание** |
| vbEmpty | 0 | Неинициализированная переменная. |
| vbNull | 1 | Не содержит данных. |
| vbInteger | 2 | Короткое целое число. |
| vbLong | 3 | Длинное целоей число. |
| vbSingle | 4 | Число с плавающей точкой одинарной  точности. |
| vbDouble | 5 | Число с плавающей точкой двойной  точности. |
| vbCurrency | 6 | Тип данных Currency. |
| vbDate | 7 | Дата. |
| vbString | 8 | Строка. |
| vbObject | 9 | Объект. |
| vbError | 10 | Ошибка. |
| vbBoolean | 11 | Булевское значение. |
| vbVariant | 12 | Вариант. |
| vbDataObject | 13 | Объект доступа к данным. |
| vbDecimal | 14 | Число с фиксированной точкой. |
| vbByte | 17 | Однобайтовое беззнаковое целое. |
| vbArray | 8192 | Массив. |

**Прочие константы**

|  |  |  |
| --- | --- | --- |
| **Имя** | **Значение** | **Описание** |
| vbObjectError | -2147221504 | Пользовательские номера ошибок в объектах автоматизации (Automation Object) должны быть больше чем данная константа.  Например:  Err.Raise vbObjectError + 1000 |

#### Операторы

Операторы VBScript подразделяются на пять категорий: арифметические, сравнения, слияния, логические и присваивания.

##### Арифметические операторы

|  |  |  |
| --- | --- | --- |
| **Оператор** | **Пример** | **Описание** |

|  |  |  |
| --- | --- | --- |
|  | **использования** |  |
| ^ | number ^ exponent | Возводит number в степень exponent. Number может быть меньше нуля только в случае целочисленной степени. Если один из операндов Null, все выражение принимает значение Null. Если несколько возведений в степень выполняются подряд, результат  вычисляется слева направо. |
| \* | number1 \* number2 | Произведение двух чисел. Если операнд имеет значение Empty, то он принимается равным нулю. Если хотя бы один из операндов Null, все выражение  принимает значение Null. |
| / | number1 / number2 | Вещественное деление двух чисел. Для операндов действуют правила  аналогично оператору умножения. |
| \ | number1 \ number2 | Целочисленное деление. Перед вычислением оба операнда приводятся к типу Byte, Integer или Long. В остальном действуют правила как для оператора  деления. |
| Mod | number1 Mod number2 | Остаток от целочисленного деления. Приведение операндов к целому, а также правила обращения с Empty и Null, как у  целочисленного деления. |
| + | expression1 + expression2 | Если оба операнда числа, результатом является их арифметическая сумма. Если оба операнда строки – слияние (конкатенция) двух строк. Если один операнд число, а другой строка, то строковый операнд будет преобразован в число и прибавлен к числовому. Если хотя бы один из операндов Null, все выражение принимает значение Null.  Если оба операнда Empty, результат имеет целочисленное значение 0. Если только один оператор Empty, в качестве результата возвращается значение  второго операнда. |

|  |  |  |
| --- | --- | --- |
| - | number1 – number2 или  - number | В первом случае возвращает разность двух чисел. Во втором – инвертирует знак числа. Правила для операндов со значениями Null и Empty, как для  оператора умножения. |

**Операторы сравнения**

Формат использования операторов сравнения:

result = expression1 comparisonoperator expression2

где используются следующие операторы сравнения: < (меньше), <= (меньше или равно), > (больше), >= (больше или равно), = (равно), <> (не равно).

В зависимости от типов и значений операндов, сравнение осуществляется следующим образом:

|  |  |
| --- | --- |
| **Если** | **То** |
| Оба операнда числа. | Выполняется сравнение двух чисел. |
| Оба операнда строки. | Выполняется сравнение двух строк. |
| Один из операндов число, а второй строка. | Строковый операнд приводится к числу и выполняется сравнение двух  чисел. |
| Один из операндов Empty, а второй  число. | Операнд со значением Empty  принимается равным 0. |
| Один из операндов Empty, а второй строка. | Операнд со значением Empty принимается равным пустой строке "". Осуществляется сравнение двух  строк. |
| Оба операнда Empty. | Операнды считаются равными. |
| Хотя бы один из операндов Null. | Результат принимает значение Null. |

Специальный оператор Is применяется для сравнения двух объектных переменных и возвращает Истину, если обе переменных ссылаются на один и тотже экземпляр объекта.

##### Операторы конкатенции

В данной категории находятся два оператора: + и &. Первый подробно описан в разделе "Арифметические операторы" выше. Рассмотрим использование оператора &.

result = expression1 & expression2

Если операнд не является строкой, он приводится к строковому типу. Если оба операнда Null, то результат также принимает значение Null, однако, в отличие от остальных операторов, если только один операнд Null, то он принимается равным пустой строке. Операнд, имеющий значение Empty, также воспринимается как пустая строка "".

##### Логические операторы

VBScript предоставляет нам следующие логические операторы:

1. Логическое отрицание, инверсия (Not);
2. Логическое умножение, коньюнкция (And);
3. Логическое сложение, дизьюнкция (Or);
4. Логическое исключение (Xor);
5. Логический эквивалент (Eqv);
6. Логическая импликация (Imp).

В качестве операндов логических операторов могут выступать булевские выражения или числовые значения. В первом случае результатом будет булевская константа, во втором – число. В зависимости от оператора подача на вход одного или двух значений Null может приводить к Null результату. Оператор Not является унарным и возвращает логическое отрицание выражения. Над числовым операндом оператор Not производит побитовую инверсию. Остальные логические операторы являются бинарными. В таблице ниже приведены результаты выполнения каждого из операторов в зависимости от значения операндов Exp1 и Exp2:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Exp1 | Exp2 | And | Or | Xor | Eqv | Imp |
| True | True | True | True | False | True | True |
| True | False | False | True | True | False | False |
| False | True | False | True | True | False | True |
| False | False | False | False | False | True | True |
| True | Null | Null | True | Null | Null | Null |
| False | Null | False | Null | Null | Null | True |
| Null | True | Null | True | Null | Null | True |
| Null | False | False | Null | Null | Null | Null |
| Null | Null | Null | Null | Null | Null | Null |

В жизни чаще всего используются операторы And и Or и гораздо реже – Xor. Нам не приходилось сталкиваться с использованием на практике операторов Eqv и Imp. Если вам тяжело разбираться с приведенной выше таблицей резюмируем действие данных операторов:

 And принимает значение Истина только если оба операнда Истина.

В любом другом случае – это либо Ложь, либо Null.

 Or принимает значение Истина, если хотябы один из операндов Истина.

 Xor принимает значение Истина, если значения операндов различаются и Ложь, если они одинаковы.

При побитовом выполнении над числовыми операндами результат логического оператора определяется по следующей таблице:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Exp1 | Exp2 | And | Or | Xor | Eqv | Imp |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 |

##### Оператор присваивания

Оператор присваивания (=) подробно описан в разделе "Переменные".

##### Очередность применения операторов

Если выражение содержит несколько операторов, то они применяются в соответствии с установленным порядком, который называется приоритетом операторов. Изменить порядок по-умолчанию можно с помощью круглых скобок. Выражение внутри скобок всегда вычисляется в первую очередь.

В выражении, содержащем операторы разных категорий, арифметические действия выполняются в первую очередь, затем выполняются операторы сравнения и, в последнюю очередь, -- логические операторы. Все операторы сравнения имеют одинаковый приоритет и вычисляются слева-направо. Арифметические и логические операторы вычисляются в следующем порядке:

1. Возведение в степень (^);
2. Смена знака числа, унарный минус (-);
3. Умножение (\*) и деление (/);
4. Целочисленное деление (\);
5. Остаток от целочисленного деления (Mod);
6. Сложение (+) и вычитание (-);
7. Слияние строк (&).

Если умножение и деление встречаются в одном выражении, то операции выполняются в порядке следования слева-направо. Аналогичное правило действует в случае одновременного присутствия операторов сложения и вычитания.

Оператор слияния строк (&) не является арифметическим и по приоритету располагается между арифметическими операторами и операторами сравнения.

Очередность для логических операторов установлена следующая:

1. Логическое отрицание, инверсия (Not);
2. Логическое умножение, коньюнкция (And);
3. Логическое сложение, дизьюнкция (Or);
4. Логическое исключение (Xor);
5. Логический эквивалент (Eqv);
6. Логическая импликация (Imp).

#### Условные выражения

Условные выражения применяются для управления порядком выполнения команд программы и позволяют организовать переходы (ветвления) и повторения команд. Как правило, операторы сравнения используются вместе с условными выражениями.

##### Выражение If..Then..Else

Выражение условного перехода If позволяет выполнить ту или иную группу команд в зависимости от результата логического выражения или значения булевской переменной.

Для выполнения единственной команды при выполнении заданного условия используется однострочный синтаксис выражения:

Dim S

If DatePart("w", Now) = vbMonday Then S = "Понедельник" Application.MessageBox S, "", vbOkOnly

Обратите внимание, что секция Else в этом случае опущена. Для выполнения группы операторов следует заключить их между ключевыми словами Then и End If.

Dim S

If DatePart("w", Now) = vbMonday Then S = "Сегодня понедельник"

Application.MessageBox S, "", vbOkOnly End If

Если при выполнении условия требуется выполнить один код, а при невыполнении – другой, то используется синтаксис выражения с секцией Else:

Dim S

If DatePart("w", Now) = vbMonday Then S = "Сегодня понедельник"

Else

S = "Сегодня не понедельник" End If

Application.MessageBox S, "", vbOkOnly

При необходимости выбора из нескольких альтернатив подойдет синтаксис с конструкцией ElseIf:

Dim S, D

D = DatePart("w", Now) If D = vbMonday Then

S = "Понедельник" ElseIf D = vbTuesday Then

S = "Вторник"

ElseIf D = vbWednesday Then S = "Среда"

...

End If

Application.MessageBox S, "", vbOkOnly

Выражения If могут быть вложенными:

Dim S, D

D = DatePart("w", Now) If D = vbMonday Then

S = "Понедельник" Else

If D = vbTuesday Then S = "Вторник"

Else

If D = vbWednesday Then S = "Среда"

Else

...

End If End If

End If

Хотя количество секций ElseIf в условном выражении не ограничено, интенсивное их использование может привести к запутанному, неудобочитаемому коду. В случае выбора одной альтернативы из множества возможных в зависимости от значения некоторого селектора рекомендуется использовать выражение Select Case.

##### Выражение Select..Case

Перепишем пример с днями недели с использованием выражения выбора:

Dim S

Select Case DatePart("w", Now) Case vbMonday

S = "Понедельник" Case vbTuesday

S = "Вторник" Case vbWednesday

S = "Среда"

...

Case Else

Err.Raise 32000, "", "Неизвестный день недели" End Select

Так как выражение селектора вычисляется только один раз, использование Select..Case приводит к более эффективному коду.

Рекомендуется всегда использовать секцию Case Else для отлавливания некорректных или необработанных значений селектора.

#### Операторы цикла

Довольно часто возникает ситуация, когда код требуется запустить повторно. Для этого следует написать оператор цикла, который повторяет определенные команды снова и снова. Операторы цикла используются во многих ситуациях: при вычислении итоговой суммы по списку чисел, перемещении по записям набора данных или для запуска блока кода для нескольких объектов. Существует несколько циклов, описанных в следующих разделах. Некоторые из них выполняются, пока условие имеет значение Истина, некоторые – пока Ложь. И, наконец, есть такие, которые выполняются заданное число раз.

##### Оператор Do..Loop

Данный оператор предназначен для выполнения группы команд пока заданное условие Истинно или до тех пор, когда оно не станет Истинным. Проверка условия может осуществляться как в начале цикла:

Do [{While | Until} condition] [statements]

[Exit Do] [statements]

Loop

так и в конце:

Do

[statements] [Exit Do] [statements]

Loop [{While | Until} condition]

Команда Exit Do может встречаться неограниченное число раз в теле цикла. Обычно она используется вместе с условным выражением If..Then и позволяет передать управление на оператор, следующий непосредственно за циклом. При использовании Exit Do внутри вложенного цикла, управление перейдет во внешний цикл.

Следующий код позволяет заменить игральный кубик:

Dim Resp, Num Loop

Num = Int(6 \* Rnd + 1)

Resp = Application.MessageBox Num & " Еще число?", "",\_ vbYesNo or vbQuestion

Do Until Resp = vbNo

##### While..Wend

Представляет собой усеченную версию оператора Do..Loop и позволяет выполнять группу команд пока условие Истинно. Синтаксис оператора:

While condition [statements]

Wend

Обратите внимание, что Exit Do не действует внутри данного цикла.

Циклы While..Wend могут быть вложенными.

##### For..Next

Данный цикл повторяет заданный набор команд указанное число раз.

Синтаксис оператора имеет вид:

For counter = start To end [Step step]

[statements] [Exit For] [statements]

Next

Перед стартом цикла переменной counter присваивается значение start. Далее проверяется выполнение условия counter <= end, при step >= 0, или counter >= end, при отрицательном шаге. После выполнение блока команд переменная counter увеличивается на значение step и все повторяется сначала.

Изменение счетчика в теле цикла не запрещено, но настоятельно не рекомендуется, так как затрудняет понимание логики программы и ее отладку.

Exit For может встречаться в теле цикла произвольное число раз. Циклы могут быть вложенными. Например, такой цикл инициализирует трехмерный массив:

Dim A(9, 9, 9) Dim I, J, K

For I = 0 To 9

For J = 0 To 9

For K = 0 To 9 A(I, J, K) = 1

Next Next

Next

##### For Each..Next

Оператор цикла For Each..Next повторяет заданный набор команд для каждого элемента массива или коллекции и имеет следующий синтаксис:

For Each element In group [statements]

[Exit For] [statements]

Next

Цикл выполняется, если в массиве или коллекции присутствует хотябы один элемент. Exit For может встречаться в теле цикла произвольное количество раз.

Проиллюстрируем использование For Each..Next на примере следующего кода, который выводит на экран список файлов из корневого каталога диска с:\

Dim fso, f, f1, fc, s

Set fso = CreateObject("Scripting.FileSystemObject") Set f = fso.GetFolder("c:\")

Set fc = f.Files For Each f1 in fc

s = s & f1.name & vbNewLine Next

Application.MessageBox s, "Файлы на c:\", vbOkOnly

#### Процедуры

Для экономии памяти и структурирования программы, фрагмент кода, вызываемый многократно в разных местах, может быть оформлен ввиде процедуры. В языке VBScript существуют два вида процедур: подпрограммы (Sub) и функции (Function). Подпрограмма – это последовательность операторов, обрамленная ключевыми словами Sub и End Sub. Подпрограмма может принимать на вход параметры, но не возвращает значения. Функция – последовательность операторов, заключенная между Function и End Function, -- возвращает результат и поэтому может быть использована в выражении. Каждая процедура должна иметь имя, уникальное в пределах модуля. Имена процедур, объявленных в глобальном модуле, должны быть уникальны в рамках всего проекта.

Определение подпрограммы и функции имеет следующий синтаксис:

[Public [Default] | Private] Sub *name* [(arglist)] [statements]

[Exit Sub] [statements]

End Sub

[Public [Default] | Private] Function *name* [(arglist)] [statements]

[Exit Function] [statements]

End Function

Public процедуры являются глобальными и доступны во всех скриптах программы. Private процедуры доступны только в том скрипте, где они были объявлены. Если не указано иное, объявленная процедура является общедоступной. Ключевое слово Default может быть использовано только в теле класса и служит для указания метода по-умолчанию этого класса.

Список параметров имеет следующий синтаксис:

[ByRef | ByVal] varname [, ...]

Параметры могут передаваться по значению (ByVal) или по ссылке (ByRef). По-умолчанию все параметры передаются по значению. Константы, результаты вычисления выражений могут быть переданы только по значению. Изменение параметра, переданного по ссылке, приведет к изменению значения наружной переменной. Поясним передачу параметров внутрь процедуры на следующем примере:

Sub DoCalculation(ByRef A, ByVal B, ByVal C) A = C \* 2

B = C / 2

End Sub

Sub TestVar Dim V1, V2 V1 = 1

V2 = 2

DoCalculation V1, V2, 10

' После выполнения процедуры DoCalculation ' V1 = 20

' V2 = 2

End Sub

Переменные, объявленные внутри тела процедуры, являются локальными и уничтожаются по завершении ее выполнения. Значения локальных переменных не сохраняются.

Список параметров указывается в круглых скобках при вызове функции или при вызове подпрограммы с помощью оператора Call38. Так, вызов процедуры DoCalculation в приведенном выше примере мы могли написать следующим образом:

Call DoCalculation(V1, V2, 10)

38 Здесь авторы VBScript отличились своей непоследовательностью. Допускается использование круглых скобок без оператора Call при вызове процедуры с единственным параметром.

#### Классы VBScript

VBScript позволяет создавать новые классы, которые в дальнейшем мы будем называть VB классами. Вообще говоря, полноценными классами в понимании объектно-ориентированного программирования они не являются, поскольку не поддерживают наследование и, соответственно, полиморфизм. Так что из трех китов, на которых базируется объектно- ориентированная парадигма остается только инкапсуляция — возможность объединять в рамках одной сущности данные и методы.

Определение класса осуществляется с помощью следующей конструкции:

**Class** *name statements*

**End Class**

где *name* — это имя класса, а *statements* — это одно или несколько определений переменных, свойств, процедур или функций, называемых так же членами класса. Обратите внимание, что в отличие от Delphi, где код определения класса содержит только объявления процедур и функций, в VB классе код членов прописывается прямо в тексте класса.

Члены класса могут быть объявлены как **Private** или **Public**. Первые видны только внутри кода данного класса, вторые же доступны как для внутреннего кода, так и снаружи. Если переменная или функция (процедура) не содержат явного определения **Public** или **Private**, то они считаются общедоступными. Процедуры или функции объявленные как **Public** внутри блока класса становятся методами данного класса. Переменные, объявленные как общедоступные, становятся свойствами класса наравне со свойствами объявленными непосредственно с помощью конструкций **Property Get**, **Property Let**, **Property Set**.

##### Определение свойств класса

Выше мы уже говорили о том, что поля класса явно или не явно объявленные как Public, становятся его свойствами. Кроме этого, создать свойство класса можно определив специальные функции для считывания значения свойства (Property Get), а также для его присвоения (Property Let или Property Set).

Синтаксис определения таких функций следующий:

[**Public** [**Default**] | **Private**] **Property Get** *name* **[(***arglist***)**] [*statements*]

[[**Set**] *name* = *expression*] [**Exit Property**] [*statements*]

[[**Set**] *name* = *expression*]

**End Property**

[**Public** | **Private**] **Property Let** *name* **(**[*arglist*,] *value***)**

[*statements*] [**Exit Property**] [*statements*]

**End Property**

[**Public** | **Private**] **Property Set** *name***(**[*arglist*,] *reference***)**

[*statements*] [**Exit Property**] [*statements*]

**End Property**

Определив только одну функцию, считывания или присвоения, можно создать свойство, соответственно, только для чтения или записи. Процедура Property Let используется для присвоения простых типов данных, а Property Set для передачи ссылки на объект. Обратите внимание, что все три функции могут принимать на вход произвольный список параметров. Таким образом можно организовать, например, свойства-массивы, передавая в качестве аргумента индекс элемента.

##### Создание и уничтожение экземпляра VB класса

Создание экземпляра VB класса осуществляется с помощью оператора

**New**.

Dim X

Set X = New classname

Уничтожение ранее созданного экземпляра происходит автоматически по завершении блока кода, где была объявлена соответсвующая переменная и при условии, что на нее нет внешних ссылок. Если необходимо уничтожить экземпляр вручную, то необходимо присвоить значение Nothing переменной.

‘ объявление переменной и создание экземпляра класса

Dim X

Set X = New classname

...

‘ использование экземпляра класса

...

‘ уничтожение экземпляра класса

Set X = Notning

...

##### События Initialize и Terminate

Событие Initialize происходит при создании экземпляра класса, а Terminate — при его уничтожении. Разработчик может определить свои обработчики данных событий. Ниже приведен пример использования событий создания и удаления объекта:

Class TestClass

' Определение обработчика события Initialize. Private Sub Class\_Initialize

MsgBox("TestClass started")

End Sub

' Определение обработчика события Terminate. Private Sub Class\_Terminate

MsgBox("TestClass terminated") End Sub

End Class

' Создание экземпляра класса TestClass.

' На экран будет выведено сообщение "TestClass started"

Set X = New TestClass

' Уничтожение экземпляра.

' На экран будет выведено сообщение "TestClass terminated" Set X = Nothing

## Встроенные функции

##### Abs

Абсолютное значение, модуль числа.

**Abs(*number*)**

*number* – любое числовое выражение. Функция вернет Null, если на вход будет передан Null. Аргумент со значением Empty воспринимается как 0.

Пример:

Dim N

N = Abs(32.7) ' Присвоит 32.7 N = Abs(-32.7) ' Присвоит 32.7

##### Array

Возвращает Variant содержащий массив.

**Array(*arglist*)**

В качестве аргумента функции передается список значений через запятую, которые будут присвоены элементам массива. Функция вернет массив нулевой длины, если параметры не заданы.

Пример:

Dim A, B

A = Array("январь", "февраль", "март")

B = A(2) ' B теперь содержит строку "март"

Обратите внимание, что переменная, которая не была объявлена как массив, может содержать массив. Хотя, переменная типа вариант, содержащая массив, концептуально отличается переменной типа массив, доступ к элементам в каждом случае осуществляется одинаково.

##### Asc

Возвращает ANSI код первого символа в строке.

**Asc(*string*)**

Строковый параметр не может быть пустой строкой. Пример:

Dim CharCode

CharCode = Asc("A")

' Вернет 65

CharCode = Asc("Belarus") ' Вернет 66

##### Atn

Арктангенс числа.

**Atn(*number*)**

Функция позволяет вычислить величину угла в прямоугольном треугольнике. На вход подается отношение противолежащего катета к прилежащему. Результат возвращается в радианах в диапазоне – pi/2 до pi /

2. Для перевода из радиан в градусы следует умножить величину в радианах на 180/pi.

Функция Atn является инверсной по отношению к функции Tan (тангенс числа), которая по заданному углу вычисляет отношение двух сторон прямоугольного треугольника. Не путайте арктангенс с котангенсом, который является простым обратным значением (1/Tan).

##### CBool

Конвертирует значение в вариант с подтипом **Boolean**.

**CBool(*expression*)**

Если выражение равно 0, то возвращается False. В остальных случаях возвращается True. Если выражение не может быть интерпретировано как числовое значение – возникает ошибка.

Пример:

Dim A

A = CBool(5) A = CBool(0)

' A = True ' A = False

A = CBool("True") ' A = True

##### CByte

Конвертирует переданное значение в вариант с подтипом **Byte**.

**CByte(*expression*)**

Если expression является строковым выражением, то преобразование осуществляется согласно региональным настройкам операционной системы. Значение не должно выходить за рамки диапазона Byte, в противном случае будет сгенерирована ошибка. Дробные числа

округляются, причем, если дробная часть равна 0.5, то округление осуществляется до ближайшего четного числа.

Пример:

Dim MyDouble, MyByte MyDouble = 55.7899

MyByte = CByte(MyDouble) ' MyByte = 56

##### CCur

Конвертирует переданное значение в вариант с подтипом **Currency**.

**CCur(*expression*)**

Если expression является строковым выражением, то конвертация в число осуществляется согласно региональным настройкам операционной системы.

Пример:

Dim MyDouble, MyCurr MyDouble = 123.67894532

MyCurr = CCurr(MyDouble) ' MyCurr = 123.6790

##### CDate

Конвертирует переданное значение в вариант с подтипом **Date**.

**CDate(*date*)**

Переданное значение может быть строкой, числом или литералом даты.39 С помощью функции **IsDate** можно проверить является ли значение корректной датой и/или временем. При конвертации из числа, целая часть обозначает количество дней начиная с 30 декабря 1899 года. Негативные значения соответствуют более ранним датам. Дробная часть конвертируется во время суток, начиная с полуночи.

При конвертации из строки учитываются региональные установки операционной системы. Конвертация невозможна, если строка, кроме даты, будет содержать еще и день недели.

Пример:

39 Литерал даты – это строковое представление даты в формате mm/dd/yyyy, обрамленное символами решетки (#). Например, литерал #12/31/2007# обозначает 31 декабря 2007 года.

Dim MyDate

MyDate = CDate(1.5)

' 31 декабря 1899 года, 12:00

MyDate = CDate("31.12.2007") ' присвоит значение

' 31 декабря 2007 года

Совет: не указывайте даты строками в тексте программы, так как их преобразование может зависеть от региональных установок компьютера. Используйте литералы даты.

##### CDbl

Преобразует переданное значение в вариант с подтипом **Double**.

**CDbl(*expression*)**

При конвертации из строки используется информация о разделителе дробной части и триад из региональных настроек операционной системы.

Пример:

Dim MyDouble

MyDouble = CDbl(5 \* 5) ' MyDouble = 25

##### Chr

Возвращает символ соответствующий заданному ANSI коду.

**Chr(*charcode*)**

Коды в диапазоне от 0 до 31 соответствуют непечатаемым символам в кодировке ASCII. Chr(10) – перевод строки, Chr(13) – возврат каретки и т.д.

Пример:

Dim MyChar

MyChar = Chr(65) ' MyChar = "A"

##### CInt

Преобразует переданное значение в варинт с подтипом **Integer**.

**CInt(*expression*)**

Преобразование строковых значений осуществляется в соответствии с региональными настройками операционной системы. Результирующее число должно попадать в допустимый диапазон для подтипа Integer. Округление дробных чисел производится так, что если дробная часть в точности равна 0.5, то результатом будет ближайшее четной число.

Пример:

Dim MyInt

MyInt = CInt(257.23 \* 77.56) ' MyInt = 19951

##### CLng

Преобразует переданное значение в варинт с подтипом **Long**.

**CLng(*expression*)**

Преобразование строковых значений осуществляется в соответствии с региональными настройками операционной системы. Результирующее число должно попадать в допустимый диапазон для подтипа Long. Округление дробных чисел производится так, что если дробная часть в точности равна 0.5, то результатом будет ближайшее четной число.

Пример:

Dim MyLng

MyLng = CLng(258.5) ' MyInt = 258

##### Cos

Функция вычисляет косинус угла.

**Cos(*angle*)**

Функция Cos возвращает отношение длины прилежащего катета к гипотенузе в прямоугольном треугольнике. Аргумент функции задает величину угла в радианах. Результат находится в диапазоне от -1 до 1.

##### CreateObject

Создает объект автоматизации и возвращает ссылку на него.

**CreateObject(*servername.typename*[*, location*])**

servername – имя приложения, поставляющего объект; typename – имя класса объекта;

location – необязательный параметр. Имя сетевого компьютера, где будет создан объект.

Сервер автоматизации реализует как минимум один класс объектов автоматизации. Например, текстовый процессор может предоставлять доступ к объектам приложения, документу, панели инструментов и т.п. Для создания объекта автоматизации присвойте его значение объявленной переменной:

Dim ExcelSheet

Set ExcelSheet = CreateObject("Excel.Sheet")

Приведенный код вызывает приложение (в данном случае Microsoft Excel) и создает объект типа Лист электронной таблицы (Sheet). После создания объекта мы можем обращаться к свойствам и вызвать его методы. Ниже приведен пример работы с листом электронной таблицы:

' Отображаем Excel на экране ExcelSheet.Application.Visible = True ' Вводим текст в первую клетку на листе

ExcelSheet.ActiveSheet.Cells(1, 1).Value = \_ "Колонка А, строка 1"

' Сохраняем документ ExcelSheet.SaveAs "C:\TEST.XLS" ' Закрываем Excel ExcelSheet.Application.Quit

' Уничтожаем объект

Set ExcelSheet = Nothing

Если позволяют настройки сетевой безопасности, можно создать объект на удаленном компьютере. Для этого в качестве параметра location следует указать имя компьютера или его IP адрес.

Приведенная ниже функция возвращает версию приложения Microsoft Excel, установленного на компьютере MyServer:

Function GetVersion Dim XLApp

Set XLApp = CreateObject("Excel.Application", "MyServer") GetVersion = XLApp.Version

End Function

Обратите внимание, что по завершении функции объект XLApp будет уничтожен автоматически.

##### CSng

Преобразует переданное значение в вариант с подтипом **Single**.

**CSng(*expression*)**

При конвертации из строки используется информация о разделителе дробной части и триад из региональных настроек операционной системы.

Пример:

Dim MySingle

MySingle = CSng(5 \* 5) ' MySingle = 25

##### CStr

Преобразует переданное значение в вариант с подтипом **String**.

**CStr(*expression*)**

Ниже показано, как будет осуществляться конвертация данных различных типов в строку:

Boolean Строки "True" или "False";

Date Строка, содержащая дату в коротком представлении из региональных настроек операционной системы;

Null приведет к возникновению ошибки;

Empty пустая строка "";

Error Строка содержащая слово "Error" и номер ошибки; Число Строковое представление числа.

Пример:

Dim MyDouble, MyString MyDouble = 345.876

MyString = CStr(MyDouble) ' MyString = "345.876"

##### Date

Функция возвращает текущую дату.

##### DateAdd

Прибавляет к дате указанный интервал.

**DateAdd(*interval*, *number*, *date*)**

interval Показывает, какой интервал будет прибавлен к заданной дате. Строковая константа. Список возможных значений приведен ниже.

number Показывает сколько раз будет прибавлен интервал. Допускаются как положительные, так и отрицательные значения. Нецелые значения округляются.

date Исходная дата.

Для указания интервала применяются следующие значения: yyyy год;

q квартал;

m месяц;

y номер дня в году;

d день;

w номер дня в неделе;

ww номер недели в году;

h час;

n минута;

s секунда.

Функция может быть использована для прибавления или вычитания заданного количества временных интервалов из определенной даты. Возвращаемое значение – всегда действительная дата. Так, при вызове функции:

NewDate = DateAdd("m", 1, #12/31/1995#)

Мы получим 28 февраля 1995 года, а не 31 февраля, так как последнее не является корректной датой.

Обратите внимание, что результат обратного преобразования может отличаться от исходной даты. Так, для даты 31 января 1995 года, прибавление и последующее вычитание месяца даст в итоге 28 января 1995 года.

##### DateDiff

Функция возвращает количество временных интервалов между указанными датами.

**DateDiff(*interval*, *date1*, *date2*, [,*firstdayofweek*[,**

***firstweekofyear*]])**

interval Временной интервал. Строковая константа. Список возможных значений приведен выше в описании функции DateAdd.

date1,date2 Даты, между которыми вычисляется промежуток. firstdayofweek Опциональный параметр. Определяет с какого дня

начинается неделя. Если опущен – первым днем считается воскресенье. Возможные значения приведены в разделе Константы.

firstweekofyear Опциональный параметр. Показывает, какая неделя в году считается первой. Если опущен, то таковой будет неделя на которую приходится 1-е января. Возможные значения параметра приведены в разделе Константы.

Данная функция используется для вычисления величины периода между двумя датами. С ее помощью можно получить ответы на такие вопросы, как: сколько дней прошло между двумя датами или сколько недель осталось до конца текущего года.

Для вычисления промежутка в днях можно с равным успехом использовать константы "d" (День) или "y" (День года). Если интервал задан как "w" (День недели), функция вычисляет количество полных недель (семидневных интервалов) между двумя датами. Например, если date1 приходится на понедельник, функция подсчитывает количество понедельников до date2 включительно. При этом, дата date1 в расчет не включается. Если интервал задан как "ww" (Неделя), функция возвращает количество календарных недель, т.е. результат равен числу воскресений в промежутке между date1 и date2. При этом, date2 включается в расчет, а date1 – нет.

Если date1 содержит более позднюю дату, чем date2, результат будет отрицательным значением.

Параметр firstdayofweek определяет как будут трактоваться символы "w" и "ww".

Допускается для параметров date1 и date2 указать только месяц и день. В этом случае будет подставлен текущий год. Например, следующий вызов функции вычисляет количество дней между 28-м февраля и первым марта для текущего года:

Days = DateDiff("d", "02/28", "03/01")

В 2004-м году, високосном, переменная Days получит значение 2, а в 2005 и 2006 – 1.

##### DatePart

Возвращает указанную часть даты.

**DatePart(*interval*, *date*[, *firstdayofweek*[, *firstweekofyear*]])**

interval Строковая константа. Указывает какую часть даты необходимо получить. Список возможных значений приведен в описании функции DateAdd.

date Дата, из которой извлекается значение.

firstdayofweek Опциональный параметр. Определяет с какого дня начинается неделя. Если опущен – первым днем считается воскресенье. Возможные значения приведены в разделе Константы.

firstweekofyear Опциональный параметр. Показывает, какая неделя в году считается первой. Если опущен, то таковой будет неделя на которую приходится 1-е января. Возможные значения приведены в разделе Константы.

Как и в случае с предыдущей функцией, в дате могут быть указаны только месяц и день. В этом случае текущий год будет подставлен автоматически.

Параметры firstdayofweek и firstweekofyear определяют, как будут трактоваться символы "w" и "ww".

Пример:

Dim Q, W

Q = DatePart("q", Now) ' Q = номеру текущего квартала

W = DatePart("ww", Now, vbMonday, vbFirstJan1)

' W = номеру текущей недели в году, причем, первым днем ' недели считается понедельник и первой неделей считается ' неделя, на которую приходится 1-е января

##### DateSerial

Возвращает вариант с подтипом **Date** для указанного года, месяца и дня.

**DateSerial(*year*, *month*, *day*)**

Если значение аргумента выходит за пределы допустимого диапазона в большую или меньшую сторону, то следующий по порядку аргумент увеличивается или уменьшается соответственно. Поясним сказанное на примерах:

Dim MyDate

MyDate = DateSerial(2005, 12, 31) ' 31 декабря 2005 г.

MyDate = DateSerial(2005, 12, 140)' 19 апреля 2006 г.

MyDate = DateSerial(2005, -5, -20)' 10 июня 2006 г.

Значения аргумента year от 0 до 99 включительно интерпретируются, как года в диапазоне 1900-1999. В остальных случаях следует задавать год полностью, четырехзначным числом.

Если значение аргумента выходит за границы диапазона от -32768 до 32767 или результат функции не является корректной датой, генерируется ошибка.

##### DateValue

Конвертирует значение в вариант с подтипом **Date**.

**DateValue(*date*)**

Чаще всего, параметр date – это строковое выражение. Информация о времени не возвращается данной функцией, даже если она присутствует в строке. Однако, если строка содержит некорректное время, будет сгенерирована ошибка.

Строка распознается в соответствии с коротким форматом даты, установленным в региональных настройках операционной системы. Функция достаточно интеллектуальна, чтобы понять, что и "31.12.2005", "12.31.2005" и "2005.12.31" это одна и таже дата – 31-е декабря 2005 года. Кроме этого, DateValue способна получить дату из строки, которая содержит полное или краткое наименование месяца.

Если в строке не указан год, будет автоматически подставлен номер текущего года.

Пример:

Dim MyDate

MyDate = DateValue("10, сентябрь, 2005") ' 10.09.2005

##### Day

Возвращает номер дня в месяце, целое число в диапазоне от 1 до 31.

**Day(*date*)**

##### Eval

Вычисляет выражение и возвращает результат.

**[result = ]Eval(*expression*)**

result Имя переменной, которой присваивается результат. expression Любое корректное выражение на языке VBScript.

В языке VBScript выражение x = y может быть интерпретировано двояко. В первом случае, как оператор присваивания: переменной x будет присвоено значение y. Во втором -- как оператор сравнения, который проверяет равны ли между собой значения x и y. Функция Eval всегда использует вторую интерпретацию (сравнение), в то время, как выражение Execute всегда использует первую (присваивание).

Пример:

Dim A, B A = Rnd

B = Rnd

If Eval("A > B") Then

Application.MessageBox "A больше B", "", vbOkOnly Else

Application.MessageBox "A меньше либо равно B", \_ "", vbOkOnly

End If

##### Exp

Вычисляет указанную степень числа *е*.

**Exp(*number*)**

Максимальное значение степени составляет 709.782712893.

##### Filter

Извлекает из массива строк элементы, удовлетворяющие указанному критерию и помещает их в результирующий массив.

##### Filter(inputstrings, value[, include[, compare]])

inputstrings Массив строк из которого осуществляется выбор. value Искомая подстрока.

include Опциональный параметр. Если равен True, только те строки, которые содержат подстроку value, будут отобраны. Если False —извлекаются строки не содержащие искомую подстроку.

compare Опциональный параметр. Определяет как будет осуществляться поиск: с учетом регистра символов (vbBinaryCompare) или без учета (vbTextCompare).

Результирующий массив может быть пустым, если ни одна строка не отвечает критерию отбора.

Пример:

Dim A

A = Filter(Array("100", "200", "300"), "1") ' A(0) = "100"

##### FormatCurrency

Форматирует значение как денежную строку.

**FormatCurrency(*expression*[, *numdigits*[, *leadingdigit*[, *useparens* [, *groupdigits*]]]])**

expression Число.

numdigits Опциональный параметр. Количество знаков после десятичной точки. -1 означает, что будут применены текущие региональные установки.

leadingdigit Опциональный параметр. Определяет будет ли отображаться ноль слева от десятичной точки для дробных значений меньше единицы. Возможные значения приведены в подразделе "Константы тройного состояния".

useparens Опциональный параметр. Показывает следует ли заключать отрицательные значения в скобки. Возможные значения приведены в подразделе "Константы тройного состояния".

groupdigits Опциональный параметр. Показывает на необходимость группировки цифр в числе. Возможные значения приведены в подразделе "Константы тройного состояния".

Знак валюты и его положение относительно числа определяются региональными настройками операционной системы. Значения по- умолчанию берутся с вкладки Валюта региональных настроек операционной системы. Значение по-умолчанию для параметра leadingdigit берется с вкладки Число.

Пример:

Dim MyCurrency

MyCurrency = FormatCurrency(1000) ' MyCurrency="1 000.00 р."

##### FormatDateTime

Форматирует выражение как дату или время.

**FormatDateTime(*expression*[, *format*])**

Для возможных значений параметра format смотрите подраздел "Константы форматирования даты и времени".

##### FormatNumber

Форматирует число.

**FormatNumber(*expression*[, *numdigits*[, *leadingdigit*[, *useparens* [, *groupdigits*]]]])**

Подробное описание аргументов приведено в описании функции FormatCurrency. Значения по-умолчанию берутся с вкладки Число региональных установок операционной системы.

##### FormatPercent

Форматирует число как проценты, умножает на 100 и добавляет символ %.

**FormatPercent(*expression*[, *numdigits*[, *leadingdigit*[, *useparens* [, *groupdigits*]]]])**

Подробное описание аргументов приведено в описании функции FormatCurrency. Значения по-умолчанию берутся с вкладки Число региональных установок операционной системы.

##### GetLocale

Возвращает идентификатор текущих региональных установок.

##### GetObject

Создает объект автоматизации из дискового файла.

##### GetObject([*pathname*][, appname.objecttype])

pathname Полное имя файла с путем и расширением. Если параметр не указан, appname и objecttype должны быть определены.

appname.objecttype Опциональный параметр. Имя сервера автоматизации и класс объекта.

Если указано только имя файла, создается объект автоматизации в соответствии с его типом. Если для указанного файла возможно создание объектов разных типов, то с помощью appname.objecttype можно конкретно указать, объект какого типа будет создан. Некоторые приложения позволяют открыть часть файла (например, один лист в электронной таблице Microsoft Excel). В этом случае имя части файла следует прибавить к имени файла, используя разделителем символ "!".

Если первый параметр задан пустой строкой (""), то новый экземпляр указанного типа будет создан.

Если указан только тип – appname.objecttype, -- то функция вернет ссылку на активный объект данного типа. Если активных объектов на момент вызова функции нет, будет получено сообщение об ошибке.

##### GetRef

Возвращает ссылку на процедуру.

***Object*.*EventName* = GetRef(*procname*)**

object Объект автоматизации, для которого назначается обработчик события.

eventname Имя события.

procname Имя процедуры или функции, которая присваивается обработчику события.

Пример:

Option Explicit Dim xmlHttp

Function HandleRequest()

MsgBox "Состояние выполнения запроса: " &\_ xmlHttp.readyState

If xmlHttp.ReadyState = 4 Then Set xmlHttp = Nothing End Function

Sub TestVBScript

Set xmlHttp = CreateObject("Microsoft.XMLHTTP") xmlHttp.OnReadyStateChange = GetRef("HandleRequest") xmlHttp.Open "GET", ["http://gsbelarus.com/gs/index.php",\_](http://gsbelarus.com/gs/index.php)

True xmlHttp.Send Null

End Sub

##### Hex

Возвращает строку – представление числа в шестнадцатиричной системе счисления.

**Hex(*number*)**

Действие Hex подобно действию функции Oct.

Обратите внимание, что вы можете использовать шестнадцатиричные значения в тексте программы непосредственно, предворив их префиксом &H.

##### Hour

Извлекает номер часа из времени.

**Hour(*date*)**

Функция возвращает целочисленное значение в диапазоне от 0 до 23 включительно.

##### InputBox

В Гедымине не используется, так как окно не является модальным.

**InStr**

Возвращает позицию искомой последовательности символов в заданной строке.

**InStr([*start*, ] *string1*, *string2*[, *compare*])**

start Позиция, начиная с которой осуществляется поиск. Если аргумент не задан, поиск начинается с первого символа строки.

string1 Строка, в которой идет поиск.

string2 Искомая последовательность символов.

compare Опциональный параметр. Может принимать значения vbVinaryCompare (поиск с учетом регистра) и vbTextCompare (поиск без учета регистра символов).

В зависимости от значений входящий параметров функция вернет следующий результат:

###### Значение параметра Результат

string1 пустая строка 0

string1 Null Null

string2 пустая строка start

string2 Null Null

string2 не найдена 0

string2 найдена Позиция первого символа string2.

start > Len(string1) 0

##### InStrRev

Действие функции и параметры аналогичны InStr с тем исключением, что поиск начинается с конца строки.

##### Int, Fix

Функции возвращают целую часть числа.

**Int(*number*), Fix(*number*)**

Обе функции отбрасывают дробную часть числа. Различие между ними заключается в том, что для отрицательных чисел Int вернет первое целое, меньшее или равное заданному числу, а Fix – большее или равное. Действие Fix эквивалентно выражению Sgn(number) \* Int(Abs(number)).

Пример:

Dim I

I = Int(-8.8) ' I = -9

I = Fix(-8.8) ' I = -8

##### IsArray

Возвращает True, если выражение является массивом.

**IsArray(*expression*)**

##### IsDate

Возвращает True, если выражение является датой или может быть преобразовано в дату.

**IsDate(*expression*)**

##### IsEmpty

Возвращает True, если переменная не была инициализирована.

**IsEmpty(*expression*)**

##### IsNull

Возвращает True, если переменная не содержит данных.

**IsNull(*expression*)**

##### IsNumeric

Возвращает True, если переданное значение является числом.

**IsNumeric(*expression*)**

##### IsObject

Возвращает True, если на вход передан объект автоматизации.

**IsObject(*expression*)**

##### Join

Объединяет значения элементов массива в одну строку.

**Join(*list*[, *delimiter*])**

list Одномерный массив.

delimiter Опциональный параметр. Задает символ разделитель. Если не указан, используется пробел. Может быть пустой строкой.

Пример:

Dim A, S

A = Array(1, 2, 3, 4, 5)

S = Join(A, ";") ' S = "1;2;3;4;5"

##### LBound

Возвращает нижнюю границу индекса массива.

##### LBound(arrayname[, dimension])

Функция используется вместе с UBound для определения размера массива. В VBScript нижняя граница любой размерности массива всегда равна 0.

##### LCase

Приводит символы строки к нижнему регистру.

**LCase(*string*)**

##### Left

Возвращает указанное количество первых символов строки.

**Left(*string*, *length*)**

Ограничения на параметры и применение аналогичны функции Right.

##### Len

Возвращает длину строки. **Len(*string*) LoadPicture**

##### Log

Натуральный логарифм числа.

**Log(*number*)**

Number должен быть больше нуля. Логарифм числа X по основанию 10 вычисляется по следующей формуле:

Log10 = Log(X) / Log(10)

##### LTrim, RTrim, Trim

Возвращают копию строки без начальных (LTrim), концевых (RTrim) или и начальных и концевых (Trim) пробелов.

##### Mid

Извлекает подстроку заданной длины, начиная с указанной позиции из переданной строки.

##### Mid(string, start[, length])

string Строковое выражение. Функция вернет Null, если передан Null.

start Первый символ извлекаемой подстроки. Если start > Len(string) функция вернет пустую строку.

length Длина подстроки. Если параметр опущен, весь остаток строки, начиная с позиции start будет возвращен.

Пример:

Dim S

S = Mid("Republic of Belarus", 10, 2) ' S = "of"

##### Minute

Извлекает номер минуты из времени.

**Minute(*time*)**

Возвращает целое число в диапазоне от 0 до 59.

##### Month

Извлекает номер месяца из даты.

**Month(*date*)**

Возвращает целое число в диапазоне от 1 до 12.

##### MonthName

Возвращает наименование месяца.

**MonthName(*month*[, *abbreviate*])**

month Номер месяца. Целое число в диапазоне от 1 (январь) до 12 (декабрь).

abbreviate Булевский опциональный параметр. Определяет какое наименование вернет функция: полное (False) или краткое (True). По-умолчанию установлен в False.

Пример:

Dim MName

MName = MonthName(2) ' MName = "Февраль"

##### MsgBox

Выводит на экран диалоговое окно с сообщением.

Внимание! Использовать функцию MsgBox из Гедымина не рекомендуется, так как диалоговое окно не является модальным. Вместо нее следует использовать метод MessageBox глобального объекта Application.

##### Now

Возвращает текущие дату и время.

##### Now

**Oct**

Возвращает строку, представление числа в восьмеричной системе исчисления.

**Oct(*number*)**

Параметр должен быть числовым выражением. Дробные величины округляются до целого. Функция вернет Null, если передан Null, и 0, если входящий параметр Empty.

Обратите внимание, что восьмеричные значения можно указывать непосредственно в тексте программы, предворив их префиксом &O.

Пример:

Dim Octal

Octal = Oct(8) ' Octal = 10

##### Replace

Заменяет в строке заданную подстроку новым значением.

**Replace(*expression*, *find*, *replacewith*[, *start*[, *count*[, *compare*]]])**

expression Исходная строка.

find Искомая подстрока.

replacewith Новое значение, которое будет подставлено вместо подстроки find.

start Опциональный параметр. Номер символа в исходной строке, начиная с которого будет вестись поиск. По- умолчанию принимается равным 1. Данный параметр используется вместе с параметром count.

count Опциональный параметр. Показывает количество замен, которые будут выполнены. Значение по- умолчанию, -1, означает, что все найденые подстроки будут заменены. Используется вместе с параметром start.

compare Опциональный параметр. Может принимать значения vbBinaryCompare и vbTextCompare. Определяет каким образом будет осуществляться поиск: с учетом регистра символов или без него.

В зависимости от значений входных параметров результат функции будет следующим:

###### Если Replace вернет

expression пустая строка Пустую строку. expression равно Null Произойдет ошибка. find пустая строка Копию исходной строки.

replacewith пустая строка Исходную строку с удаленными вхождениями find.

start > Len(expression) Пустую строку. count = 0 Исходную строку.

Пример:

Dim S

S = Replace("1;2;3;4;5", ";", "-", 3, -1) ' S = "1;2-3-4-5"

##### RGB

Возвращает целое число, представляющее цвет в системе координат RGB.

##### RGB(red, green, blue)

Составляющие цвета должны быть числами в диапазоне от 0 до 255.

Б**о**льшие значения приравниваются к 255.

Первый, наименее значимый байт результата, содержит величину красной составляющей цвета, второй – зеленой, и третий – синей.

##### Right

Возвращает указанное число последних символов строки.

**Right(*string*, *number*)**

string Строка, из которой извлекается подстрока. Если указан Null, функция вернет также Null.

number Число символов. Функция вернет пустую строку, если number равно 0. Если number больше либо равно длине строки, то возвращается вся исходная строка.

Пример:

Dim S

S = Right("Golden Software", 8) ' S = "Software"

##### Rnd

Генерирует случайное число.

**Rnd([*number*])**

Функция возвращает значение в диапазоне 0 <= Rnd < 1. Значение параметра number определяет как будет генерироваться значение:

Меньше нуля Одно и то же значение при каждом вызове. Number используется для инициализации генератора случайных чисел.

Больше нуля Следующее случайное число в последовательности.

Равно нулю Повторно вернуть последнее сгенерированное случайное число.

Не указано Следующее случайное число в последовательности.

Оператор Randomize используется для инициализации генератора случайных чисел.

##### Randomize [number]

Если number не указан, используется текущее значение системного таймера. Для получения каждый раз одинаковой последовательности чисел, непосредственно перед обращением к Randomize должен идти вызов функции Rnd с отрицательным аргументом:

Rnd -1

Randomize 5

...

Rnd 1

...

##### Round

Возвращает число, округленное до указанного количества знаков после запятой.

**Round(*expression*[, *numdecimalplaces*])**

expression Числовое выражение.

numdecimalplaces Опциональный параметр. Количество знаков после запятой. Если не указан, округление будет производиться до целого числа.

##### ScriptEngine, ScriptEngineBuildVersion, ScriptEngineMajorVersion, ScriptEngineMinorVersion

Функции возвращают информацию о наименовании и версии скриптового языка.

##### Second

Извлекает секунды из времени.

**Second(*time*)**

Функция возвращает целочисленное значение в диапазоне от 0 до 59 включительно.

##### SetLocale Sgn

Знак числа.

**Sgn(*number*)**

Функция возвращает 1, если number больше нуля, -1, -- если меньше и 0, если заданный аргумент равен нулю.

##### Sin

Вычисляет синус аргумента.

**Sin(*number*)**

number Угол в радианах.

Функция вычисляет отношение противолежащего катета к гипотенузе для заданного угла в прямоугольном треугольнике. Результат находится в диапазоне от -1 до 1.

##### Space

Формирует строку из указанного количества пробелов.

**Space(*number*)**

Функция вернет пустую строку, если number = 0.

##### Split

Разбивает переданную строку на подстроки и заполняет ими возвращаемый массив.

**Split(string[, *delimiter*[, *count*[, *compare*]]])**

string Исходная строка. Если передана пустая строка "", функция вернет пустой массив нулевой длины.

delimiter Опциональный параметр. Символ или строка разделитель. Если в качестве разделителя задана пустая строка, функция вернет массив с одним элементом, равным исходной строке.

count Опциональный параметр. Максимальное количество элементов в массиве. По-умолчанию параметр имеет значение -1, что означает, что все выделенные подстроки будут занесены в массив.

compare Опциональный параметр. Показывает какой тип сравнения будет использоваться для поиска разделителя в строке: двоичный (без учета регистра) или текстовый (с учетом регистра символов). Параметр задается с помощью констант vbBinaryCompare и vbTextCompare.

Пример:

Dim A, I

A = Split("1, 2, 3, 4, 5", ",")

I = A(1) ' I = "2"

##### Sqr

Вычисляет квадратный корень из числа.

**Sqr(*number*)**

number Любое числовое выражение, больше или равно нулю. Пример:

Dim R

R = Sqr(4) ' R = 2

R = Sqr(-1) ' Ошибка

##### StrComp

Возвращает результат сравнения двух строк.

**StrComp(*string1*, *string2*[, *compare*])**

Функция вернет -1, если string1 < string2, 1, если string1 > string2 и 0, если строки равны. Если хотя бы одна из строк Null результат функции так же будет равен Null. Параметр compare является опциональным и задает тип сравнения: vbBinaryCompare – двоичное, с учетом регистра символов, или vbTextCompare – текстовое, без учета регистра символов. По-умолчанию используется двоичное сравнение. При сравнении строк разной длины, более короткая дополняется справа пробелами.

Пример:

Dim I

I = StrComp("БЕРЕЗА", "Береза", vbTextCompare) ' 0 I = StrComp("БЕРЕЗА", "Береза") ' -1

##### String

Формирует строку указанной длины, заполненную указанным символом.

**String(*number*, *char*)**

number Количество повторений. Если передан Null, функция вернет Null значение.

char Код символа или строка, первый символ которой будет использован для заполнения. Если передан Null, функция вернет Null значение. Код символа превышающий 255 будет сконвертирован по формуле char Mod 256.

Пример:

Dim S

S = String(10, "=") ' S = "=========="

##### StrReverse

Переставляет символы в переданной строке в обратном порядке.

**StrReverse(*string*)**

Если на вход функции передать Null – возникнет ошибка. Пример:

Dim S

S = StrReverse("Гедымин") ' S = "нимыдеГ"

##### Tan

Тангенс числа.

**Tan(*number*)**

number Задает угол в радианах. Для перевода градусов в радианы умножте величину в градусах на pi / 180.

Функция вычисляет тангенс угла прямоугольного треугольника, отношение длины противолежащего катета к прилежащему.

##### Time

Возвращает текущее время.

##### Time

Результат является вариантом с подтипом **Date**.

##### Timer

Возвращает количество прошедших с полуночи секунд.

##### Timer

Функция обеспечивает точность до одной миллисекунды (т.е. возвращает три знака после запятой) и удобна для измерения времени выполнения участка кода.

Пример:

Dim OldTime, TimeElapsed, N, I OldTime = Timer

N = 0

For I = 1 To 100000 N = N + 1

Next

TimeElapsed = Timer – OldTime ' Время выполнения цикла

##### TimeSerial

Возвращает вариант с подтипом **Date**, содержащий указанное время.

**TimeSerial(*hour*, *minute*, *second*)**

Для формирования времени в пределах суток значения параметров должны быть целочисленными и укладываться в следующие диапазоны (включая границы): для часов – 0..23; для минут и секунд – 0..59.

Превышение значением аргумента допустимого диапазона приводит к увеличению следующего, большего аргумента. Например, TimeSerial(0, 120, 0) вернет значение "2:00:00".

Отрицательные аргументы позволяют задать время относительно значения большего аргумента. Например, TimeSerial(5, -30, 0) соответствует времени "4:30:00". Знак аргумента часы не учитывается.

##### TimeValue

Возвращает вариант с подтипом **Date**, содержащий время.

**TimeValue(*time*)**

time Строка, литерал времени или вариант с подтипом дата. Допускается использование как 12-ти часового, так и 24-х часового формата времени.

Если на вход передается дата и время, функция вернет только время.

##### TypeName

Возвращает строковое наименование подтипа варианта.

**TypeName(*varname*)**

Функция возвращает следующие значения:

Byte Однобайтовое целое.

Integer Двубайтовое целое.

Long Четырехбайтовой целое.

Single Число с плавающей точкой одинарной точности. Double Число с плавающей точкой двойной точности.

Currency Число с фиксированной точкой. Decimal Число с фиксированной точкой.

Date Дата или время.

String Строка.

Boolean Булевское значение.

Empty Переменная не инициализирована.

Null Переменная не содержит данных.

<object type> Имя типа объекта. Object Объект.

Unknown Объект неизвестного типа.

Nothing Переменная объектного типа, которой не присвоена ссылка на объект.

Error Ошибка. Пример:

Dim A, V, Creator, T

Set Creator = New TCreator A = 5

V = Array(1, 2, 3)

T = TypeName(A) ' T = "Integer" T = TypeName(Creator) ' T = "TCreator"

T = TypeName(V) ' T = "Variant()"

##### UBound

Возвращает верхнюю границу массива.

**UBound(*array*[, *dimension*])**

arrayname Имя массива.

dimension Опциональный параметр. Номер размерности массива, начиная с 1. Если не задан, подразумевается первая размерность.

Функция UBound используется вместе с LBound для определения размера массива.

Пример:

Dim A(200, 4) Dim D1, D2

D1 = UBound(A, 1) ' D1 = 200 D2 = UBound(A, 2) ' D2 = 4

##### UCase

Преобразует символы строки к верхнему регистру.

**UCase(*string*)**

Пример:

Dim CityName

CityName = UCase("Mensk") ' CityName = "MENSK"

##### VarType

Возвращает подтип переменной.

**VarType(*varname*)**

Список возможных значений функции приведен в разделе Константы, Подтипы варианта.

Для массива значение функции будет являтся суммой констант vbArray и константы, соответствующей типу элементов массива. Например, для массива строк значение будет: vbArray + vbString = 8192 + 8 = 8200. Если объект имеет свойство по-умолчанию (default property), то функция VarType(object) вернет тип этого свойства.

##### Weekday

Номер дня недели для указанной даты.

##### Weekday(date[, firstdayofweek])

date Дата.

firstdayofweek Опциональный параметр. Константа, которая показывает какой день недели считается первым. Если не задана, то воскресенье будет первым днем. Список возможных значений приведен в разделе "Константы".

Пример:

Dim WD

WD = Weekday(#02/12/1976#, vbMonday) ' WD = 4 -- четверг

##### WeekdayName

Возвращает наименование дня недели.

**WeekDay(*weekday*, *abbreviate*[, *firstdayofweek*])**

weekday Номер дня недели. Число от 1 до 7.

abbreviate Если True, функция вернет сокращенное наименование дня недели, если False – полное.

firstdayofweek С какого дня начинается неделя. См. раздел "Константы" для более подробной информации. Параметр опциональный, если не указан – первым днем будет считаться воскресенье.

Пример:

Dim MyDate

MyDate = WeekdayName(4, False, vbMonday) ' Вернет "четверг"

##### Year

Извлекает год из даты.

**Year(*date*)**

date Любое выражение, представляющее дату.

## Обработка ошибок

VBScript имеет весьма скудные встроенные средства обработки ошибок времени выполнения, которые сводятся к двум операторам:

On Error GoTo 0

On Error Resume Next

Первый – отключает обработку ошибок (состояние по-умолчанию). Если ошибка случится, то на экран будет выдано сообщение и выполнение текущей процедуры или функции прекратится.

При включенном режиме **On Error Resume Next**, ошибка не приводит к прерыванию выполнения кода и программист имеет возможность самостоятельно разобраться с ней. Информация об ошибке содержится в глобальном встроенном объекте **Err**. Ниже приводится описание его свойств и методов:

Description Текстовое описание ошибки.

HelpContext Номер топика в файле справки, который будет открыт, если пользователь воспользуется клавишей F1 в окне сообщения об ошибке.

HelpFile Имя файла справки.

Number Целочисленный номер ошибки. 0 означает отсутствие ошибки.

Source Наименование приложения, в котором произошла ошибка.

Методы:

Clear Очищает все свойства объекта, сбрасывает информацию об ошибке. Обратите внимание, что объект Err автоматически очищается при выполнении следующих операторов: **On Error Resume Next**, **Exit Sub**, **Exit Function**.

Raise Генерирует ошибку времени выполнения. Полный формат вызова: Err.Raise(number, source, description, helpfile, helpcontext). Только первый параметр (номер ошибки) является обязательным. Если опущены, значения остальных параметров подставляются из соответствующих свойств объекта.

Обработку ошибок проиллюстрируем на примере кода, который в цикле запрашивает у пользователя строку и пытается преобразовать ее в дату. Если введенная строка не является датой, то на экран выводится сообщение. Цикл повторяется, пока пользователь не введет корректную дату.

Dim InputD, InputS On Error Resume Next Do

Err.Clear

InputS = InputBox("Введите дату")

If IsNumeric(InputS) Then Err.Number = vbObjectError + 1

Else

InputD = CDate(InputS) End If

If Err.Number <> 0 Then Application.MessageBox \_

"Неверный формат даты! Повторите ввод.", \_ "Ошибка", vbOkOnly + vbExclamation + vbSystemModal

End If

Loop Until Err.Number = 0 On Error Goto 0

## Использование TCreator

Бережное отношение к ресурсам компьютера – неприменный атрибут качественной программы. Сегодня, когда 1 Гб оперативной памяти на рабочей станции обычное дело, а операционные системы не имеют таких жестких ограничений на количество внутренних объектов, которые присутствовали во времена Windows 9х, многие разработчики не уделяют должного внимания вопросам экономного создания и удаления объектов. Как результат, приложение, вроде бы неплохо работающее на настольном компьютере, не запускается на POS терминале, оснащенном 128 Мб ОЗУ. Едва заметные утечки памяти могут привести к фатальным последствиям, если пользователь не выключает компьютер на протяжении нескольких дней или использует приложение особенно интенсивно. Многочисленные одновременные подключения к базе данных из программ, небрежно обращающихся с такими объектами, как транзакция или запрос, способны поставить в затруднительное положение даже очень производительный сервер.

Следует создавать объекты по мере необходимости и уничтожать их незамедлительно, по окончании использования. Исключением является ситуация, когда многократное создание и удаление объекта влекут значительные временные затраты. В этом случае можно рекомендовать использование глобальной переменной или пула объектов.

Программируя на Гедымине мы имеем дело с объектами трех видов:

1. Экземпляры VBScript классов, созданные оператором **New**.
2. COM объекты автоматизации (ActiveX объекты), созданные функцией

###### CreateObject.

1. Объекты платформы Гедымин, созданные вызовом метода

###### Designer.CreateObject.

Для первых двух в среде выполнения VBScript действует автоматическая сборка мусора. Как только исчезает необходимость в объекте – он уничтожается. Как правило, это происходит по выходу из процедуры или функции, в рамках которой был создан объект. Программист имеет возможность форсировать данный процесс, присвоив значение **Nothing** соответствующей локальной переменной. С последними -- дело обстоит сложнее. Поскольку создаются они внутри платформы и не являются COM объектами, VBScript не может ни контролировать их использование, ни самостоятельно их удалять. Такая обязанность всецело возлагается на программиста.

Казалось бы, чего проще: Designer.CreateObject-Designer.DestroyObject – и все дела (см. пример ниже).

Dim Obj

Set Obj = Designer.CreateObject(...)

...

Designer.DestroyObject(Obj)

Однако, если мы не используем режим **On Error Resume Next**, то любая ошибка в процессе выполнения приведет к преждевременному выходу из тела процедуры. Строка кода Designer.DestroyObject выполнена не будет и объект останется в памяти до завершения выполнения программы. Ниже приведен пример такой, небезопасной функции. Ее задача – вернуть наименование контакта по заданному идентификатору. Если внутри функции произойдет ошибка (например, на вход будет передана строка буквенных символов вместо целочисленного идентификатора), то объект TIBSQL, сохраненный в переменной q, уничтожен не будет.

Function UnSafe\_GetContactName(ID) Dim q

Set q = Designer.CreateObject(nil, "TIBSQL", "") q.Transaction = gdcBaseManager.ReadTransaction q.SQL.Text = \_

"SELECT name FROM gd\_contact WHERE id=" & CLng(ID) q.ExecQuery

UnSafe\_GetContactName = q.Fields(0).AsString Designer.DestroyObject(q)

End Function

К счастью, разработчики платформы нашли элегантное решение данной проблемы. Как отмечалось выше, локальные VBScript объекты автоматически уничтожаются при выходе из процедуры или функции. Был создан вспомогательный класс TCreator, который хранит список созданных объектов и уничтожает их в деструкторе. Теперь мы можем сделать безопасной нашу функцию:

Function Safe\_GetContactName(ID) Dim Creator, q

Set Creator = New TCreator

Set q = Creator.GetObject(nil, "TIBSQL", "") q.Transaction = gdcBaseManager.ReadTransaction q.SQL.Text = \_

"SELECT name FROM gd\_contact WHERE id=" & CLng(ID) q.ExecQuery

Safe\_GetContactName = q.Fields(0).AsString End Function

От своего небезопасного аналога данная функция отличается лишь тем, что мы добавили переменную Creator и создаѐм объект через метод Creator.GetObject вместо Designer.CreateObject. Обратите внимание, что мы не удаляем Creator непосредственно -- об этом позаботится VBScript.

Класс TCreator входит в пакет **Общие данные**. Ниже приведен его полный исходный код.

Class TCreator Private FCount

Private FObjectArray()

Public Sub DestroyAllObjects

for I = UBound(FObjectArray) to LBound(FObjectArray) step -1

if VarType(FObjectArray(I)) = vbObject then FObjectArray(I).DestroyObject FObjectArray(I) = Empty

end if next

FCount = -1 End Sub

Public Function GetObject(Params, ClassName, Name) if FCount >= UBound(FObjectArray) then

ReDim Preserve FObjectArray((UBound(FObjectArray) + 1)

\* 2 - 1)

end if

set FObjectArray(FCount + 1) = Designer.CreateObject(Params, ClassName, Name)

set GetObject = FObjectArray(FCount + 1) FCount = FCount + 1

End Function

'Используется, в случае необходимости,

'для уничтожения объктов созданных Креатором

Public Sub DestroyObject(Object)

for I = UBound(FObjectArray) to LBound(FObjectArray) step -1

if VarType(FObjectArray(I)) = vbObject then

if Addr(FObjectArray(I)) = Addr(Object) then FObjectArray(I) = Empty Object.DestroyObject

exit sub end if

end if next

call Exception.Raise("Exception", "В списке не найден переданный объект.")

End Sub

Private Sub Class\_Initialize ReDim FObjectArray(7) FCount = -1

End Sub

Private Sub Class\_Terminate DestroyAllObjects

End Sub End Class

# Глава VII. Краткий курс SQL

Хорошее знание SQL и теоретических основ реляционных баз данных – одно из основных требований, предъявляемых к разработчику на платформе Гедымин. Без него невозможно создать отчет, разработать новый бизнес-класс или понять что же происходит "под капотом" во время работы системы. Данная глава содержит краткий учебный курс и рассчитана на начинающего пользователя, который либо совсем не знаком с реляционными базами и SQL, либо владеет только самыми элементарными познаниями и хотел бы узнать больше. Для получения более подробной информации мы рекомендуем обратиться к специализированным учебникам.

Специалистам, имеющим опыт работы с такими РСУБД40, как Microsoft SQL Server, Oracle, SyBase и т.д., будет интересно ознакомиться с синтаксисом языка триггеров и хранимых процедур сервера Firebird, также с другими его особенностями, отличными от стандарта SQL.

Практическая часть учебного курса рассчитана на последовательное изучение. Как правило, каждый очередной приведенный пример зависит от ранее созданных объектов и/или занесенной в базу данных информации.

Все приведенные в настоящей главе упражнения мы будем выполнять на демонстрационной базе данных в окне **SQL Редактора**.

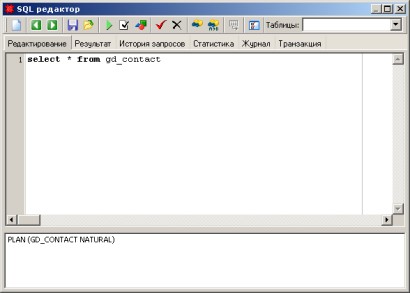
#### Знакомство с окном SQL Редактора

Выполним следующие действия:

1. Запустим Гедымин.
2. Выберем из списка демонстрационную базу данных.
3. Введем наименование учетной записи **Administrator** и ее пароль.
4. В главном окне программы из меню **Сервис** выберем пункт **SQL Редактор**.

На экране откроется окно следующего вида:

40 СУБД – Система управления базами данных. Соответственно, РСУБД – Реляционная СУБД.



*Рис. 91. Окно SQL Редактора*

В верхней части окна (см. Рис. 91) находится панель инструментов. Под ней – набор вкладок для работы с запросом. Поле в нижней части отображает либо план запроса, в случае его успешного выполнения, либо сообщение об ошибке. Список доступных команд приведен в следующейтаблице:

|  |  |
| --- | --- |
|  | Очистить текст запроса. Подготовить окно для ввода нового  запроса (горячая клавиша: Ctrl-Shift-N). |
|  | Открыть предыдущий запрос. |
|  | Открыть следующий запрос. |
|  | Сохранить запрос в файле. |
|  | Загрузить запрос из файла. |
|  | Выполнить запрос (F9). |
|  | Подготовить запрос к выполнению, показать его план (Ctrl-F9). |
|  | Разобрать запрос парсером, подставить поля. |
|  | Подтвердить транзакцию (Ctrl-Alt-C) -- комит. |
|  | Откатить транзакцию (Ctrl-Alt-R) -- роллбэк. |
|  | Поиск в тексте запроса (Ctrl-F). |
|  | Поиск и замена в тексте запроса. |
|  | Открыть диалоговое окно со свойствами SQL Редактора. |
| Таблицы: | Список таблиц в базе данных. |

Обычно, работа начинается с вкладки **Редактирование**. Здесь мы вводим текст запроса и выполняем его с помощью клавиши F9. Если запрос содержит параметры, имена которых начинаются с символа двоеточия, то на экране появится окно для их ввода.

При наборе текста система всегда готова прийти нам на помощь. Достаточно в той части запроса, где подразумевается ввод имени таблицы или поля, нажать коомбинацию клавиш **Ctrl-Пробел** и на экране появится список всех доступных объектов.

После успешного выполнения откроется вкладка **Результат** с выходным набором данных. Обратите внимание на локальную панель инструментов данной вкладки:

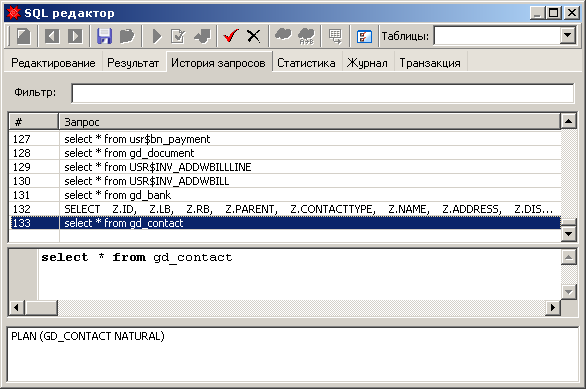


*Рис. 92. Панель инструментов вкдадки Результат*

Кроме кнопок навигации, на ней находятся команды редактирования, удаления, объединения и сохранения в файл выделенных записей. Указанные команды доступны только в том случае, если SQL запрос выводит список бизнес-объектов.

##### История запросов

На вкладке (см. Рис. 93) отображается список ранее выполненных запросов.

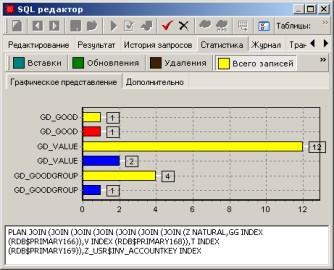


*Рис. 93. Вкладка История запросов*

Двойным щелком можно переслать выделенный запрос на вкладку Редактирование, а контекстное меню содержит дополнительные команды для работы со списком.

##### Статистика

Важная информация для оптимизации запроса. На гистограмме показывается количество индексированных и неиндексированных чтений из таблиц, входящих в запрос (см. Рис. 94).



*Рис. 94. Вкладка со статистикой запроса*

Эта же информация приводится на вкладке Дополнительные в числовом виде.

Более подробно тема оптимизации SQL запросов будет обсуждаться в одном из разделов ниже.

##### Журнал

Вкладка журнал содержит лог (все выполненные запросы) текущей сессии.

##### Транзакция

На данной вкладке можно просмотреть и/или изменить параметры текущей транзакции. Более подробно о транзакциях речь пойдет в соответствующем разделе ниже.

## Введение в реляционные БД

Информация в реляционной базе данных хранится в **таблицах** и **связях** между ними. Таблицы двумерны, имеют фиксированное число **колонок** и произвольное, ограниченное только свободным пространством на жестком диске, количество **записей**. Каждая колонка имеет уникальное в пределах таблицы наименование и содержит данные одного типа. Ниже мы будем использовать термины **поле, столбец** и **строка,** как равнозначные синонимы слов колонка и запись соответственно.

Вы наверняка знакомы с такими понятиями как "файл-сервер" и "клиент-сервер". Они описывают способ взаимодействия приложения с

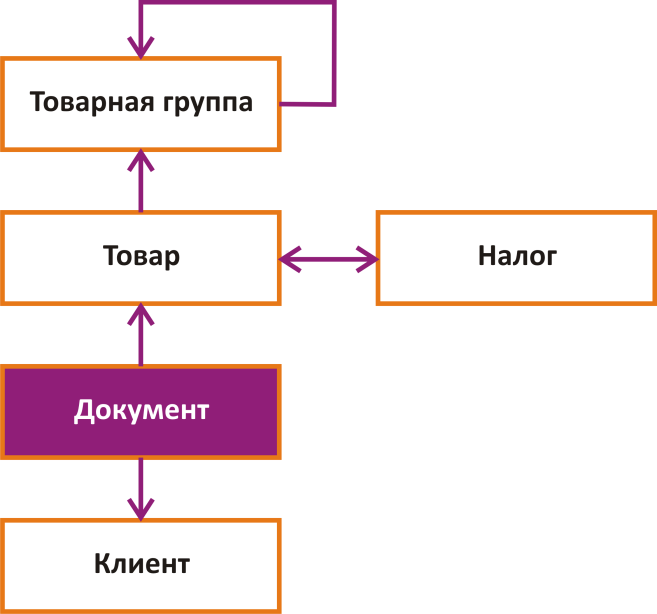
данными, которые хранятся на выделенном компьютере в сети. В первом случае программа обращается напрямую к файлам, считывает их и выполняет всю обработку непосредственно на локальной машине. При более простой реализации, такой способ порождает избыточный сетевой трафик и страдает пониженной устойчивостью к поломкам данных в случае сбоев сетевого оборудования, компьютеров или программного обеспечения. Файл-серверные СУБД были широко распространены во времена операционной системы DOS и являлись результатом перевода настольных приложений в многопользовательский режим работы. Вряд ли найдется компьютерный специалист, не знакомый с такими названиями как dBase, FoxPro или Paradox.

В случае архитектуры клиент-сервер, **клиент** посылает на **сервер** запрос на специальном языке. Приняв и обработав запрос, сервер возвращает клиенту готовый результат или сообщение об ошибке. Единственную машину, занимающуюся хранением и обработкой данных, гораздо проще защитить от сбоев, а меньший, по сравнению с файл-сервером, сетевой трафик позволяет использовать низкоскоростные соединения для подключения удаленных компьютеров.

Платформа Гедымин построена по принципу клиент-сервер с использованием СУБД Firebird в качестве хранилища данных.

В настоящее время, для комуникации между клиентом и сервером наибольшее распространение получил язык **SQL** (акроним от английского Structured Query Language -- структурированный язык запросов). SQL подразделяется на три подмножества: Data Definition Language, Data Manipulation Language и Data Control Language. **DDL** определяет набор команд для создания **метаданных** или, иными словами, структуры базы данных: таблиц, доменов, индексов, процедур и т.п. На языке **DML** пишутся запросы на извлечение, вставку, изменение или удаление данных. И, наконец, язык **DCL** содержит набор операторов разграничения доступа для различных пользователей СУБД.

В духе нашей книги, осваивать азы SQL мы будем на практическом примере. Поставим перед собой следующую задачу: реализовать учет складского движения. Ради краткости и простоты ограничимся всего двумя операциями: покупка товара (приход) и его последующая продажа (расход). Из всех возможных характеристик оставим только наименование и принадлежность товара к определенной группе. Последний атрибут будет необязательным для заполнения.



*Рис. 95. Схема базы данных*

Анализ задачи позволяет выделить четыре класса объектов: Товарная группа;



Товар; Контрагент; Документ.

Для каждого класса создадим таблицу, колонки которой определяются набором его атрибутов. Конкретный объект будет соответствовать записи в таблице. Для идентификации объектов мы будем использовать уникальное целое число.

Установим характеристики и атрибуты для каждой сущности:

##### Товарная группа

Может содержать ноль, один или несколько товаров. Имеет уникальный идентификатор и наименование.

##### Товар

В дальнейшем мы будем использовать аббревиатуру ТМЦ (Товарно- матеральная ценность) в качестве синонима слова товар. Имеет уникальный идентификатор и наименование. Может входить в товарную группу.

##### Контрагент

Выступает в роли поставщика или покупателя товаров. Как и предыдущие объекты имеет идентификатор и наименование.

##### Документ

Регистрирует факт покупки (поступления) или продажи (расхода) товара. Так как по одному документу могут проходить несколько товаров, для для хранения данных потребуются две связанные таблицы: Заголовок и Позиция документа.

Заголовок документа:

* 1. Идентификатор.
  2. Номер.
  3. Дата.
  4. Контрагент. Позиция документа

1. Идентификатор.
2. Ссылка на заголовок.
3. Ссылка на товар.
4. Количество (приход отражается положительным числом, расход

– отрицательным).

1. Цена единицы товара.

Как правило, задачи подобного рода решаются в следующей последовательности:

1. Создаются типы данных;
2. Создаются таблицы;
3. Для каждой таблицы создаются триггеры и хранимые процедуры, отвечающие за логическую целостность данных;
4. Корректность созданных структур проверяется на тестовом наборе данных.

#### Типы данных

Выше мы отмечали, что колонка таблицы содержит однородные данные.

К нашим услугам следующие встроенные типы:

|  |  |
| --- | --- |
| **Тип данных** | **Описание** |
| INTEGER | 32-х битовое целое число в диапазоне -2 147 483  648..2 147 483 647. |
| SMALLINT | 16-ти битовое целое число в диапазоне –32 768..32  767. |
| DOUBLE PRECISION | 64-ти битовое число с плавающей точкой в  диапазоне 5.0 x 10–324 .. 1.7 x 10308 (15-16 значащих цифр). |
| FLOAT | 32-ти битовое число с плавающей точкой в диапазоне 1.5 x 10–45 .. 3.4 x 1038 (7-8 значащих  цифр). |
| NUMERIC(x, y) DECIMAL(x, y) | Вещественное число с фиксированной десятичной точкой. Параметр x задает максимальное количество цифр в числе, y – количество цифр после десятичной точки. Например, поле, определенное как NUMERIC(6, 2), позволяет хранить числа от -9999.99 до 9999.9941.  Тип данных NUMERIC(18, 0) определяет 64-х битное целое число. |
| DATE | Дата в диапазоне от 1-го января 100-го года до 29-го  февраля года 32768. |
| TIME | Время в диапазоне от 00:00:00 до 23:59:59.9999 |
| TIMESTAMP | Дата и время. |
| CHAR(n), CHARACTER(n) | Строка фиксированной длины из n символов. Короткие строки, при записи в поле типа CHAR, дополняются справа пробелами до ширины  колонки, длинные -- обрезаются. Если длина |

41 На самом деле поле, определенное как NUMERIC(6, 2), позволяет хранить гораздо большие числа. Попробуем разобраться почему. Физически число с фиксированной точкой хранится как целое. Например, при записи в нашу колонку вещественного значения 5023.89 сервер переместит десятичную точку вправо и сохранит его как 502389. При считывании будет произведена обратная операция – перемещение десятичной точки влево на две позиции. Сервер умеет работать с тремя целочисленными типами -- 16, 32 и 64 бита длиною соответственно. При создании колонки типа NUMERIC(x, y) для хранения данных будет выбрано самое короткое целое в котором можно раместить хотя бы x цифр. Получается, что в поле типа NUMERIC(6, 2) можно поместить значение до 21 474 836.47, так как для его хранения будет использовано 32-х битное целое число.

|  |  |
| --- | --- |
|  | строки не указана, принимается n=1. |
| VARCHAR(n), CHARACTER VARYING(n) | Строка переменной длины. Тип VARCHAR хранит ровно столько символов, сколько было помещено в базу командой INSERT или UPDATE. При создании колонки в скобках обязательно указывается  максимальная длина строки. |
| BLOB | Двоичный объект. Применяется для хранения в базе картинок, файлов, больших текстов, потоков  данных и т.п. |

##### Кодовая таблица символов

Кодовая таблица задается для строкового поля42 и определяет набор допустимых символов и порядок их сравнения. Например, CHARACTER SET WIN1251 COLLATE PXW\_CYRL установит кодовую таблицу WIN1251 (кирилица) и порядок сравнения в соответствии с алфавитом русского языка. В зависимости от выбранной кодовой таблицы для хранения одного символа может потребоваться от одного до трех байт. Так как максимальный размер строкового поля составляет 32 767 байт, максимальная длина строки в символах находится в диапазоне от 32 767 (для однобайтовых кодировок) до 10 922 (кодировка UNICODE\_FSS – три байта на символ).

##### Подтипы BLOB

Для большого двоичного объекта можно дополнительно указать его подтип. Подтипом является целое число. Значение 0 соответствует неструктурированным двоичным данным; 1 – тексту. В последнем случае, как и для строкового поля, можно указать кодовую таблицу символов. Прочие положительные значения зарезервированы для системных нужд. Отрицательные подтипы пользователь вправе назначать и использовать произвольным образом.

##### Пользовательские типы данных

Диапазоны значений встроенных типов в большинстве случаев намного шире, чем требуется для конкретного атрибута бизнес сущности. Так, в задаче по учету основных фондов предприятия срок полезной эксплуатации объекта должен быть строго больше нуля и вряд ли когда-

42 Если не задана непосредственно, применяется кодовая таблица, установленная для базы данных.

либо превысит 200 лет. Стоимость единицы не может быть отрицательной, а повышающий коэффициент задается в диапазоне от 1 до 2. Конечно, в данных случаях можно использовать один из встроенных числовых типов, полагаясь на аккуратность оператора или проверку допустимых значений на уровне пользовательского интерфейса. Но, что произойдет, если человек ошибется? В дальнейшем мы получим неверные данные или сообщения об ошибках.

Правилом хорошего тона является создание типа данных в соответствии с областью допустимых значений поля. Такие типы в терминологии SQL называются **доменами**.

#### Создание домена

Приступим к созданию типов в рамках рассматриваемой задачи.

Напишем и выполним следующую команду:

CREATE DOMAIN lsn\_dintkey AS INTEGER NOT NULL

Мы создали целочисленный тип с именем **lsn\_dintkey**, обязательный для заполнения – **NOT NULL**. В дальнейшем, мы будем использовать его для идентификаторов записей.

Как видно, синтаксис команды создания домена очень прост: достаточно указать имя домена, базовый тип данных и некоторые накладываемые ограничения.

Пойдем дальше. Для хранения имени объекта создадим следующий обязательный для заполнения строковый тип длиной в 60 символов, в кирилической кодировке:

CREATE DOMAIN lsn\_dname AS VARCHAR(60)

CHARACTER SET WIN1251 NOT NULL COLLATE PXW\_CYRL

Домен для колонки номера документа определим следующим образом:

CREATE DOMAIN lsn\_dnumber AS VARCHAR(20)

DEFAULT 'б/н'

CHARACTER SET WIN1251 NOT NULL COLLATE PXW\_CYRL

Обратите внимание, как с помощью DEFAULT 'б/н' мы задали значение по-умолчанию. Это значение будет присвоено колонке, если она явно не задана в операторе вставки записи.

По логике нашей задачи, цена единицы товара положительная величина строго большая нуля. С помощью секции CHECK оператора создания домена мы можем накладывать ограничения на данные. Так как в момент

создания домена имя будущей колонки еще не известно, следует использовать ключевое слово VALUE:

CREATE DOMAIN lsn\_dprice AS NUMERIC(15, 2) CHECK(VALUE > 0)

Таким образом, мы создали необходимые для нашей задачи типы даных и заодно познакомились с несколькими примерами использования команды CREATE DOMAIN. Полный синтаксис команды создания домена выглядит следующим образом:

CREATE DOMAIN domain [AS] <datatype> [DEFAULT {literal | NULL | USER}]

[NOT NULL] [CHECK (<dom\_search\_condition>)] [COLLATE collation]

<datatype> = SMALLINT | INTEGER

| FLOAT | DOUBLE PRECISION

| {DECIMAL | NUMERIC} [(precision [, scale])]

| {DATE | TIME | TIMESTAMP}

| {CHAR | CHARACTER | CHARACTER VARYING | VARCHAR}

[(int)] [CHARACTER SET charname]

| {NCHAR | NATIONAL CHARACTER | NATIONAL CHAR} [VARYING]

[(int)]

<dom\_search\_condition> = VALUE <operator> <val>

| VALUE [NOT] BETWEEN <val> AND <val>

| VALUE [NOT] LIKE <val> [ESCAPE <val>]

| VALUE [NOT] IN (<val> [, <val> …])

| VALUE IS [NOT] NULL

| VALUE [NOT] CONTAINING <val>

Однажды созданный домен может быть изменен с помощью команды ALTER DOMAIN:

ALTER DOMAIN { name | old\_name TO new\_name } { [SET DEFAULT {literal | NULL | USER}]

| [DROP DEFAULT]

| [ADD [CONSTRAINT] CHECK (<dom\_search\_condition>)]

| [DROP CONSTRAINT]

| new\_col\_name

| TYPE data\_type

}

Обратите внимание, что признак NOT NULL не может быть изменен с помощью данной команды, равно как изменение типа данных возможно только в том случае, если область допустимых значений нового типа шире, чем существующего.

Последняя, третья команда для работы с доменами – DROP DOMAIN – удаляет существующий домен:

DROP DOMAIN name

Команда завершится с сообщением об ошибке, если указанный домен уже используется в объявлении, хотя бы одной колонки. В этом случае следует сначала удалить колонку из таблицы, а затем удалить домен.

Вообще, стоит заметить, что почти для всех объектов метаданных DDL предусматривает по три команды: CREATE – для создания, ALTER – для изменения и DROP – для удаления.

#### Использование префиксов в именах объектов

У вдумчивого читателя к этому моменту уже наверняка возник естественный вопрос: почему мы используем такие неудобочитаемые имена

– lsn\_dname, в место того, чтобы просто назвать домен name? Для ответа достаточно беглого взгляда на структуру любой реальной базы данных. Как правило, она содержит несколько тысяч структурных объектов: таблиц, триггеров, хранимых процедур, представлений. Объектов, созданных разными программистами в разное время. Только грамотное использование префиксов позволяет нам не потеряться в этой грандиозной куче и избежать конфликтов имен.

Согласно принятым соглашениям мы даем каждому объекту префикс, состоящий из идентификатора подсистемы и сокращенного обозначения типа метаданных, разделенных символом подчеркивания. Таким образом, префикс "lsn\_d" в имени "lsn\_dname" указывает на домен – "d", из подсистемы "lsn" -- "Учебный курс SQL" (сокращение lsn от английского lesson -- урок).

#### Создание таблицы

Теперь, когда мы разобрались с доменами, самое время перейти к созданию таблиц. Как вы уже догадались, соответствующая команда называется CREATE TABLE. Начнем с самого простого – справочника товарных групп:

CREATE TABLE lsn\_group ( id lsn\_dintkey,

name lsn\_dname

)

и имеет следующий синтаксис:

CREATE TABLE table

(<col\_def> [, <col\_def> | <tconstraint> ...])

Таблица должна иметь уникальное имя и хотя бы одну колонку. Обязательными параметрами для создания колонки являются ее наименование, уникальное в пределах таблицы, и тип данных. Последний задается указанием одного из встроенных типов, доменом, либо выражением, на основании которого вычисляется значение поля.

<col\_def> = col {datatype | COMPUTED [BY] (<expr>) | domain} [DEFAULT {literal | NULL | USER}]

[NOT NULL] [<col\_constraint>] [COLLATE collation]

<datatype> =

{SMALLINT|INTEGER|FLOAT|DOUBLE PRECISION}[<array\_dim>]

| {DATE|TIME|TIMESTAMP}[<array\_dim>]

| {DECIMAL | NUMERIC} [(precision [, scale])] [<array\_dim>]

| {CHAR | CHARACTER | CHARACTER VARYING | VARCHAR} [(int)]

[<array\_dim>] [CHARACTER SET charname]

| {NCHAR | NATIONAL CHARACTER | NATIONAL CHAR}

[VARYING] [(int)] [<array\_dim>]

| BLOB [SUB\_TYPE {int | subtype\_name}] [SEGMENT SIZE int] [CHARACTER SET charname]

| BLOB [(seglen [, subtype])]

Как и в случае с доменом мы имеем возможность указать значение по умолчанию и признак обязательного заполнения поля. Новым для нас является возможность задать одно или несколько правил целостности данных:

<col\_constraint> = [CONSTRAINT constraint] <constraint\_def> [<col\_constraint> ...]

<constraint\_def> = UNIQUE | PRIMARY KEY

| CHECK (<search\_condition>)

| REFERENCES other\_table [(other\_col [, other\_col …])] [ON DELETE {NO ACTION|CASCADE|SET DEFAULT|SET NULL}] [ON UPDATE {NO ACTION|CASCADE|SET DEFAULT|SET NULL}]

<tconstraint> = [CONSTRAINT constraint] <tconstraint\_def> [<tconstraint> ...]

<tconstraint\_def> = {PRIMARY KEY | UNIQUE} (col [, col …])

| FOREIGN KEY (col [, col …])

REFERENCES other\_table [(other\_col [, other\_col …])] [ON DELETE {NO ACTION|CASCADE|SET DEFAULT|SET NULL}] [ON UPDATE {NO ACTION|CASCADE|SET DEFAULT|SET NULL}]

| CHECK (<search\_condition>)

<search\_condition> =

<val> <operator> {<val> | (<select\_one>)}

| <val> [NOT] BETWEEN <val> AND <val>

| <val> [NOT] LIKE <val> [ESCAPE <val>]

| <val> [NOT] IN (<val> [, <val> ...] | <select\_list>)

| <val> IS [NOT] NULL

| <val> {[NOT] {= | < | >} | >= | <=}

{ALL | SOME | ANY} (<select\_list>)

| EXISTS (<select\_expr>)

| SINGULAR (<select\_expr>)

| <val> [NOT] CONTAINING <val>

| <val> [NOT] STARTING [WITH] <val>

| (<search\_condition>)

| NOT <search\_condition>

| <search\_condition> OR <search\_condition>

| <search\_condition> AND <search\_condition>

##### Первичный ключ записи

По условиям нашей задачи требуется создать пять таблиц. Начнем со справочника товарных групп:

CREATE TABLE lsn\_group ( id lsn\_dintkey,

name lsn\_dname UNIQUE,

CONSTRAINT lsn\_pk\_id PRIMARY KEY (id)

)

CREATE TABLE lsn\_company ( id lsn\_dintkey,

name lsn\_dname UNIQUE

)

**Древовидные структуры**

Деревом называется упорядоченный граф, в котором от каждого узла к вершине ведет только один путь. С помощью дерева можно представить структуру каталогов и файлов на жестком диске, иерархию подразделений и сотрудников компании или административно-территориальное строение государства.



*Рис. 96. Дерево*

На **Ошибка! Источник ссылки не найден.** приведен пример дерева. Вершина А называется **корнем** дерева. Узлы, которые не имеют поддеревьев

— C, D, F, G и H — называются **листьями**. Дуги дерева показывают отношения родитель-потомок между его узлами. Дерево называется n- мерным, если каждый его узел имеет не более чем n потомков. Частный случай n-мерного дерева — двоичное дерево, когда каждый узел содержит не более двух потомков.

#### Простое дерево

Пожалуй, самый простой способ представления дерева в реляционной базе — это помещение матрицы смежности в таблицу. Каждая запись такой таблицы соответствует узлу дерева и хранит его уникальный идентификатор и ссылку на родительский узел.

Ниже приведен DDL запрос для создания таблицы TEST1:

CREATE TABLE test1 (

id INTEGER NOT NULL,

parent INTEGER,

PRIMARY KEY (id),

FOREIGN KEY (parent) REFERENCES test1 (id) ON UPDATE CASCADE

ON DELETE CASCADE

);

Обратите внимание на правило **ON DELETE CASCADE**, заданное для внешнего ключа по полю PARENT. Благодаря ему, при удалении узла, будет автоматически удалено и поддерево, для которого этот узел является корнем.

Очевидно, что при такой организации таблицы, с помощью одного SQL запроса можно извлечь только определенное число дочерних уровней. Ниже приведен запрос, который извлекает идентификаторы всех потомков для узла, заданного параметром Р.

SELECT id FROM

test1 WHERE

parent = :P

Если мы хотим узнать список потомков для всех потомков узла Р, то запрос следует переписать следующим образом:

SELECT t1.id FROM

test1 t1 JOIN test1 t2 ON t1.parent = t2.id

WHERE

t2.parent = :P

Для продвижения вглубь дерева придется добавлять все больше и больше таблиц в самообъединение.

Еще хуже обстоит дело, если необходимо извлечь список всех узлов поддерева, независимо от уровня вложенности. В этом случае придется использовать оператор **UNION** для объединения результатов нескольких запросов, возвращающих узлы конкретного уровня. Например, SQL для извлечения всех потомков узла Р и всех потомков потомков будет выглядеть следующим образом:

SELECT id FROM

test1 WHERE

parent = :P UNION

SELECT t1.id FROM

test1 t1 JOIN test1 t2 ON t1.parent = t2.id

WHERE

t2.parent = :P

Если высота дерева заранее неизвестна, то для извлечения всех узлов поддерева, для которого заданный узел является вершиной, придется создать рекурсивную процедуру.

Пример такой процедуры приведен ниже:

CREATE PROCEDURE RecTest1(AnID INTEGER, Self INTEGER) RETURNS (ID INTEGER)

AS BEGIN

IF (:Self <> 0) THEN BEGIN

ID = :AnID;

SUSPEND;

END

FOR

SELECT id FROM test1 WHERE parent = :AnID INTO :ID DO

FOR

SELECT id FROM RecTest1(:ID, 1) INTO :ID DO

SUSPEND;

END;

Второй входящий параметр процедуры определяет будет ли присутствовать заданный идентификатор вершины поддерева в результирующей выборке.

Тогда запрос, извлекающий все вложенные уровни для узла с идентификатором, заданным параметром Р, независимо от их количества, будет выглядеть следующим образом:

SELECT t.\*

FROM test1 t

JOIN RecTest1(:P, 0) r ON t.id = r.id

Многие сервера баз данных имеют ограничение на возможное количество рекурсивных вызовов процедуры. Например, для сервера Firebird это ограничение составляет порядка одной тысячи.

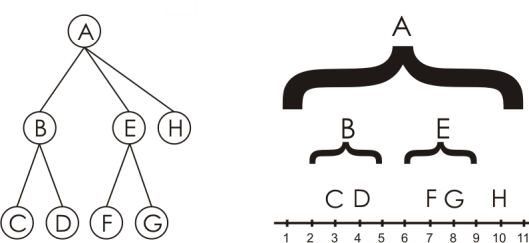
#### Интервальное дерево

Выше мы пытались работать с древовидной структурой, как с графом. Проблемы, которые у нас возникали: использование рекурсии, дополнительной таблицы или громоздкого строкового поля были вызваны тем, что язык SQL по своей природе, это язык предназначенный для работы со множествами, а не с графами.

Для того, чтобы представить древовидную структуру ввиде вложенных множеств43 необходимо для каждого узла ввести два целочисленных параметра: левую и правую границу, так чтобы выполнялись следующие условия:

1. Правая граница узла больше, либо равна левой;
2. Левые границы потомков больше чем левая граница родительского узла;
3. Правые границы потомков меньше, либо равны правой границе родительского узла;
4. Интервалы, определенные левыми и правыми границами узлов, имеющих одного родителя, не должны пересекаться.

Следующий рисунок призван проиллюстрировать представление дерева ввиде вложенных множеств:



*Рис. 97. Представление дерева в виде вложенных множеств*

Для хранения интервальной древовидной структуры создадим следующую таблицу:

CREATE TABLE test4 (

id INTEGER NOT NULL,

parent INTEGER,

lb INTEGER NOT NULL,

rb INTEGER NOT NULL,

43 Представление деревьев ввиде вложенных множеств подробно рассмотрено Joe Celko в журнале [DBMS Online.](http://www.dbmsmag.com/9603d06.html) Стоит заметить, что в настоящей статье предложен вариант организации интервальных деревьев, когда связь родитель-потомок задается в первую очередь через отношение полей id-parent. Границы интервалов, поля lb и rb, имееют скорее вспомогательную роль и используются при необходимости организовать нерекурсивный обход дерева. В варианте предложенном Joe Celko, границы интервалов, и только они, определяют иерархию узлов в дереве.

PRIMARY KEY (id),

FOREIGN KEY (parent) REFERENCES test4 (id) ON UPDATE CASCADE

ON DELETE CASCADE,

CHECK (lb <= rb)

);

CREATE INDEX test4\_x\_lb ON test4 (lb);

CREATE DESCENDING INDEX test4\_x\_rb ON test4 (rb);

Запрос, извлекающий все узлы поддерева вершина которого имеет идентификатор P, будет выглядеть следующим образом:

SELECT t.\*

FROM test4 t JOIN test4 t2

ON t.lb > t2.lb AND t.rb <= t2.rb WHERE

t2.id = :P

Заменив строгое неравенство по левой границе на не строгое мы добавим саму вершину поддерева в результирующую выборку.

По сравнению с ранее рассмотренными структурами, интервальное дерево требует наибольших усилий при добавлении нового узла или при переносе узла из одной ветви в другую. Помимо двух триггеров нам понадобятся еще две хранимые процедуры. Рассмотрим их подробнее:

CREATE TRIGGER test4\_bi FOR test4 BEFORE INSERT

POSITION 32000 AS

BEGIN

/\* проверим: добавляется корень нового дерева

/\* или потомок к существующему узлу?

IF (NEW.parent IS NULL) THEN BEGIN

/\* если добавляется новое дерево, то

\*/

\*/

\*/

/\* расположим его левую границу за максимальной, \*/

/\* известной нам, правой границей из существующих\*/

/\* записей в таблице. \*/

NEW.lb = NULL;

SELECT MAX(rb) FROM test4 INTO NEW.lb;

IF (NEW.lb IS NULL) THEN

/\* добавляется самый первый элемент в таблицу \*/

/\* в нашем примере, мы используем целые числа \*/

/\* >0 для хранения границ интервалов. \*/

NEW.lb = 1; ELSE

NEW.lb = NEW.lb + 1; END ELSE

BEGIN

/\* вызовем процедуру, которая отыщет возможную \*/

/\* левую границу для потомка. Если интервал \*/

/\* родителя исчерпан, то он будет расширен. \*/ EXECUTE PROCEDURE el\_test4 (NEW.parent, 1, -1, -1)

RETURNING\_VALUES NEW.lb;

END

NEW.rb = NEW.lb; END;

/\* Данная процедура отыскивает левую границу \*/

/\* для потомка узла, заданного идентификатором \*/

/\* PARENT. Если интервал родителя исчерпан, то \*/

/\* он будет расширен на величину не меньшую, чем\*/

/\* DELTA. При расширении интервала, будут \*/

/\* расширены все вышележещие интервалы, а также \*/

/\* сдвинуты все интервалы, находящиеся правее. \*/

/\* При сдвиге и расширении из обработки \*/

/\* исключаются интервалы, попадающие в интервал \*/

/\* заданный границами LB2, RB2. Исключение \*/

/\* интервалов помогает избежать зацикливания при\*/

/\* при перемещении поддерева от одного родителя \*/

/\* к другому. \*/

CREATE PROCEDURE EL\_TEST4 ( PARENT INTEGER,

DELTA INTEGER, LB2 INTEGER, RB2 INTEGER)

RETURNS (

LEFTBORDER INTEGER)

AS

DECLARE VARIABLE R INTEGER; DECLARE VARIABLE L INTEGER; DECLARE VARIABLE R2 INTEGER;

DECLARE VARIABLE MKey INTEGER; DECLARE VARIABLE MultiDelta INTEGER;

BEGIN

/\* Получаем границы родителя \*/ SELECT rb, lb

FROM test4

WHERE id = :Parent INTO :R, :L;

/\* Получаем крайнюю правую границу потомков, \*/

/\* родителя, если они существуют \*/

R2 = NULL;

SELECT MAX(rb) FROM test4 WHERE parent = :Parent

/\*lb > :L AND lb <= :R\*/ INTO :R2;

/\* Если потомков нет берем левую границу \*/

/\* родителя \*/

IF (:R2 IS NULL) THEN R2 = :L;

/\* проверяем: хватит ли места для размещения \*/

/\* еще одного потомка в интервале родителя \*/

IF (:R - :R2 < :Delta) THEN BEGIN

/\* Если места не достаточно -- раздвигаем \*/

/\* интервал \*/

MultiDelta = :R - :L + 100;

/\* Проверяем удовлетворяет ли нас новый \*/

/\* диапазон \*/

IF (:Delta > :MultiDelta) THEN MultiDelta = :Delta;

/\* Сдвигаем правую границу родителей \*/

IF (:LB2 > -1) THEN

UPDATE test4 SET rb = rb + :MultiDelta WHERE lb <= :L AND rb >= :R

AND NOT (lb >= :LB2 AND rb <= :RB2); ELSE

UPDATE test4 SET rb = rb + :MultiDelta WHERE lb <= :L AND rb >= :R;

/\* Сдвигаем обе границы интервалов, лежащих\*/

/\* правее, от родительского интервала \*/

/\* FOR

SELECT id FROM test4

WHERE lb > :R AND NOT (lb >= :LB2 AND rb <= :RB2) ORDER BY lb DESC

INTO :MKey DO

UPDATE test4

SET lb = lb + :MultiDelta, rb = rb + :MultiDelta WHERE id = :MKey;

\*/

IF (:LB2 > -1) THEN

UPDATE test4

SET lb = lb + :MultiDelta, rb = rb + :MultiDelta WHERE lb > :R

AND NOT (lb >= :LB2 AND rb <= :RB2);

ELSE

UPDATE test4

SET lb = lb + :MultiDelta, rb = rb + :MultiDelta WHERE lb > :R;

END

/\* возвращаем найденную границу \*/ LeftBorder = :R2 + 1;

END;

/\*

CREATE EXCEPTION tree\_e\_invalid\_parent

‘Invalid parent’;

\*/

CREATE TRIGGER test4\_bu FOR test4 BEFORE UPDATE

POSITION 32000 AS

DECLARE VARIABLE OldDelta INTEGER; DECLARE VARIABLE L INTEGER; DECLARE VARIABLE R INTEGER; DECLARE VARIABLE NewL INTEGER;

BEGIN

/\* Проверяем факт изменения PARENT \*/ IF ((NEW.parent <> OLD.parent)

OR (OLD.parent IS NULL AND NOT NEW.parent IS NULL) OR (NEW.parent IS NULL AND NOT OLD.parent IS NULL))

THEN BEGIN

/\* Делаем проверку на зацикливание \*/

/\*

IF (EXISTS(SELECT \* FROM test4 WHERE id = NEW.parent AND lb >= OLD.lb AND rb <= OLD.rb)) THEN

BEGIN

EXCEPTION tree\_e\_invalid\_parent; END ELSE

BEGIN

\*/

IF (NEW.parent IS NULL) THEN BEGIN

/\* Получаем крайнюю правую границу \*/

SELECT MAX(rb)

FROM test4

WHERE parent IS NULL INTO :NewL;

/\* Предполагается, что существует \*/

/\* хотя бы один элемент с нулл парентом \*/

NewL = :NewL + 1; END ELSE

BEGIN

/\* Получаем значение новой левой границы \*/

EXECUTE PROCEDURE el\_test4 (NEW.parent, OLD.rb - OLD.lb + 1, OLD.lb, OLD.rb)

RETURNING\_VALUES :NewL; END

/\* Определяем величину сдвига. \*/

/\* +1 выполняется в процедуре \*/

OldDelta = :NewL - OLD.lb;

/\* Сдвигаем границы основной ветви \*/

NEW.lb = OLD.lb + :OldDelta; NEW.rb = OLD.rb + :OldDelta;

/\* Сдвигаем границы детей \*/ UPDATE test4

SET lb = lb + :OldDelta, rb = rb + :OldDelta WHERE lb > OLD.lb AND rb <= OLD.rb;

/\* END

\*/ END

END;

В платформе Гедымин применяются как простые, так и интервальные деревья.

# Глава VIII. Бизнес-объект

Платформа Гедымин имеет множество составляющих: подсистема выполнения программного кода, среда быстрой разработки, набор типовых форм и визуальных компонентов для построения пользовательского интерфейса и т.д. Но, бесспорно, если мы попытаемся выделить самое важное, попробуем сформулировать суть и соль Гедымина в одном слове, этим словом будет бизнес-объект. Удивительно, но краеугольный камень платформы не является чем-то уникальным в своем роде. На Рис. 98 хорошо видно, что TgdcBase – базовый класс для всех бизнес-объектов – является… датасетом!

TDataSet

TIBCustomDataSet

TgdcBase

*Рис. 98. Диаграмма наследования класса TgdcBase*

Так почему же мы говорим, что бизнес-объект является основой платформы? Рассмотрим функцию AddCompany, которая заносит в базу данных запись о компании. Имя компании и ее ИНН передаются в качестве входящих параметров. Из материала главы "Структура базы данных" мы помним, что информация о компании хранится в трех таблицах: GD\_CONTACT, GD\_COMPANY и GD\_COMPANYCODE, связанных между собой один-к-одному. Так как наша функция работает с БД напрямую, нам придется выполнить три запроса на вставку записи в соответствующие таблицы:

Option Explicit

Function AddCompany(CompanyName, TaxID)

Dim q, Tr, Creator

Set Creator = New TCreator

Set q = Creator.GetObject(Application, "TIBSQL", "")

Set Tr = Creator.GetObject(Application, \_ "TIBTransaction", "")

AddCompany = gdcBaseManager.GetNextID

Задания для самопроверки 237

Tr.DefaultDatabase = gdcBaseManager.Database Tr.StartTransaction

q.Transaction = Tr

q.SQL.Text = "INSERT INTO gd\_contact " &\_ "(id, parent, contacttype, name) " &\_

"VALUES (:id, :parent, :contacttype, :name) " q.ParamByName("id").AsInteger = AddCompany q.ParamByName("parent").AsInteger = \_

gdcBaseManager.GetIDByRUIDString("147002208\_31587988") q.ParamByName("contacttype").AsInteger = 3 q.ParamByName("name").AsString = CompanyName

q.ExecQuery

q.SQL.Text = "INSERT INTO gd\_company " &\_

"(contactkey, fullname) VALUES (:contactkey, :fullname)" q.ParamByName("contactkey").AsInteger = AddCompany q.ParamByName("fullname").AsString = CompanyName q.ExecQuery

q.SQL.Text = "INSERT INTO gd\_companycode " &\_ "(companykey, taxid) VALUES (:companykey, :taxid) "

q.ParamByName("companykey").AsInteger = AddCompany q.ParamByName("taxid").AsString = TaxID q.ExecQuery

Tr.Commit

End Function

Не правда ли, сравнительно простая операция добавления в базу данных компании потребовала 24-х строк кода (не учитывая пустые строки, переносы и коментарии). Проделаем теперь тоже самое, но с использование бизнес объекта TgdcCompany:

Option Explicit

Function AddCompany2(CompanyName, TaxID)

Dim Creator, Obj

Set Creator = New TCreator

Set Obj = Creator.GetObject(Application, "TgdcCompany",\_ "")

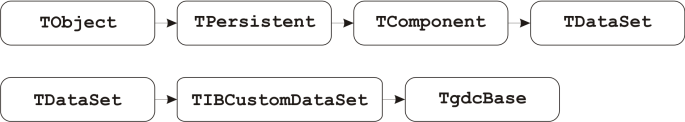
Obj.Open Obj.Insert

Obj.FieldByName("parent").AsInteger = \_ gdcBaseManager.GetIDByRUIDString("147002208\_31587988")

Obj.FieldByName("name").AsString = CompanyName Obj.FieldByName("taxid").AsString = "123456789" Obj.Post

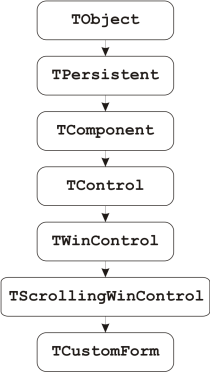
AddCompany2 = Obj.ID End Function

Всего тринадцать строк кода. Почти в два раза короче!

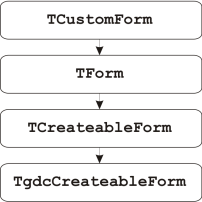


*Рис. 99. Иерархия наследования класса TgdcBase*

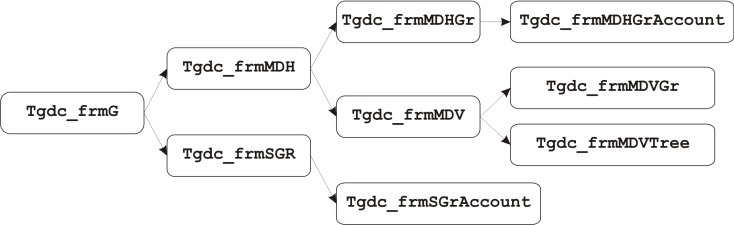
# Глава IX. Экранные формы



*Рис. 100. Иерархия классов*



*Рис. 101. Иерархия классов*



*Рис. 102. Классы форм просмотра*



*Рис. 103. Классы диалоговых окон*

# Глава X. Документ

Любое событие в хозяйственной жизни предприятия, которое имеет интерес с точки зрения бухгалтерского или складского учета, отражается в базе данных с помощью соответствующего документа. Приведем несколько примеров:

 Платежное поручение – отражает факт передачи распоряжения банку на перевод денег с расчетного счета;

 Товарная накладная – фиксирует перемещение и/или передачу прав на товарно-материальные ценности;

 Договор на оказание услуг – определяет сроки и условия взаимоотношений между предприятием и клиентом.

Все документы подразделяются на три больших класса:

 **Документ пользователя** – фиксирует некоторый факт, может приводить к формированию бухгалтерских проводок;

 **Складской документ** – в дополнение к предыдущему формирует складское движение и заносит его в соответствующие таблицы;

 **Прайс-лист** – документирует факт установления отпускных цен на продукцию/товары предприятия. В отличие от других документов не приводит к формированию проводок и складского движения.

Документ может быть реализован на уровне платформы. Код бизнес- объекта такого документа написан на Делфи и откомпилирован в составе выполняемого файла gedemin.exe, а необходимые таблицы находятся в эталонной базе данных. На сегодняшний день, такой подход практически не применяется ввиду его недостаточной гибкости. Как правило, документ создается настройщиком и поставляется в составе прикладного решения (настройки)44. Из встроенных документов сохранились только банковская выписка и картотека.

#### Структура метаданных

По структуре документы можно подразделить на простые (однопозиционные) и сложные (многопозиционные). Простой документ состоит из одной записи. Сложный – имеет шапку (заголовок) и позиции. В общем случае число позиций может принимать значение от 0 (пустой

44 Само название – Документ пользователя – говорит нам о том, что когда-то разграничение на встроенные документы (т.е. создаваемые программистом) и документы, создаваемые настройщиком (т.е. пользователем платформы) имело принципиальное значение.

документ) и до того количества, которое ограничено только техническими спецификациями сервера и платформы.

Каждый документ и позиция сложного документа имеют общую часть, которая хранится в таблице GD\_DOCUMENT, и специфические данные, которые находятся в отдельной таблице – таблице документа или таблице позиций документа. Между ними устанавливается связь один-к-одному. Первичный ключ таблицы документа или позиций документа всегда называется documentkey и одновременно является ссылкой на GD\_DOCUMENT.

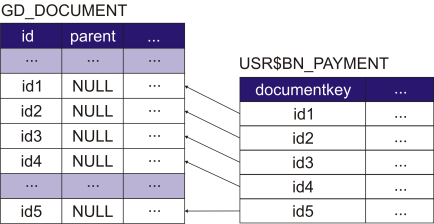
##### GD\_DOCUMENT

Таблица имеет древовидную структуру для хранения связей между шапками и позициями сложных документов. Сам документ не может входить в другой документ. Таким образом максимальная глубина дерева – два уровня. Ниже дано подробное описание каждого поля:

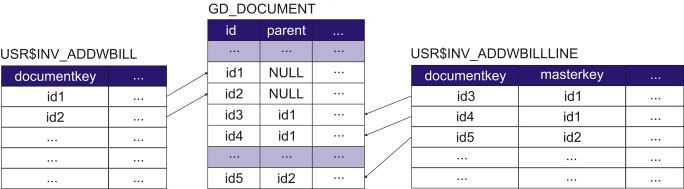
|  |  |
| --- | --- |
| **Поле, Тип данных** | **Описание** |
| ID, DINTKEY | Идентификатор документа. |
| PARENT, DFOREIGNKEY | Для шапки документа содержит NULL,  для позиции – ссылку на шапку. |
| DOCUMENTTYPEKEY,  DINTKEY | Ссылка на тип документа  (GD\_DOCUMENTTYPE). |
| TRTYPEKEY | В настоящее время не используется. |
| TRANSACTIONKEY,  DFOREIGNKEY | Ссылка на типовую хозяйственную  операцию (AC\_TRANSACTION). |
| NUMBER, DDOCUMENTNUMBER | Номер документа. Максимальная длина 20 символов. Поле обязательно для  заполнения. |
| DOCUMENTDATE,  DDOCUMENTDATE | Дата документа. Поле обязательно для  заполнения. |
| DESCRIPTION | Произвольный комментарий. |
| SUMNCU, DCURRENCY | Сумма в НДЕ. Используется по  усмотрению разработчика. Не обязательно к заполнению. |
| SUMCURR, DCURRENCY | Сумма в валюте. Используется по  усмотрению разработчика. |
| SUMEQ, DCURRENCY | Сумма в эквиваленте. Используется по  усмотрению разработчика. |
| DELAYED, DBOOLEAN | Флаг отложенного документа. Если  установлен, то не будут формироваться |

|  |  |
| --- | --- |
|  | проводки и складское движение при  сохранении документа в базу данных. |
| AFULL, DSECURITY | Дескриптор полного доступа. |
| ACHAG, DSECURITY | Дескриптор доступа на изменение. |
| AVIEW, DSECURITY | Дескриптор доступа на просмотр. |
| CURRKEY, DFOREIGNKEY | Ссылка на валюту, в которой выражено  значение SUMCURR. |
| COMPANYKEY, DINTKEY | Ссылка на рабочую организацию, к  которой относится данный документ. |
| CREATORKEY, DINTKEY | Ссылка на пользователя в таблице  GD\_CONTACT, который создал документ. |
| CREATIONDATE,  DCREATIONDATE | Дата, когда документ был помещен в базу  данных. |
| EDITORKEY, DINTKEY | Ссылка на пользователя в таблице GD\_CONTACT, который последним  вносил изменения в документ. |
| EDITIONDATE,  DEDITIONDATE | Дата последнего редактирования  документа. |
| PRINTDATE, DDATE | Дата, когда документ был последний раз  открыт для просмотра перед печатью. |
| DISABLED, DDISABLED | Признак удаленной записи. |
| RESERVED, DRESERVED | Зарезервировано для будущего  использования. |

Первичный ключ таблицы документа одновременно является внешней ссылкой на GD\_DOCUMENT.



*Рис. 104. Структура данных простого документа*

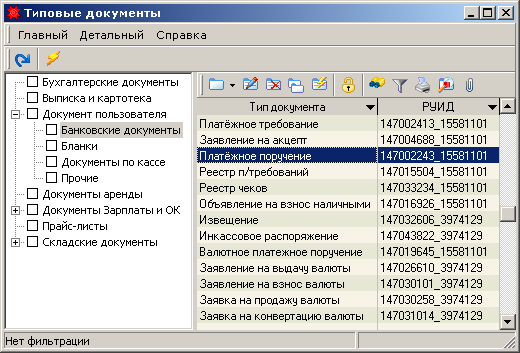


*Рис. 105. Структура данных сложного документа*

## Тип документа

Выше мы уже отмечали, что все документы платформы Гедымин подразделяются на три класса: Документы пользователя, Складские документы и Прайс-листы. В рамках каждого класса настройщик может создавать свои типы документов.

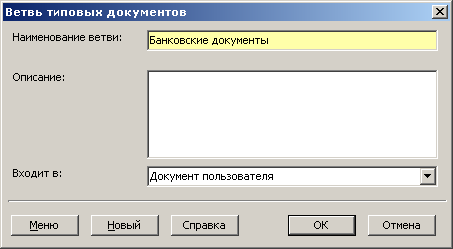
Для создания нового типа или изменения параметров существующего следует открыть окно **Типовые документы** из раздела Сервис Исследователя системы.



*Рис. 106. Окно Типовые документы*

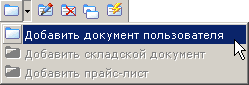
Вид окна показан на Рис. 106. В левой части находится дерево папок, в правой – список типов документов из выбранной папки. **Складские документы** могут располагаться только в одноименной корневой папке и ее вложенных папках. Аналогично, корневая папка **Прайс-листы** служит для размещения документов типа Прайс-лист. Остальная конфигурация дерева папок, включая вложенные уровни, произвольна и зависит от личных предпочтений разработчика. Как правило, типовые документы из одной настройки группируются в одну или несколько вложенных папок.

Команды создания, изменения и удаления выбранной папки вызываются из меню окна или по правой кнопке мыши. Диалоговое окно редактирования папки показано ниже.



*Рис. 107. Диалоговое окно редактирования папки Типовых документов*

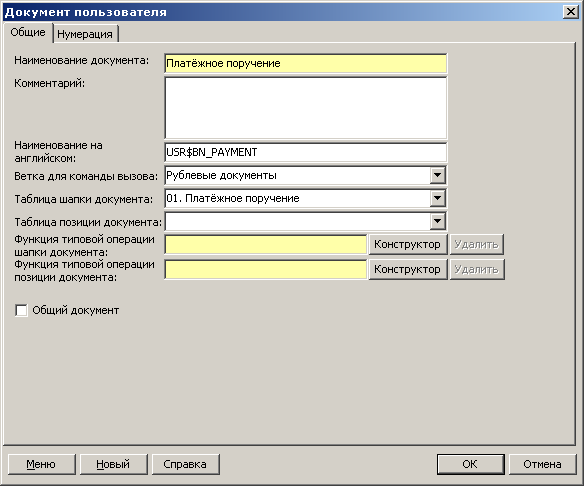
Обратимся теперь к правой части окна Типовые документы. На Рис. 108 паказано меню на панели инструментов с командами создания новых типов.



*Рис. 108. Команды создания типовых документов*

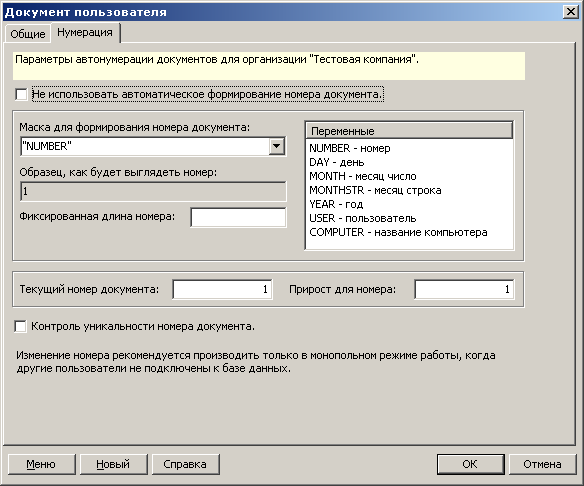
Следует заметить, что в зависимости от того, какая папка выбрана в данный момент, будет доступна только одна из трех команд: Добавить документ пользователя, Добавить складской документ или Добавить прайс- лист.

#### Создание документа пользователя

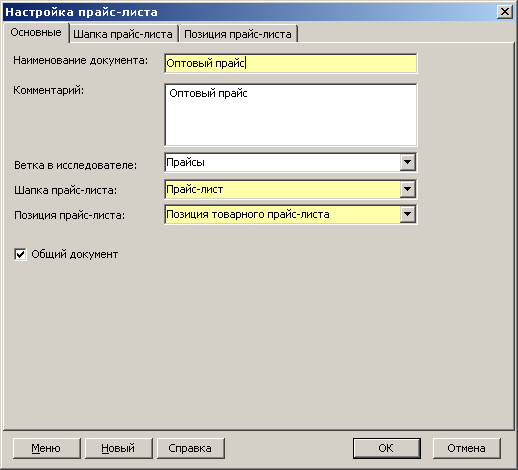


*Рис. 109. Документ пользователя. Вкладка Общие*

|  |  |
| --- | --- |
| Наименование  документа |  |
| Комментарий |  |
| Наименование на  английском |  |
| Ветка для команды  вызова |  |
| Таблица шапки  документа |  |
| Таблица позиции  документа |  |
| Функция типовой операции шапки  документа |  |
| Функция типовой  операции позиции документа |  |
| Общий документ |  |



*Рис. 110. Документ пользователя. Вкладка Нумерация*



*Рис. 111. Прайс-лист*

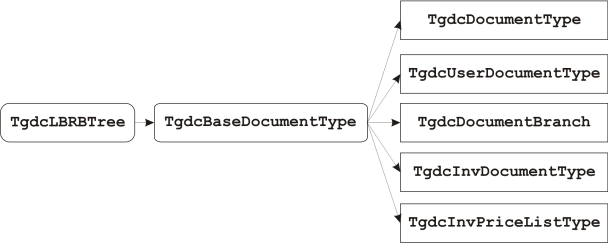
##### GD\_DOCUMENTTYPE

|  |  |
| --- | --- |
| Поле, Тип данных | Описание |
| ID, DINTKEY | Идентификатор типа документа. |
| PARENT, DFOREIGNKEY |  |
| LB, DLB |  |
| RB, DRB |  |
| NAME, DNAME | Наименование типа документа или |

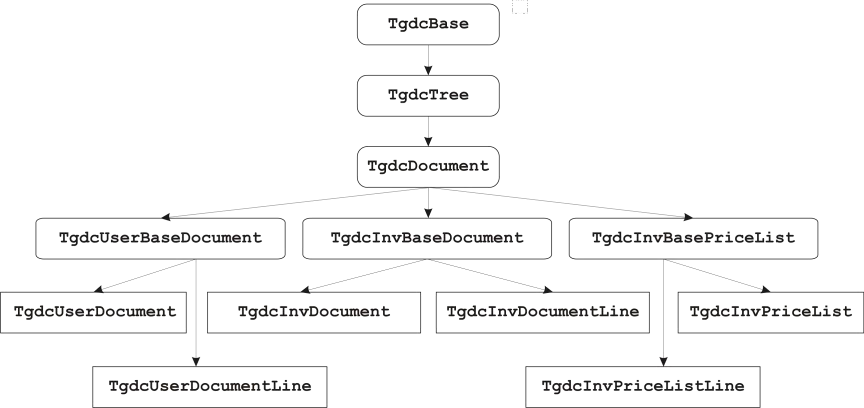
|  |  |
| --- | --- |
|  | папки. |
| DESCRIPTION, DTEXT180 | Произвольное текстовое описание. |
| CLASSNAME, DCLASSNAME | Имя Делфи класса типа документа. Может принимать следющие значения: 'TgdcUserDocumentType', 'TgdcInvDocumentType' или  'TgdcInvPriceListType'. |
| DOCUMENTTYPE,  DDOCUMENTTYPE | Тип записи: 'D' – тип документа, 'B' –  папка. |
| OPTIONS, DBLOB1024 | Параметры типа документа в двоичном  формате. |
| HEADERRELKEY,  DFOREIGNKEY | Таблица шапки документа. Ссылка на  таблицу AT\_RELATIONS. |
| LINERELKEY, DFOREIGNKEY | Таблица позиций документа. Ссылка на  AT\_RELATIONS. |
| AFULL, DSECURITY |  |
| ACHAG, DSECURITY |  |
| AVIEW, DSECURITY |  |
| DISABLED, DDISABLED |  |
| RUID, DRUID | РУИД записи, который используется в  качестве идентификатора типа документа, независимого от базы данных. |
| BRANCHKEY, DFOREIGNKEY |  |
| REPORTGROUPKEY,  DFOREIGNKEY |  |
| RESERVED, DRESERVED |  |
| ISCOMMON, BOOLEAN | Признак общего документа. Если установлен, то при просмотре списка будут выводиться все записи. В противном случае, в списке будут отображаться только документы, относящиеся к текущей рабочей  организации45. |
| ISCHECKNUMBER, | Признак определяет: контролировать |

45 Пример общего документа -- прайс лист. Все прайс листы будут видны при просмотре под любой из рабочих организаций. Не общий документ --: платежное поручение. При работе под конкрентой рабочей организацией будут отображаться платежные поручения только этой организации.

|  |  |
| --- | --- |
| DBOOLEAN | уникальность номера для данного  документа или нет. |
| HEADERFUNCTIONKEY, DFOREIGNKEY | Ссылка на скрипт-функцию, выбирающую типовую хозяйственную  операцию для шапки документа. |
| HEADERFUNCTIONTEMPLA  TE, DBLOB80 |  |
| LINEFUNCTIONKEY |  |
| LINEFUNCTIONTEMPLATE |  |



*Рис. 112. Иерархия классов типов документов*

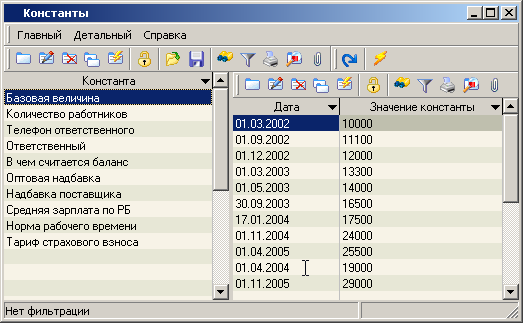


*Рис. 113. Иерархия классов документов*

# Глава XI. Константы

Практически в любой задаче можно выделить некоторые величины, участвующие в расчетах или отображаемые на экране или печатном устройстве. Если такая величина является абсолютной константой, достаточно прописать ее в программном коде, используя директиву Const. Для параметров, изменяющихся во времени и/или зависящих от рабочей организации или текущего пользователя, предусмотрен специальный класс бизнес-объектов – Константа46. Примеры констант: размер базовой величины или бюджета прожиточного минимума, устанавливаемого правительством, адрес налоговой инспекции, который печатается в бланках расчетов по налогам, ставка налога с продаж и т.п.

Список констант можно просмотреть используя команду Сервис- Константы Исследователя системы.

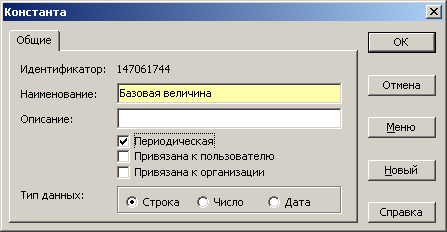


*Рис. 114. Окно со списком констант*

Как видно на Рис. 114 выше, окно разделено на две части. Слева отображается список констант, справа – список значений выбранной константы.

При создании константы (см. Рис. 115) указывается ее наименование, произвольный комментарий, тип константы и тип данных.

46 Возникает естественный вопрос: насколько правомерно называть константой величину, изменяющуюся во времени или в зависимости от выбранной организации и текущего пользователя? В данном случае, термин константа указывает на то, что данное значение инвариантно относительно данных, т.е. записи или группы записей участвующих в расчете.



*Рис. 115. Диалоговое окно редактирования константы*

Тип константы задается тремя флагами:

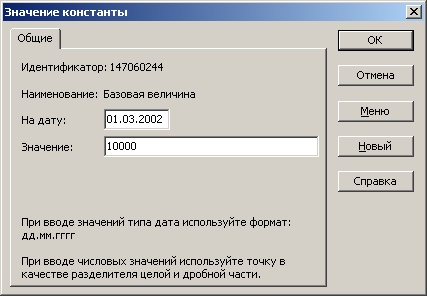
 **Периодическая** – может иметь несколько значений, заданных на разные даты. На Рис. 114 хорошо видно, как размер базовой величины изменялся со временем;

 **Привязана к пользователю** – значения константы различаются для каждого пользователя системы;

 **Привязана к организации** – значения константы различаются для каждой рабочей организации.

Обратите внимание, что указанные флаги не являются взаимоисключающими. Т.е. константа может быть периодической и привязанной к организации одновременно.

Ввод значения константы осуществляется с помощью диалогового окна, показанного на Рис. 116 ниже.



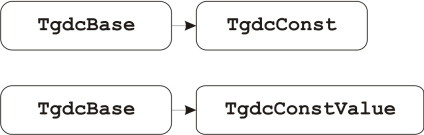
*Рис. 116. Диалоговое окно редактирования значения константы*

В не зависимости от типа данных константы, ее значения в базе хранятся в строковом представлении. Поэтому, при вводе следует строго придерживаться формата "дд.мм.гггг" для дат и использовать точку в

качестве разделителя целой и дробной части вещественных чисел. Максимальная длина значения – 120 символов.

#### Бизнес-классы констант

Для работы с константами и их значениями предусмотрены два бизнес- класса, наследованных от TgdcBase (см. Рис. 117).



*Рис. 117. Бизнес-классы констант*

#### TgdcConst

##### IsCompany

Возвращает значение Истина для константы, привязанной к рабочей организации.

##### function IsCompany: Boolean

**IsPeriod**

Возвращает значение Истина для периодической константы.

##### function IsPeriod: Boolean

**IsUser**

Возвращает значение Истина для константы, привязанной к пользователю.

##### function IsUser: Boolean

**QGetValueByID**

Возвращает значение константы по ее идентификатору. **function QGetValueByID(AnID: Integer): Variant** AnID Идентификатор константы.

Для периодических констант, констант, зависящих от пользователя и/или рабочей организации, используются: текущая дата, идентификатор текущего пользователя и текущей рабочей организации, соответственно.

Если константа с указанным идентификатором не существует, генерируется исключение. Если не существует значения константы для заданной коомбинации даты, идентификатора пользователя и/или рабочей организации – функция вернет Unassigned.

##### QGetValueByID2

Возвращает значение константы по ее идентификатору, дате действия, идентификаторам пользователя и рабочей организации.

##### function QGetValueByID2(AnID: Integer; ADate: TDateTime; AUserKey: Integer; ACompanyKey: Integer): Variant

AnID Идентификатор константы.

ADate Дата, на которую следует извлечь значение периодической константы. 0 означает текущую системную дату.

AUserKey Идентификатор пользователя для константы, привязанной к пользователю системы. -1 означает идентификатор текущего пользователя системы.

ACompanyKey Идентификатор компании, для константы привязанной к рабочей организации. -1 означает идентификатор текущей рабочей организации.

Если константа с указанным идентификатором не существует, генерируется исключение. Если не существует значения константы для заданной коомбинации даты, идентификатора пользователя и/или рабочей организации – функция вернет Unassigned.

##### QGetValueByIDAndDate

Возвращает значение константы по ее идентификатору и дате действия.

##### function QGetValueByIDAndDate(AnID: Integer; ADate: TDateTime): Variant

AnID Идентификатор константы.

ADate Дата, на которую следует извлечь значение периодической константы. 0 означает текущую системную дату.

Если константа с указанным идентификатором не существует, генерируется исключение. Для констант, зависящих от пользователя и/или рабочей организации, используются идентификаторы текущего

пользователя системы и текущей рабочей организации, соответственно. Если не существует значения константы для заданной коомбинации даты, идентификатора пользователя и/или рабочей организации – функция вернет Unassigned.

##### QGetValueByName

Возвращает значение константы по ее имени.

##### function QGetValueByName(AName: String): Variant

AName Наименование константы. Не чувствительно к регистру.

Функция ведет себя аналогично QGetValueByID, за тем исключением, что поиск константы осуществляется по ее имени, а не идентификатору.

##### QGetValueByName2

Возвращает значение константы по ее имени, дате действия, идентификаторам пользователя и текущей рабочей организации.

##### function QGetValueByName2(AName: String; ADate: TDateTime; AUserKey: Integer; ACompanyKey: Integer): Variant

AName Наименование константы. Не чувствительно к регистру.

ADate Дата, на которую следует извлечь значение периодической константы. 0 означает текущую системную дату.

AUserKey Идентификатор пользователя для константы, привязанной к пользователю системы. -1 означает идентификатор текущего пользователя системы.

ACompanyKey Идентификатор компании, для константы привязанной к рабочей организации. -1 означает идентификатор текущей рабочей организации.

Функция ведет себя аналогично QGetValueByID2, за тем исключением, что поиск константы осуществляется по ее имени, а не идентификатору.

##### QGetValueByNameAndDate

Возвращает значение константы по ее имени и дате действия.

##### function QGetValueByName(AName: String; ADate TDateTime): Variant

AName Наименование константы. Не чувствительно к регистру.

ADate Дата, на которую следует извлечь значение периодической константы. 0 означает текущую системную дату.

Функция ведет себя аналогично QGetValueByIDAndDate, за тем исключением, что поиск константы осуществляется по ее имени, а не идентификатору.

#### TgdcConstValue

Бизнес-класс не предоставляет никаких дополнительных свойств или методов.

#### Обращение к константе из макросов

Считывание значения константы из макросов осуществляется, как правило, по ее RUID-у. Ниже приведен пример кода определения телефона ответственного лица для последующего вывода в бланке налоговой декларации:

...

Const cst\_gs\_ResponsibleTel = "147009658\_116121635" Dim Creator, gdcConst, CurCompanyResponsibleTel Set Creator = New TCreator

Set gdcConst = Creator.GetObject(nil, "TgdcConst", "")

CurCompanyResponsibleTel = gdcConst.QGetValueByIDAndDate(\_ gdcBaseManager.GetIdByRUIDString(cst\_gs\_ResponsibleTel),\_ Date)

...

## Заключение

Использование констант позволяет устранить зависимость макросов от изменения законодательства или параметров организации.

# Глава XII. Организация бухгалтерского

**учета**

**Бухгалтерский учет** представляет собой особую форму учета, в основу которой положены принципы сплошного, беспрерывного и взаимосвязанного отражения фактов хозяйственной деятельности предприятия в денежном измерении.

Учет хозяйственных средств предприятия и источников их образования осуществляется на счетах бухгалтерского учета.

**Бухгалтерский счет** можно представить в виде таблицы, левая сторона которой называется **дебет** (от латинского слова "debet" – должен), а правая – **кредит** (от латинского слова "credo" – верить). Для каждого вида хозяйственных средств и источников их образования открывается отдельный счет.

Бухгалтерским балансом называется таблица, в левой половине которой отражаются остатки по счетам учета хозяйственных средств предприятия, а в правой -- остатки по счетам учета источников возникновения средств. Счета в левой и правой части баланса группируются по однородным статьям. Например, статья баланса "Производственные запасы" содержит счета материалы, топливо, запасные части, малоценные и быстроизнашивающиеся предметы. Левая часть баланса называется **Актив**, а правая – **Пассив**. В любой момент времени итоговые суммы левой и правой частей баланса должны совпадать.

По принадлежности к стороне баланса счета подразделяются на **Активные**, **Пассивные** и **Активно-пассивные**. Остаток активного счета и суммы на его увеличение записываются по дебету. Уменьшение активного счета отражается по его кредиту. Наоборот, остаток пассивного счета и суммы увеличивающие его записываются по кредиту, а уменьшающие – по дебету. Остаток по счету на начало учетного периода называется **сальдо**. Активно-пассивные счета могут иметь одновременно дебетовое и кредитовое сальдо и показываются как в активе, так и в пассиве баланса.

Сумма записей по дебету или кредиту счета называется **оборотом**. Разница между оборотом по дебету и оборотом по кредиту счета с учетом остатка на начало составит сальдо на конец заданного периода.

Хозяйственная операция документируется **бухгалтерской проводкой**. При этом соблюдается принцип **двойной записи**, т.е. сумма проводки фиксируется по кредиту одного счета и дебету другого. Взаимосвязь счетов в рамках проводки называется **корреспонденцией счетов**. **Простая проводка** имеет по одному корреспондирующему счету по дебету и кредиту. В **сложной проводке** одному дебету могут сооветствовать кредитовые части нескольких счетов или, наоборот, одному кредиту – несколько дебетов.

По отношению к участвующим в проводке счетам различают четыре типа хозяйственных операций. Операции первого типа приводят к изменениям только в активе баланса. Одна статья увеличивается за счет другой, при этом итоговая сумма остается неизменной. Аналогичным образом операции второго типа затрагивают только пассивные счета. Операции третьего типа приводят к увеличиению счета в левой части баланса и одновременному учеличению счета в правой части на такую же сумму. И, наконец, операции четвертого типа приводят к синхронному уменьшению актива и пассива баланса.

**Синтетический учет** осуществляется по счетам бухгалтерского учета в обобщенном виде в денежном выражении. Детальный, **аналитический учет** осуществляется в рамках синтетического счета в разрезе субсчетов или отдельных объектов. **Количественный учет** позволяет оценить состояние счета не только в денежном, но и в натуральном измерении.

## Бухучет в рамках платформы Гедымин

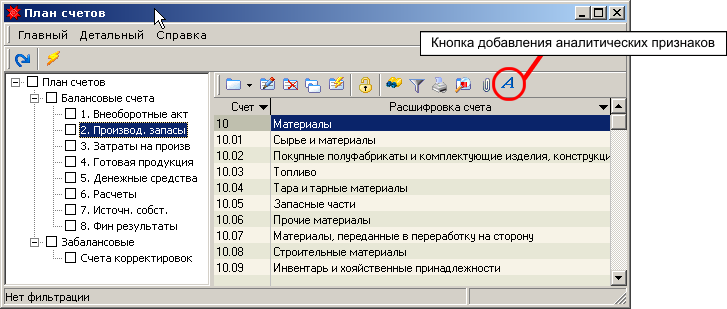
Как инструмент в первую очередь предназначенный для решения экономических задач, платформа Гедымин обладает необходимыми структурами данных для хранения бухгалтерской информации, а также встроенными механизмами формирования проводок и получения специфической отчетности.

В этой главе мы будем рассматривать команды **Исследователя системы**, расположенные в разделе **Бухгалтерия**. Знакомство с подсистемой мы начнем со справочника **План счетов**.

#### План счетов

**План счетов бухгалтерского учета** представляет собой систематизированный перечень счетов бухгалтерского учета. В рамках одной базы данных учет может вестись одновременно по нескольким планам счетов. Например, по плану счетов утвержденному национальным Министерством экономики и финансов и по международному плану счетов, в соответствии с рекомендациям GAAP.

Окно справочника вызывается из раздела **Бухгалтерия** Исследователя системы.



*Рис. 118. Окно с планом счетов*

Как видно на Рис. 118 окно разделено на две части. Слева – древовидный список планов счетов и их разделов, а справа – перечень счетов из выбранного раздела. Как правило, в рамках одного плана счетов, счета сначала подразделяются на **Балансовые** и **Забалансовые**. Последние -- не отображаются в балансе предприятия и не подчиняются принципу двойной записи.

Обратите внимание, что группы **Балансовые счета** и **Забалансовые счета**, приведенные на Рис. 118 служат лишь для более удобного представления информации на экране. Признак забалансового счета устанавливается в диалоговом окне редактирования счета, которое подробно рассматривается ниже.

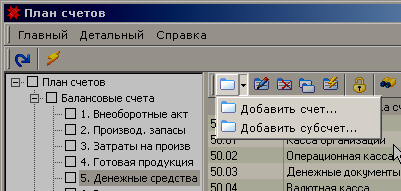
В рамках группы **Балансовые счета** записи группируются в соответствии со статьями баланса предприятия. Еще раз акцентируем внимание на том, что группировка счетов по разделам плана счетов не несет никакой семантической нагрузки и используется лишь для облегчения визуального восприятия. Иными словами, вовсе не обязательно, что счета вложенного раздела будут являться субсчетами счетов родительского раздела плана счетов.

Для создания плана счетов или раздела, редактирования или удаления, используйте команды из контекстного меню.



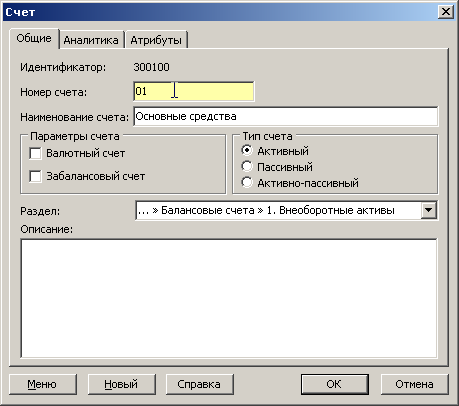
*Рис. 119. Команды создания плана счетов и раздела в контекстном меню*

Для создания счета или субсчета в рамках выбранного раздела следует воспользоваться кнопкой на панели инструментов (см. Рис. 120) или командой из контекстного меню.



*Рис. 120. Команды создания счета и субсчета*

При этом на экране откроется диалоговое окно, как показано на Рис. 121.



*Рис. 121. Диалоговое окно редактирования бухгалтерского счета*

Ниже приведено описание полей для вкладки Общие:

|  |  |
| --- | --- |
| Идентификатор | Идентификатор записи в базе данных. |
| Номер счета | Номер счета (субсчета). Может содержать как цифры, так и буквы. Пользователь вправе использовать любой символ для разделения номера счета и субсчета. Обычно, используется точка. Например, номер "01.10" задает десятый субсчет к счету 01. Количество уровней вложенных субсчетов ограничено только длиной строки номера – 40 символов. Таким образом, при использовании двухзначных номеров для счетов и субсчетов, глубина вложенности может  достигать тринадцати (!) уровней. |
| Наименование счета | Общепризнанное наименование счета. |
| Валютный счет | Признак валютного счета. Если установлен, то |

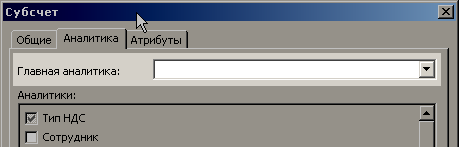
|  |  |
| --- | --- |
|  | при вводе ручной проводки, кроме суммы в НДЕ пользователь должен указать также валюту и  сумму в валюте. |
| Забалансовый счет | Признак забалансового счета. Как уже отмечалось выше, для забалансовых счетов система не контролирует выполнение правила  двойной записи. |
| Тип счета | Выбор типа счета: активный, пассивный или  активно-пассивный. |
| Раздел | Раздел плана счетов, к которому относится  редактируемый счет. |
| Описание | Произвольное описание. |

На вкладке **Аналитика** (см. Рис. 122) отмечаются те аналитические признаки, которые будут использоваться при учете по данному счету, а также настраивается количественный учет. Более подробно организация аналитического и количественного учета в Гедымине будет рассмотрена в следующих разделах.



*Рис. 122. Окно редактирования бухгалтерского счета. Вкладка Аналитика*

Обратите внимание, что для активно-пассивных счетов на вкладке Аналитика появляется дополнительный выпадающий список (см. Рис. 123) для выбора главной аналитики, которая используется при построении журнала-ордера по счету из оборотной ведомости.



*Рис. 123. Выбор главной аналитики активно-пассивного счета*

Вкладка **Атрибуты** окна редактирования бухгалтерского счета не заполнена по умолчанию и практически никогда не используется.

#### Настройка аналитического учета

Бухгалерская проводка может быть детализирована с помощью аналитических признаков. Количество признаков ограничено только возможностями сервера баз данных и может достигать нескольких сотен, что многократно перекрывает потребности задач, встречающихся на практике.

На каждый аналитический признак в базе данных создаются два одноименных поля. Одно – в таблице AC\_ENTRY для хранения значения признака; и, второе – в таблице AC\_ACCOUNT для хранения флага использования данного признака по конкретному счету.

Платформа Гедымин позволяет использовать аналитические признаки любого типа, который может быть создан в разделе Исследователя Сервис- Атрибуты-Домены, за исключением множеств и перечислений.

Обратите внимание, что тип данных аналитического признака не может содержать ограничение NOT NULL.

Благодаря тому, что поле признака добавляется в таблицу AC\_ENTRY, где хранятся все проводки, мы имеем возможность использовать один и тот же признак для разных счетов. Как правило, пакет бухгалтерского учета уже содержит такие основные аналитические признаки, как Клиент, Сотрудник, ТМЦ, Договор и т.п. При конфигурировании плана счетов нам достаточно указать какие признаки будут использоваться для каждого счета, как показано на Рис. 122 выше.

##### Создание аналитического признака

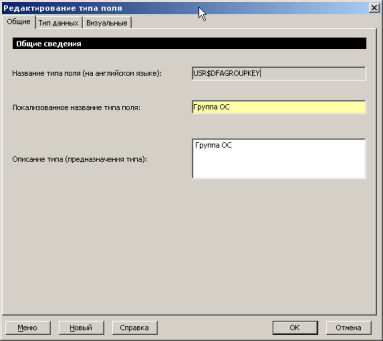
Если необходимый нам признак отсутствует, его можно добавить самостоятельно. Рассмотрим следующую задачу: стандартная настройка бухгалтерского учета устанавливает по счету "01. Основные средства" три аналитических признака:

1. Сотрудник – материально ответственное лицо;
2. Подразделение – где числится основное средство;
3. ТМЦ – объект из справочника товаров, репрезентующий основное средство.

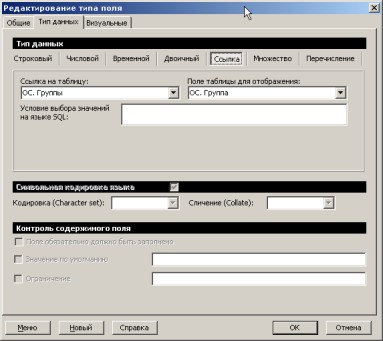
Пакет "Основные средства" содержит справочник-классификатор групп основных средств. Собственно задача: добавить в план счетов признак "Группа ОС" и задействовать его для счета 01.

Решение:

1. Перейдем в раздел Сервис-Атрибуты-Домены Исследователя системы и создадим тип данных ссылка на объект справочника групп основных средств. Заполняйте поля значениями, как показано на Рис. 124 и Рис. 125 ниже.

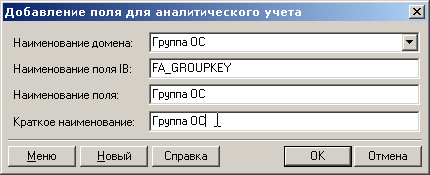


*Рис. 124. Создание типа данных для аналитического признака*



*Рис. 125. Параметры типа данных для аналитического признака*

1. Откроем окно справочника планов счетов и воспользуемся кнопкой добавления поля аналитического признака  на панели инструментов (см. Рис. 118). Заполним поля в появившемся диалоговом окне, как показано на Рис. 126.



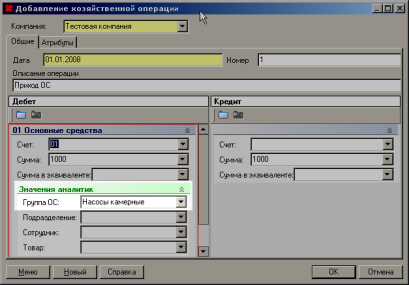
*Рис. 126. Добавление поля аналитического признака*

Описание полей диалогового окна приводится ниже:

|  |  |
| --- | --- |
| Наименование домена | Тип поля аналитического признака. Выбирается  из списка. |
| Наименование поля IB | Наименование полей, которые будут добавлены в таблицы AC\_ENTRY и AC\_ACCOUNT. Префикс  USR$ будет прибавлен автоматически. |
| Наименование поля | Локализованное наименование поля для  отображения на экране. |
| Краткое наименование | Краткое локализованное наименование поля для  отображения в списке аналитик. |

1. После нажатия кнопки Ок будет сформирован и выполнен скрипт с SQL командами создания метаданных. Так как речь идет о создании поля- ссылки необходимо **перезагрузить** Гедымин.
2. После перезагрузки заходим в справочник **План счетов**. Открываем на редактирование счет 01 и на вкладке Аналитика устанавливаем галочку напротив позиции "Группа ОС".

С этого момента, при вводе проводки по счету у нас появится возможность указать группу основного средства.



*Рис. 127. Заполнение аналитического признака при вводе проводки по счету 01*

Так как разработчики платформы не предусмотрели специального бизнес- объекта для аналитического признака, при добавлении его в настройку придется выполнить следующие действия: 1) добавить в настройку домен; 2) добавить в настройку поле из таблицы AC\_ACCOUNT; 3) добавить в настройку поле из таблицы AC\_ENTRY.

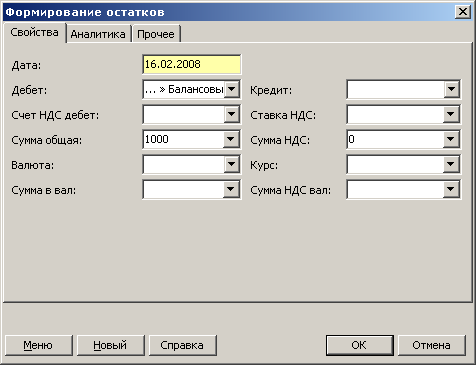
#### Настройка количественного учета

В рамках одного счета можно задать несколько измерений для ведения учета не только в денежном, но и в натуральном выражении. Количественные признаки настраиваются в диалоговом окне редактирования счета, на нижней половине вкладки Аналитика (см. Рис. 122). Если необходимая единица измерения отсутствует ее необходимо добавить в соответствующий справочник.

#### Ввод начальных остатков

Внедрение программного продукта редко совпадает по времени с началом жизнедеятельности предприятия. Как правило, одним из первых шагов по вводу системы в эксплуатацию является формирование начальных остатков по счетам учета и ввод их в базу.

Команда Формирование остатков в разделе Бухгалтерия Исследователя системы открывает на экране список документов. Остаток по каждому счету и объекту аналитического учета вводится отдельной записью. Диалоговое окно документа показано на Рис. 128.



*Рис. 128. Ввод первоначального сальдо по счету*

|  |  |
| --- | --- |
| Дата | Дата, на которую вносится начальный остаток. |
| Дебет, Кредит | В зависимости от типа счета (активный, пассивный, результатный) сальдо отражается по его дебету или кредиту в корреспонденции со  счетом 00. |
| Сумма общая | Сумма остатка |
| Счет НДС, дебет, Ставка НДС, Сумма НДС | Поля предназначены для внесения остатков по расчетам с поставщиками для корректного формирования книги покупок. В поле **Счет НДС** указывается счет учета НДС по соответствующей операции. В поле **Ставка НДС** вводится ставка налога. На основании введенных значений поле **Сумма НДС** заполнится автоматически. При сохранении документа сформируются проводки  как на общую сумму, так и на сумму НДС. |
| Валюта |  |
| Курс |  |
| Сумма в валюте |  |
| Сумма НДС вал |  |

Остатки по счетам, для которых предусмотрен аналитический учет, следует вносить в разрезе объектов аналитики. Соответствующие признаки заполняются на вкладке Аналитика диалогового окна Формирование остатков.

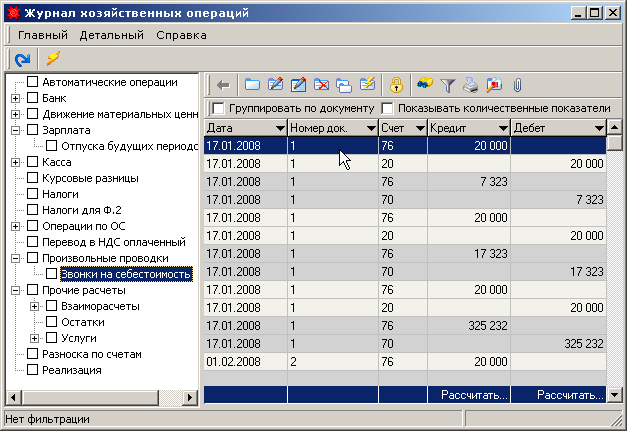
## Журнал хозяйственных операций

Журнал хозяйственных операций (ЖХО) содержит все бухгалтерские проводки. Схема на Рис. 129 показывает три источника возникновения проводок: документы, автоматические хозяйственные операции (АХО) и ручные проводки, которые называются так, поскольку бухгалтер вносит их вручную без привязки к конкретному документу.



*Рис. 129. Схема движения бухгалтерских проводок*

Окно просмотра ЖХО вызывается из раздела Бухгалтерия Исследователя системы командой **Журнал хозяйственных операций**.



*Рис. 130. Журнал хозяйственных операций*

Проводки отображаются в разрезе типовых хозяйственных операций, дерево которых выводится в левой части окна. Так как форма просмотра

ЖХО является обычным окном мастер-дитэйл, перетаскивание разделительной вертикальной линии влево до упора приведет к отображению всего списка проводок без привязки к типовым хозяйственным операциям.

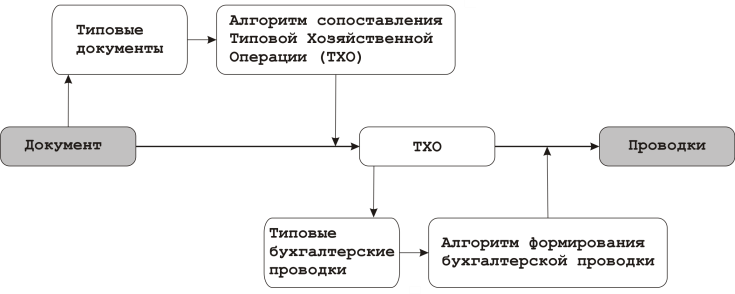
Общее количество проводок на предприятии среднего размера может достигать десятков и сотен тысяч, а на крупном – нескольких миллионов. При работе с ЖХО рекомендуется ограничивать период просмотра с помощью фильтра.

**Ввод ручной проводки**

## Связь между документом и проводками

Коль скоро событие из хозяйственной жизни предприятия регистрируется документом соответствующего типа, а в бухгалтерском учете приводит к формированию проводок, между ними – документом и проводками – должна существовать тесная связь. Создание, изменение или удаление документа должно сопровождаться добавлением, корректировкой или удалением записей из журнала хозяйственных операций.

Схематически механизм формирования проводок, заложенный в платформу Гедымин, можно отобразить следующим образом:



*Рис. 131. Схема формирования бухгалтерских проводок*

Рассмотрим процесс создания нового документа:

1. Тип документа определяется полем DOCUMENTTYPEKEY, которое содержит ссылку на таблицу GD\_DOCUMENTTYPE.
2. С каждым типом документа связан алгоритм, который на основании его данных выбирает одну из типовых хозяйственных операций (ТХО) из

таблицы AC\_TRANSACTION и заносит ее идентификатор в поле TRANSACTIONKEY.

1. С ТХО связана одна или несколько типовых проводок (ТП). Типовые проводки хранятся в таблице AC\_TRRECORD. Поле TRANSACTIONKEY указывает на принадлежность проводки некоторой операции.
2. Каждая ТП содержит алгоритм формирования проводок на основании данных документа. Заголовок проводки находится в таблице AC\_RECORD и среди прочего хранит ссылки на:

TRRECORDKEY Типовую проводку. TRANSACTIONKEY Типовую хозяйственную операцию. DOCUMENTKEY Документ или позицию документа. MASTERDOCKEY Шапку документа.

Если проводка сформирована по шапке документа, то MASTERDOCKEY=DOCUMENTKEY.

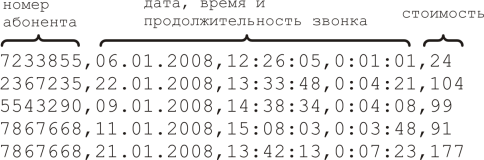
1. Дебетовые и кредитовые части проводки хранятся в таблице AC\_ENTRY. Как и заголовок проводки, ее позиция содержит следующие ссылки на:

TRANSACTIONKEY Типовую хозяйственную операцию. DOCUMENTKEY Документ или позицию документа. MASTERDOCKEY Шапку документа.

Избыточность информации в данном случае позволяет впоследствии создавать оптимальные SQL запросы.

## Практический пример

Рассмотрим следующий пример. Между предприятием и оператором сотовой связи заключен договор, в рамках которого каждому сотруднику выделен телефонный номер. Внутренняя политика предприятия предусматривает оплату звонков на сумму 20 000 рублей в месяц за счет себестоимости. Превышение этой суммы удерживается из заработной платы. Ежемесячно, поставщик услуг связи выставляет предприятию общий счет к оплате и сопровождает его текстовым файлом (см. Рис. 132) с детальной расшифровкой каждого соединения.



*Рис. 132. Структура файла с данными*

Указанная стоимость уже включает НДС по ставке 18% и налог с продаж. Бухгалтерия предприятия использует следующие проводки:

|  |  |  |
| --- | --- | --- |
| Д-т | К-т | Примечание |
| 76 | 51 | На сумму оплаты согласно выставленного счета-  фактуры. |
| 26 | 76 | На затраты предприятия отнесена сумма согласно  установленного лимита, за вычетом НДС. |
| 18.03 | 76 | Принят к зачету НДС от суммы, отнесенной на затраты. |
| 70 | 76 | Удержана из зарплаты сотрудника стоимость  телефонных разговоров сверх установленного лимита. |

Решим задачу автоматизации процесса разноски сумм по сотрудникам предприятия и формирования платежного поручение на оплату услуг согласно выставленного счета.

##### Шаг 1. Анализ задачи

Решение любой задачи стоит начинать с ее детального анализа, формализации постановки, проектирования документооборота, структур данных и программного кода.

Бухгалтерские проводки на платформе Гедымин формируются по документу. Следовательно, в нашем случае речь идет о создании нового документа, назовем его "Услуги мобильной связи". Листинг поступает в наше распоряжение один раз в месяц и содержит информацию по нескольким абонентским номерам. Для его отражения наилучшим образом подойдет сложный документ, шапка которого будет содержать дату, а каждая позиция – информацию по одному абоненту: количество звонков, их общая продолжительность и стоимость.

Заполнение документа будет происходить автоматически с помощью макроса на основании текстового файла листинга. Сам файл мы будем также сохранять в базе на тот случай, если в будущем у сотрудника предприятия или контролирующих органов возникнут вопросы по начислениям.

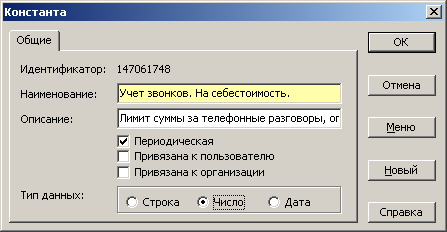
Бухгалтерские проводки будут формироваться для каждой позиции документа на основании типовой хозяйственной операции "Услуги мобильной связи" и определенных для нее типовых проводок.

Для хранения предельной суммы затрат, относимой на себестоимость и действующей ставки НДС в процентах введем две периодические константы.

Мы будем использовать префикс MCC (Mobile Communication Cost) для всех метаданных и макросов, созданных в рамках нашей задачи.

##### Шаг 2. Создание констант

С константами мы подробно ознакомились в разделе "Константы" выше. Откроем Исследователь системы, раздел Сервис и выберем в нем команду Константы. В списке, в левой части окна создадим новый объект и заполним его поля, как показано на Рис. 133 ниже47.

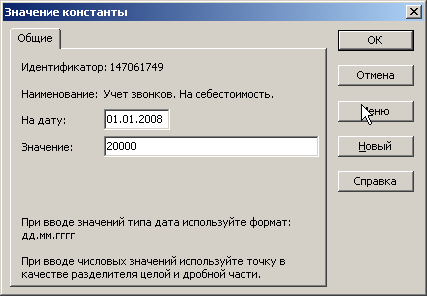


*Рис. 133. Добавление константы*

Обратите внимание, что мы установили флаг **Периодическая** [константа]. Таким образом, если в будущем предприятие изменит сумму лимита или вообще откажется от оплаты телефонных разговоров своих сотрудников, мы всегда сможем корректно выполнить пересчет на любую дату в прошлом.

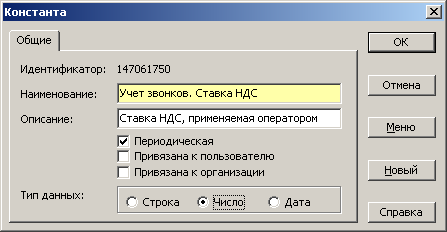
Создав константы, добавим ее значение:

47 Поле Описание, которое не видно на рисунке целиком, содержит строку "Лимит суммы за телефонные разговоры, оплачиваемой предприятием".



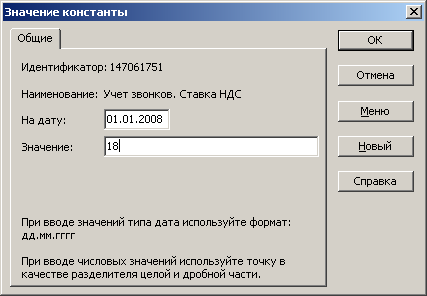
*Рис. 134. Ввод значения константы*

Аналогичным образом создадим константу для хранения ставки НДС по входящим счетам от оператора мобильной связи, см. Рис. 135 ниже.



*Рис. 135. Добавление константы "Ставка НДС"*

И добавим ее значение:



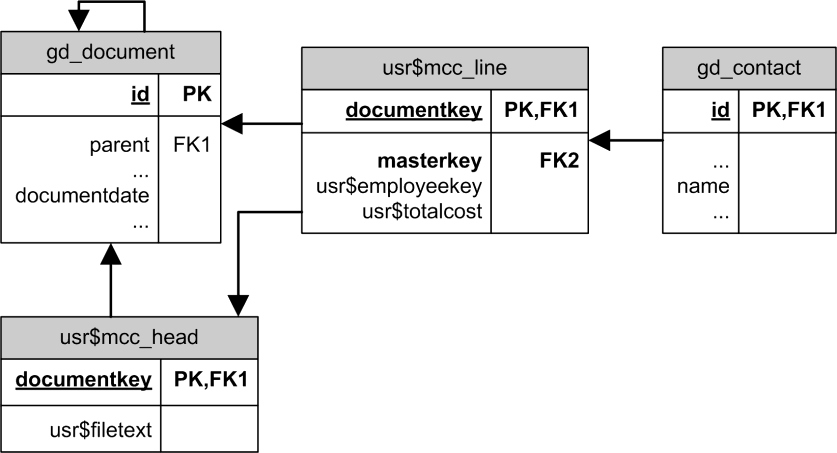
*Рис. 136. Добавление значения константы "Ставка НДС"*

##### Шаг 3. Проектирование структур данных

Данные шапки документа хранятся в двух таблицах: общей таблице gd\_document, и таблице, специфической для документов данного типа, связанных между собой один-к-одному через поля id и documentkey. Атрибут дата документа изначально присутствует в таблице gd\_document (поле documentdate). Так что, нам остается добавить единственный атрибут в присоедененную таблицу – BLOB поле usr$filetext для хранения листинга.

Аналогично шапке, данные о позиции документа хранятся в объединении таблиц gd\_document и таблицы с позициями. В последнюю мы добавим два поля:

usr$employeekey Ссылка на сотрудника предприятия. usr$totalcost Общая стоимость звонков.

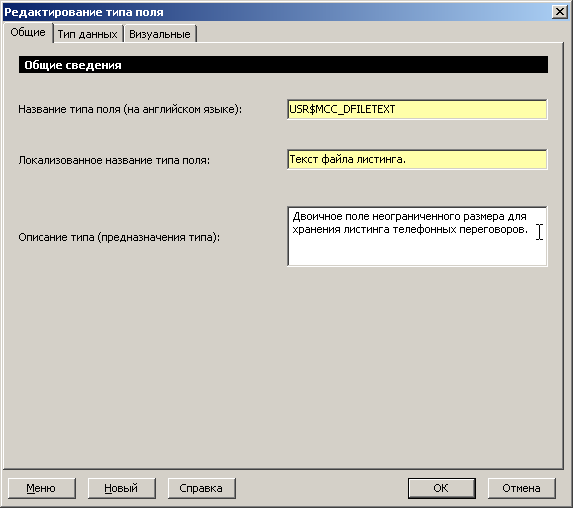


*Рис. 137. Структура БД документа "Услуги мобильной связи"*

На Рис. 137 выше приведена схема связей между таблицей gd\_document и таблицами шапки и позиций нашего документа. Поле usr$employeekey содержит ссылку на запись из справочника gd\_contact, который так же показан на данной схеме. Атрибуты, не представляющие для нас интереса в рамках рассматриваемой задачи, опущены.

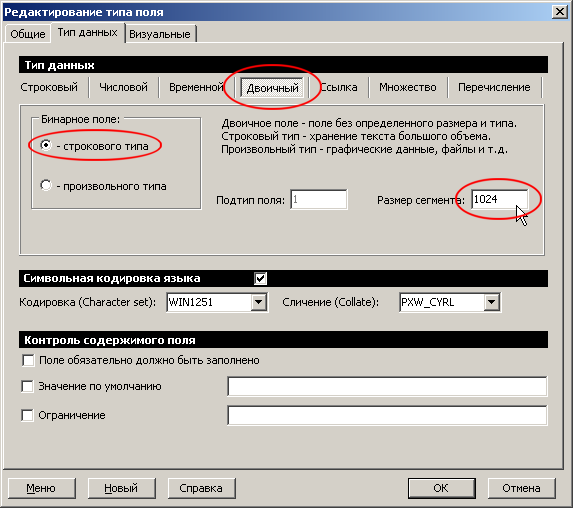
##### Шаг 4. Создание доменов (типов данных)

Создадим домены для трех полей: usr$filetext, usr$employeekey и usr$totalcost. Для этого зайдем в раздел Сервис-Атрибуты Исследователя системы и выберем команду Домены. В появившемся окне выберем команду Создать объект. Первым, создадим домен для поля usr$filetext.



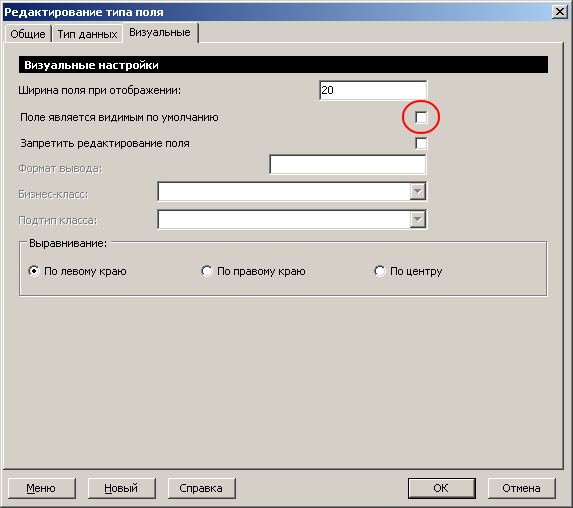
*Рис. 138. Создание домена USR$DFILETEXT*

Введем наименование домена и локализованное название, как показано на Рис. 138. Затем перейдем на вкладку Тип данных. Укажем тип Двоичный, Строковый подтип, размер сегмента 1024 (см. Рис. 139).



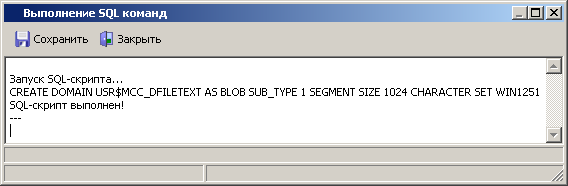
*Рис. 139. Создание домена USR$DFILETEXT. Вкладка тип данных*

Последним шагом, на вкладке Визуальные снимем флаг Поле является видимым по умолчанию (см. Рис. 140).



*Рис. 140. Создание домена USR$DFILETEXT. Вкладка визуальные*

После нажатия на кнопку Ок на экране появится окно с выполненным SQL скриптов, как показано на Рис. 141.

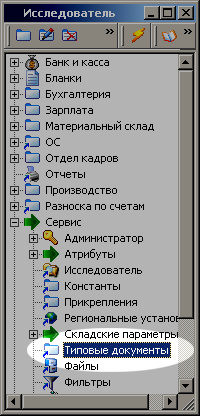


*Рис. 141. SQL скрипт создания домена*

Закроем его и аналогичным образом создадим еще два домена.

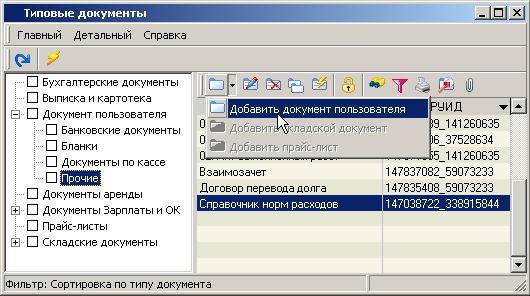
##### Шаг 5. Создание типового документа

Новые документы создаются в окне Типовые документы, которое вызвается командой Сервис-Типовые документы Исследователя системы, как показано на Рис. 142.



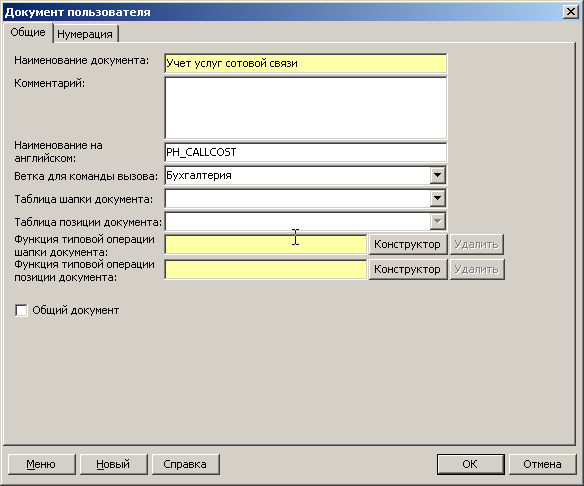
*Рис. 142. Исследователь системы. Команда Типовые документы*

Окно со списком типовых документов (см. Рис. 143) разделено на две части. Слева располагается древовидный список групп, справа – содержимое выбранной группы. Две группы, Складские документы и Прайс-листы, могут содержать только документы соответствующих классов. Все остальные группы предназначены для типовых документов класса Документ пользователя. Их количество и конфигурация задается произвольным образом.



*Рис. 143. Окно Типовые документы*

Создадим наш документ в подгруппе **Прочее** группы **Документ пользователя**. Для этого установим курсор в дереве на соответствующую позицию и в правой части окна, на панели инструментов над списком выберем команду Добавить документ пользователя.

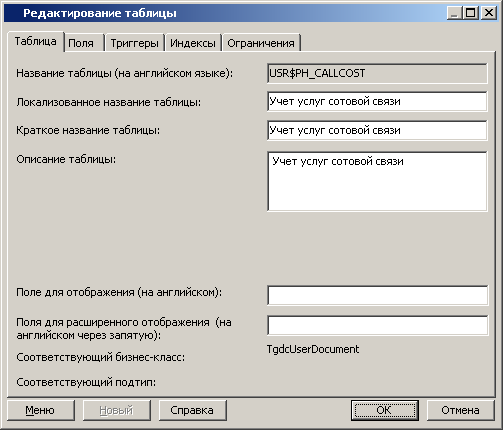


*Рис. 144. Создание типового документа*

Заполним поля диалогового окна следующим образом:

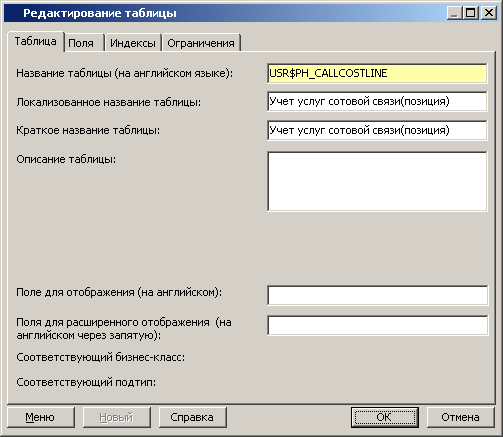
|  |  |
| --- | --- |
| **Наименование поля** | **Значение** |
| Наименование документа | Учет услуг сотовой связи |
| Наименование на  английском | PH\_CALLCOST |
| Ветка для команды вызова | Бухгалтерия |

Установим курсор в поле Таблица шапки документа и нажмем клавишу F2 для создания новой таблицы.



*Рис. 145. Создание таблицы шапки документа*

Как видно на Рис. 145 выше, основные поля диалогового окна уже заполнены автоматически, на основании данных типового документа. Мы не будем добавлять никаких полей, так как необходимый нам атрибут – дата – присутствует в любом документе изначально. Нажмем кнопку Ок после чего будет выполнен и отображен на экране SQL скрипт создания новой таблицы.



*Рис. 146. Создание таблицы позиции документа*

##### Шаг 6. Настройка типовых проводок Шаг 7. Макрос импорта данных из файла

option explicit

sub Macros147061871\_1023724933(OwnerForm, D)

Dim Arr

Set Arr = CreateObject("Scripting.Dictionary")

Dim FSO, F

Set FSO = CreateObject("Scripting.FileSystemObject") Set F = FSO.OpenTextFile("C:\Documents and

Settings\Andrei\Рабочий стол\Kurs\data.txt") While Not F.AtEndOfStream

Dim S, A

S = F.ReadLine

A = Split(S, ",")

If UBound(A) = 4 Then

If Not Arr.Exists(A(0)) Then Arr.Add A(0), 0

End If

Arr.Item(A(0)) = Arr.Item(A(0)) + A(4)

End If

WEnd

Dim Creator, q, K

Set Creator = New TCreator

Set q = Creator.GetObject(nil, "TIBSQL", "") q.Transaction = gdcBaseManager.ReadTransaction q.SQL.Text = \_

"SELECT id FROM gd\_contact WHERE phone = :P"

For Each K In Arr.Keys q.Close

q.ParamByName("P").AsString = K q.ExecQuery

If q.EOF Then

MsgBox "Номер " & K & " не найден"

Exit Sub End If

Arr.Key(K) = q.Fields(0).AsInteger Next

Dim DocHead, DocPos, DS

Set DocHead = Creator.GetObject(nil, "TgdcUserDocument", "")

DocHead.SubType = "147061830\_1023724933"

Set DocPos = Creator.GetObject(nil, "TgdcUserDocumentLine", "")

DocPos.SubType = "147061830\_1023724933"

DocPos.SubSet = "ByParent"

Set DS = Creator.GetObject(nil, "TDataSource", "") DS.DataSet = DocHead

DocPos.MasterSource = DS DocPos.MasterField = "documentkey" DocPos.DetailField = "parent"

DocHead.Open DocHead.Transaction.StartTransaction

DocHead.Insert DocHead.FieldByName("documentdate").AsDateTime = D DocHead.Post

For Each K In Arr.Keys DocPos.Insert

DocPos.FieldByName("USR$EMPLOYEEKEY").AsInteger = K DocPos.FieldByName("USR$TOTALCOST").AsCurrency =

Arr.Item(K)

If Arr.Item(K) > 20000 Then DocPos.FieldbyName("USR$ZATRATY").AsCurrency = 20000 DocPos.FieldByName("USR$IZZARPLATY").AsCurrency =

Arr.Item(K) - 20000

Else

DocPos.FieldbyName("USR$ZATRATY").AsCurrency = Arr.Item(K)

DocPos.FieldByName("USR$IZZARPLATY").AsCurrency = 0 End If

DocPos.Post Next

DocHead.Transaction.Commit

end sub

# Глава XIV. Структура базы

**данных**

В предыдущих главах мы неоднократно говорили, что в основе платформы Гедымин лежат постреляционные объектно-ориентированные технологии доступа к данным в реляционной базе.

**Типы данных (домены)**

|  |  |  |  |
| --- | --- | --- | --- |
| **Название** | **Т** | **N** | **Описание** |
| dintkey | I |  | Первичный ключ таблицы. VALUE > 0. |
| ddbkey | I |  | Идентификатор базы данных. |
| dforeignkey | I |  | Внешняя ссылка. |
| dmasterkey | I |  | Обязательная ссылка. Как правило из  позиции документа на шапку. |
| dlb | I |  | Левая граница интервала в интервальном  дереве. DEFAULT 1. |
| drb | I |  | Правая граница интервала в интервальном  дереве. DEFAULT 2. |
| dparent | I |  | Ссылка на родительский элемент в дереве. |
| dinteger | I |  | Произвольное целочисленное значение. |
| dinteger\_notnull | I |  |  |
| ddouble | D |  |  |
| dalias | V (8) |  |  |
| diso | V(3) |  |  |
| dtemplatetype | V(3) |  | DEFAULT '' |
| dnullalias | V(16) |  |  |
| dtablename | V(31) |  | (IS NULL) OR (> '') |
| dfieldname | V(31) |  | (IS NULL) OR (> '') |
| dname | V(60) |  |  |
| dlongname | V(80) |  |  |
| dbarcode | V(24) |  |  |
|  |  |  |  |

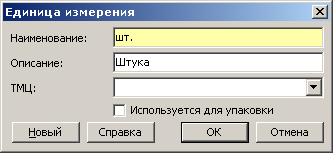
**Системные таблицы**

|  |  |
| --- | --- |
| **Имя таблицы** | **Описание** |
|  |  |
| GD\_COMPANYSTORAGE | Для каждой записи в таблице GD\_OURCOMPANY создается одна запись в GD\_COMPANYSTORAGE, котрая содержит  данные хранилища этой компании. |

|  |  |
| --- | --- |
| GD\_GLOBALSTORAGE | Всегда содержит только одну запись – данные  глобального хранилища. |
| GD\_USERSTORAGE | Хранит данные пользовательского хранилища. Связана 1-1 с учетными записями в таблице  GD\_USER. |
| GD\_DESKTOP |  |
| GD\_DOCUMENT |  |
| GD\_DOCUMENTTYPE |  |

**Справочники**

#### Единицы измерения



*Рис. 147. Диалоговое окно редактирования единицы измерения*

|  |  |
| --- | --- |
| Наименование | Сокращенное наименование единицы измерения.  Например, шт, кг, м и т.п. Поле обязательное для заполнения. |
| Описание | Подробное наименование: штука, килограмм, метр. |
| ТМЦ | Если предприятие ведет учет тары, то в данном поле можно указать связь между единицей измерения и объектом справочника товаров – конкретным видом тары. Более подробно см.  раздел **Учет тары** ниже. |
| Используется для  упаковки |  |

###### Учет тары

Рассмотрим такой пример: предприятие производит и реализует прохладительные газированные напитки в стеклянных бутылках емкостью

0.25 и 0.5 л. Бутылка сама по себе имеет цену и является объектом складского учета. Учет произведенного напитка ведется в литрах. Поступим следующим образом:

1. В справочнике товаров создадим раздел Тара и добавим в него две позиции: "Бутылка 0.25" и "Бутылка 0.5".
2. Аналогично, в справочник единиц измерения добавим следующие записи: "бут. 0.25" и "бут 0.5". Для первой, в поле ТМЦ, укажем товар "Бутылка 0.25", а для второй – "Бутылка 0.5".
3. Для товара "Газ. напиток" укажем основную единицу измерения литры.
4. На вкладке **Ед. изм.** диалогового окна редактирования товара "Газ. напиток" добавим две единицы измерения: "бут. 0.25" и "бут 0.5", указав коэффициенты перевода 0.25 и 0.5 соответственно.

Теперь, если при выписке накладной на отпуск продукции мы укажем 10 единиц в бутылках по 0.25 и 20 единиц по 0.5, то со склада спишутся следующие позиции:

|  |  |
| --- | --- |
| **Товар** | **Количество** |
| Газ. напиток, л | 12.5 |
| Бутылка 0.25 | 10 |
| Бутылка 0.5 | 20 |

### GD\_VALUE

Справочнике единиц измерения.

|  |  |
| --- | --- |
| ID, DINTKEY | Уникальный идентификатор. |
| NAME, DNAME | Сокращенное наименование. |
| DESCRIPTION, DTEXT80 | Полное наименование. |
| GOODKEY, DFOREIGNKEY | Для тары и упаковки -- ссылка на объекта  справочника товаров. |
| ISPACK, DBOOLEAN |  |

### GD\_GOODGROUP

Справочник групп ТМЦ.

|  |  |
| --- | --- |
| ID, DINTKEY | Уникальный идентификатор. |
| PARENT, DPARENT |  |
| LB, DLB |  |
| RB, DRB |  |
| NAME, DNAME |  |
| ALIAS, DALIAS |  |
| DESCRIPTION,  DBLOBTEXT80\_1251 |  |

|  |  |
| --- | --- |
| DISABLED, DDISABLED |  |
| RESERVED, DRESERVED |  |
| AFULL, DSECURITY |  |
| ACHAG, DSECURITY |  |
| AVIEW, DSECURITY |  |

### GD\_GOOD

Справочник ТМЦ.

|  |  |
| --- | --- |
| ID, DINTKEY | Уникальный идентификатор. |
| GROUPKEY, DMASTERKEY |  |
| NAME, DNAME |  |
| ALIAS, DNULLALIAS |  |
| SHORTNAME, DTEXT40 |  |
| DESCRIPTION, DTEXT180 |  |
| BARCODE, DBARCODE |  |
| VALUEKEY, DINTKEY |  |
| TNVDKEY, DFOREIGNKEY |  |
| DISCIPLINE,  DACCOUNTDISCIPLINE |  |
| ISASSEMBLY, DBOOLEAN |  |
| EDITORKEY, DFOREIGNKEY |  |
| EDITIONDATE |  |
| RESERVED, DRESERVED |  |
| DISABLED, DDISABLED |  |
| AFULL, DSECURITY |  |
| ACHAG, DSECURITY |  |
| AVIEW, DSECURITY |  |

### GD\_GOODBARCODE

Дополнительные штрих-коды по товару.

|  |  |
| --- | --- |
| ID, DINTKEY | Уникальный идентификатор. |
| GOODKEY, DINTKEY |  |
| BARCODE, DBARCODE |  |
| DESCRIPTION, DTEXT180 |  |

### GD\_CONST

Список констант.

|  |  |
| --- | --- |
| ID, DINTKEY | Уникальный идентификатор. |
| NAME, DNAME | Наименование константы. |
| COMMENT, DTEXT120 | Произвольный комментарий. |
| CONSTTYPE, DCONSTTYPE | Тип константы задается битовой маской. Первый бит – признак периодичной константы, второй – константы, привязанной к пользователю и третий – константы, зависящей от рабочей организации. Хранится в базе как малое целое с ограниченим по диапазону от 0  до 7 включительно. |
| DATATYPE, CHAR(1) | Тип данных: S или NULL – строка, N –  число, D – дата. |
| EDITORKEY, DFOREIGNKEY | Ссылка на последнего редактора записи. |
| EDITIONDATE,  DEDITIONDATE | Дата и время последнего изменения  записи. |
| AFULL, DSECURITY | Дескриптор полного доступа. |
| ACHAG, DSECURITY | Дескриптор доступа на изменение. |
| AVIEW, DSECURITY | Дескриптор доступа на просмотр. |
| RESERVED, DRESERVED | Поле зарезервировано. |

### GD\_CONSTVALUE

Список констант.

|  |  |
| --- | --- |
| ID, DINTKEY | Уникальный идентификатор. |
| USERKEY, DFOREIGNKEY | Ссылка на контакт для значения константы, привязанного к пользователю  системы. |
| COMPANYKEY, DFOREIGNKEY | Ссылка на контакт для значения константы, привязанного к рабочей  организации. |
| CONSTKEY, DINTKEY | Ссылка на запись в таблице GD\_CONST. |
| CONSTDATE, DDATE | Для периодичной константы, дата  начала действия значения. |
| CONSTVALUE, DTEXT120 | Значение константы в строковом  представлении. |
| EDITORKEY, DFOREINKEY | Ссылка на контакт, который последним  редактировал данную запись. |
| EDITIONDATE, | Дата и время последнего исправления. |

|  |  |
| --- | --- |
| DEDTIONDATE |  |

### AC\_ACCOUNT

План счетов бухгалтерского учета.

|  |  |
| --- | --- |
| Поле, Тип данных | Описание |
| ID, DINTKEY | Уникальный идентификатор записи. |
| PARENT, DPARENT | Ссылка на родительскую запись. |
| LB, DLB | Левая граница интервала. |
| RB, DRB | Правая граница интервала. |
| NAME, DTEXT180 | Наименование счета. |
| ALIAS, DACCOUNTALIAS | Номер счета. |
| ANALYTICALFIELD,  DFOREIGNKEY | Главная аналитика по счету. |
| ACTIVITY, DACCOUNTACTIVITY | Тип счета. Возможные значения: A – Активный, P – пассивный, B – Активно- пассивный. Для записей плана счетов или разделов плана счетов поле содержит  NULL. |
| ACCOUNTTYPE, DCHARTOFACCOUNTPART | Тип записи справочника плана счетов. Возможные значения: C – План счетов, F –  раздел плана счетов, A – счет, S – субсчет. |
| MULTICURR, DBOOLEAN | Признак валютного счета. |
| OFFBALANCE, DBOOLEAN | Признак забалансового счета. |
| FULLNAME | Вычисляемое поле. Полное  наименование счета составляется из номера счета и его наименования. |
| DESCRIPTION, | Произвольный комментарий. |
| DISABLED | Признак отключенного счета. |
| AVIEW |  |
| ACHAG |  |
| AFULL |  |

### AC\_ACCVALUE

Настройка количественного учета для бухгалтерских счетов.

|  |  |
| --- | --- |
| Поле, Тип данных | Описание |
| ID, DINTKEY | Идентификатор записи. |
| ACCOUNTKEY,  DMASTERKEY | Ссылка на запись справочника счетов. |
| VALUEKEY, DINTKEY | Ссылка на запись справочника единиц  измерения. |

### AC\_COMPANYACCOUNT

Для каждой рабочей организации может быть указан свой план счетов или свой набор планов счетов. Таблица AC\_COMPANYACCOUNT содержит связку между рабочей организацией и используемыми для нее планами счетов.

|  |  |
| --- | --- |
| Поле, Тип данных | Описание |
| COMPANYKEY, DINTKEY | Ссылка на таблицу GD\_OURCOMPANY. |
| ACCOUNTKEY, DINTKEY | Ссылка на таблицу AC\_ACCOUNT. |
| ISACTIVE, DBOOLEAN | Признак того, что план счетов является  активным для данной организации. |
| RESERVED, DINTEGER | Зарезервировано для будущего  использования. |

### AC\_TRANSACTION

Древовидный справочник типовых операций.

|  |  |
| --- | --- |
| Поле, Тип данных | Описание |
| ID, DINTKEY | Иденификатор записи. |
| PARENT, DPARENT | Ссылка на родительскую запись. |
| LB, DLB | Левая граница интервала. |
| RB, DRB | Правая граница интервала. |
| NAME, DNAME | Наименование типовой операции. |
| DESCRIPTION, DTEXT180 | Описание операции. |
| COMPANYKEY, DFOREIGNKEY | Ссылка на компанию для которой доступна данная типовая операция. Если не заполнена, операция доступна для  всех рабочих организаций. |
| AFULL, DSECURITY |  |

|  |  |
| --- | --- |
| ACHAG, DSECURITY |  |
| AVIEW, DSECURITY |  |
| DISABLED, DBOOLEAN | Признак отключенной операции. |
| RESERVED, DINTEGER | Зарезервировано для будущего  использования. |
| AUTOTRANSACTION,  DBOOLEAN | Признак автоматической операции. |

### AC\_TRRECORD

Справочник типовых проводок.

|  |  |
| --- | --- |
| Поле, Тип данных | Описание |
| ID, DINTKEY | Идентификатор записи. |
| TRANSACTIONKEY,  DMASTERKEY | Ссылка на типовую операцию  (AC\_TRANSACTION). |
| DESCRIPTION, DTEXT180 | Произвольный комментарий к типовой  проводке. |
| ISSAVENULL, DBOOLEAN | Признак сохранения нулевой проводки.  Если 1 – сохранять, 0 – не сохранять. |
| ACCOUNTKEY, DFOREIGNKEY | Ссылка на план счетов. Если указан, то в алгоритме формирования проводок можно использовать только счета из  выбранного плана. |
| AFULL, DSECURITY |  |
| ACHAG, DESCURITY |  |
| AVIEW, DSECURITY |  |
| DISABLED, DBOOLEAN |  |
| RESERVED, DINTEGER |  |
| FUNCTIONKEY,  DFOREIGNKEY | Ссылка на скрипт-функцию  формирования проводок. |
| DOCUMENTTYPEKEY, DFOREIGNKEY | Ссылка на типовой документ, для которого формируются проводки. Не указывается для автоматической типовой  проводки. |
| DOCUMENTPART, DTEXT10 | Часть документа, для которой формируются проводки. Может содержать значения 'шапка' или  'позиция'. |
| FUNCTIONTEMPLATE, | Шаблон функции. Двоичный код в |

|  |  |
| --- | --- |
| DBLOB80 | формате конструктора функций. |

### AC\_RECORD

Заголовок бухгалтерской проводки.

|  |  |
| --- | --- |
| Поле, Тип данных | Описание |
| ID, DINTKEY | Идентификатор записи. |
| TRRECORDKEY, DINTKEY | Типовая проводка. |
| TRANSACTIONKEY,  DINTKEY |  |
| RECORDDATE, DDATE |  |
| DESCRIPTION, DTEXT180 |  |
| DOCUMENTKEY,  DMASTERKEY |  |
| MASTERDOCKEY, DINTKEY |  |
| COMPANYKEY, DINTKEY |  |
| DEBITNCU, DCURRENCY |  |
| DEBITCURR, DCURRENCY |  |
| CREDITNCU, DCURRENCY |  |
| CREDITCURR, DCURRENCY |  |
| DELAYED, DBOOLEAN |  |
| INCORRECT, DBOOLEAN |  |
| AFULL, ACHAG, AVIEW |  |
| DISABLED, DDISABLED |  |
| RESERVED, DINTEGER |  |

### AC\_ENTRY

Позиция бухгалтерской проводки.

|  |  |
| --- | --- |
| Поле, Тип данных | Описание |
| ID, DINTKEY | Идентификатор записи. |
| RECORDKEY, DMASTERKEY | Ссылка на заголовок проводки. |
| ENTRYDATE, DDATE | Дата проводки. |
| TRANSACTIONKEY,  DINTKEY |  |
| DOCUMENTKEY,  DMASTERKEY |  |
| MASTERDOCKEY, DINTKEY |  |
| COMPANYKEY, DINTKEY |  |

|  |  |
| --- | --- |
| ACCOUNTKEY, DINTKEY |  |
| ACCOUNTPART,  DACCOUNTPART |  |
| DEBITNCU, DCURRENCY |  |
| DEBITCURR, DCURRENCY |  |
| DEBITEQ, DCURRENCY |  |
| CREDITNCU, DCURRENCY |  |
| CREDITCURR, DCURRENCY |  |
| CREDITEQ, DCURRENCY |  |
| CURRKEY, DINTKEY |  |
| DISABLED, DDISABLED |  |
| RESERVED, DINTEGER |  |
| ISSIMPLE,  DBOOLEAN\_NOTNULL |  |
| … |  |

### AC\_QUANTITY

Количественные параметры бухгалтерской проводки.

|  |  |
| --- | --- |
| Поле, Тип данных | Описание |
| ID, DINTKEY | Идентификатор записи. |
| ENTRYKEY, DMASTERKEY | Ссылка на проводку. |
| VALUEKEY, DINTKEY | Ссылка на единицу измерения  (количественный показатель). |
| QUANTITY, DCURRENCY | Количество. |

### AC\_AUTOENTRY

Проводки, сформированные автоматически.

|  |  |
| --- | --- |
| Поле, Тип данных | Описание |
| ID, DINTKEY | Идентификатор записи. |
| ENTRYKEY, DINTKEY | Ссылка на позицию проводки (запись в  таблице AC\_ENTRY). |
| TRRECORDKEY, DINTKEY | Ссылка на типовую проводку. |
| BEGINDATE,  DDATE\_NOTNULL | Начало периода, за который  сформирована автоматическая проводка. |
| ENDDATE,  DDATE\_NOTNULL | Окончание периода, за который  сформирована автоматическая проводка. |
| CREDITACCOUNT, DINTKEY | Ссылка на кредитовый счет. |

|  |  |
| --- | --- |
| DEBITACCOUNT, DINTKEY | Ссылка на дебитовый счет. |

# Глава XVI. Хранилище

В любой программе существуют десятки опций, определяющих ее функционирование. Наверное, вы уже знакомы с реестром Windows — специальной базой данных, где в иерархическом виде содержатся параметры как самой операционной системы, так и программ сторонних разработчиков. Хранилище48 — это аналог реестра Windows в платформе Гедымин. Чем же не угодил разработчикам стандартный механизм? Попробуем ответить на этот вопрос. Реестр Windows размещается на локальном жестком диске компьютера49. Если мы будем использовать его для записи визульных предпочтений пользователя, то при смене компьютера последнему придется настраивать интерфейс заново. Кроме визуальных настроек, хранилище содержит экранные формы, которые являются частью прикладного решения. Помести их в реестр и сравнительно простой процесс апгрейда приложения превратится в сущий ад. Не говоря уже о том, что размещение данных за пределами базы усложнит процедуру архивного копирования.

##### Элементы хранилища

Хранилище содержит разделы (еще именуемые папками) и параметры. Разделы хранилища образуют древовидную структуру. Путем к разделу или его полным именем называется строка, состоящая из последовательного перечисления имен родительских разделов, разделенных символом наклонной черты, а так же имени самого раздела. Пример полного имени:

\Options\Inventory\Equivalent. Одиночный символ наклонной черты ссылается на корневой раздел. Имя раздела или параметра может состоять из любых символов. Регистр символов значения не имеет. Каждый раздел может содержать произвольное число параметров. Параметр имеет имя и может быть одного из следующих типов:



Целое число; Дробное число; Строка;

Дата и время; Булевский тип;

Двоичные данные (поток).

48 Очевидно, что название хранилище было избрано разработчиками чтобы не путать систему хранения параметров платформы Гедымин с реестром Windows.

49 Даже если опытный системный администратор настроит централизованное хранение реестров пользователей на контроллере домена, все равно это не решит проблемы при переносе файла базы данных из одного офиса в другой.

Не допускается присутствие в рамках одного раздела двух параметров или подразделов с одинаковыми именами.



Подобно тому, как реестр операционной системы Windows имеет несколько предопределенных корневых ветвей (CLASSES\_ROOT, CURRENT\_USER, LOCAL\_MACHINE и т.д.) в платформе Гедымин присутствуют несколько хранилищ, каждое из которых имеет свое, строго определенное предназначение.

## Глобальное хранилище

Глобальное хранилище предназначено для параметров общих для всех пользователей и всех рабочих организаций платформы. В частности, в корневом разделе глобального хранилища находятся:

|  |  |
| --- | --- |
| **Имя раздела** | **Описание** |
| DFM | В данном разделе содержатся:  Стандартные экранные формы платформы, настроенные под учетной записью Administrator (т.е. настроенные для всех пользователей системы).  Экранные формы, созданные для пользовательских подтипов бизнес-объектов.  Каждый класс экранных форм располагается в своем подразделе, который имеет один или несколько параметров (по числу пользовательских подтипов). Если для данного класса не определены пользовательские подтипы, то форма хранится в единственном параметре  c именем **default**. |
| NewForm | Список произвольных пользовательских форм. Каждая форма располагается в своем подразделе, который имеет до пяти параметров, три из которых присутствуют у форм всех типов:  имя класса формы, данные формы, внутренний тип.  Два параметра, присутствуют только у форм бизнес- объекта:  тип бизнес объекта,  подтип. |
| Options | Глобальные параметры платформы. |

|  |  |
| --- | --- |
| RegionalSettings | Региональные установки: формат числа, даты, времени  и т.п. |
| SubTypes | Список определенных пользователем подтипов для  каждого бизнес-класса платформы. |

## Пользовательское хранилище

Пользовательское хранилище содержит параметры, привязанных к конкретному пользователю. На верхнем уровне располагаются разделы с настройками экранных форм. Их легко распознать по имени, состоящем из наименования экранной формы и присоединенного к нему, в скобках, имени класса этой формы. Например:

dlgInvDocument147004729\_45137928(TdlgInvDocument)

Так же на верхнем уровне присутствуют следующие системные разделы:

|  |  |
| --- | --- |
| **Имя раздела** | **Описание** |
| DFM | Экранные формы, настроенные под конкретного пользователя.  Данный раздел имеет подразделы, по одному для каждого класса экранных форм. Каждый подраздел имеет один или несколько параметров, по числу пользовательских подтипов. Если для данного класса не определены пользовательские подтипы, то форма хранится в  единственном параметре с именем **default**. |
| GDC | Параметры бизнес классов. Для каждого бизнес класса  имеется соответствующий подраздел. |
| Options | Параметры системы, привязанные к конкретному  пользователю. |

## Хранилище компании

Параметры платформы, привязанные к рабочей организации. Одним из самых распространенных способов использования данного хранилища, является сохранение в формах просмотра и диалоговых окнах выбранного пользователем рассчетного счета компании.

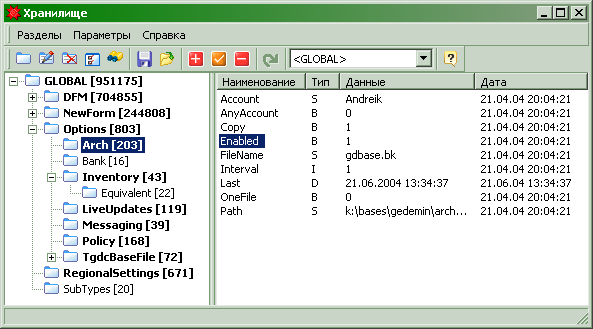
## Хранилище рабочего стола

Хранилище параметров, привязанных к конкретному рабочему столу.

На сегодняшний день практически не используется.

## Окно Хранилища

Откроем **Исследователь системы**, перейдем в раздел **Сервис** и вызовем команду **Хранилище**. На экране откроется окно следующего вида:



*Рис. 148. Окно Хранилища*

Как видно на рисунке, окно содержит меню, панель инструментов, дерево разделов, список параметров и строку состояния. После имени раздела, в квадратных скобках, приводится размер занимаемой им памяти, в байтах. Если раздел не пустой, т.е. имеет хотя бы один параметр или вложенный подраздел, то его имя выводится **жирным** начертанием. Если раздел или параметр входит хотя бы в одну настройку, то его имя выводится синим цветом. При перемещении по дереву в строке состояния отображается полное наименование текущего раздела.

Название корневого раздела соответствует имени хранилища. Для хранилищ пользователя и рабочей компании в скобках указывается имя соответствующего объекта. Например, USER(Administrator) или COMPANY(Тестовая организация). Выбор хранилища осуществляется с помощью выпадающего списка на панели инструментов.

Остановимся подробнее на пункте меню **Разделы**:



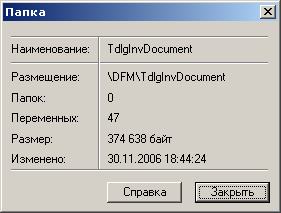
|  |  |  |
| --- | --- | --- |
| **Команда** | **Горячие**  **клавиши** | **Описание** |
| Создать | Ins | Создает новый подраздел в текущем  разделе. |
| Переименовать | F2 | Переименовывает выбранный раздел. |
| Удалить | Ctrl-Del | Удаляет выбранный раздел, включая все  его содержимое. |
| Свойства… |  | Открывает диалоговое окно со  свойствами выбранного раздела (см. Рис. |





|  |  |  |
| --- | --- | --- |
|  |  | 149). |
| Поиск… | F3 | Открывает диалоговое окно поиска. |
| Сохранить в файл… |  | Сохраняет выбранный раздел и все его подразделы в файле. При сохранении можно выбрать формат файла: текстовый, имеет расширение stt, или  двоичный – stb. |
| Загрузить из файла… |  | Добавляет данные из файла в Хранилище. |
| Добавить в настройку |  | Добавляет выбранный раздел в указанную настройку. |
| Обновить | F5 | Перечитывает Хранилище из базы  данных. |

Окно со свойствами раздела (см. Рис. 149) показывает его размер, количество вложенных папок и переменных, дату и время последнего изменения.



*Рис. 149.Свойства раздела Хранилища*

Совет: рекомендуется перед установкой новых настроек создавать архивную копию Глобального Хранилища. Для этого следует открыть Глобальное Хранилище, установить курсор на корневой раздел и вызвать команду Сохранить в файл.

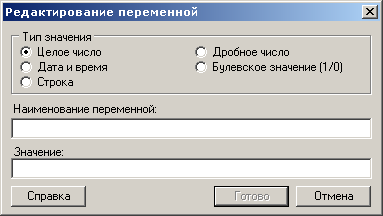
Пункт меню **Параметры** содержит следующие команды:

|  |  |  |
| --- | --- | --- |
| **Команда** | **Горячие**  **клавиши** | **Описание** |
| Создать… |  | Открывает окно создания нового |

|  |  |  |
| --- | --- | --- |
|  |  | параметра (см. Рис. 150). |
| Изменить… |  | Позволяет изменить значение  параметра. |
| Удалить |  | Удаляет выбранный параметр. |
| Переименовать… |  | Позволяет изменить имя выбранного  параметра. |
| Добавить в настройку… |  | Добавляет выбранный параметр в указанную настройку. |

Ниже приведено окно для создания/изменения параметра.

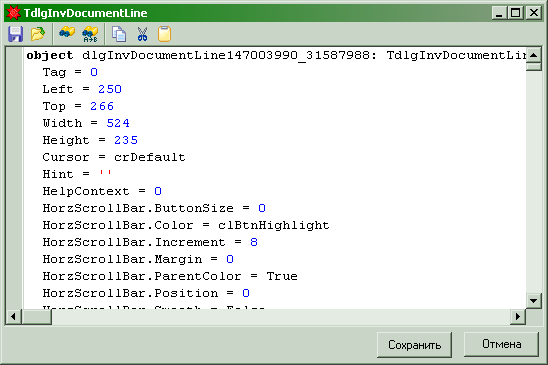




*Рис. 150. Создание нового параметра*

Изменить тип параметра после его создания невозможно. Если был создан параметр с неверным типом, необходимо удалить его и создать новый параметр.

Редактирование параметра типа поток с текстовыми данными осуществляется в отдельном окне. На Рис. 151 показано окно редактирования DFM экранной формы.

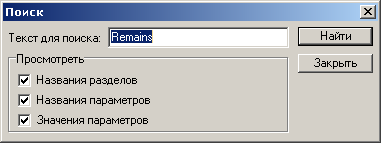


*Рис. 151. Окно редактирования параметра типа поток*

#### Окно поиска

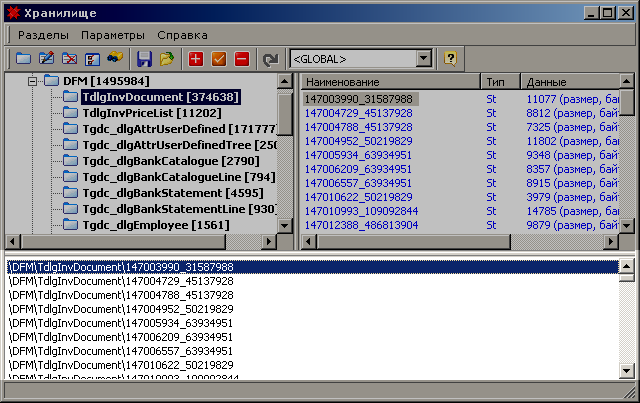
По выбору команды **Поиск…** из меню или по нажатию на клавишу **F3**

открывается диалоговое окно поиска:



*Рис. 152. Поиск в хранилище*

Следует ввести искомую подстроку и указать какие области просматривать. После выполнения поиска, внизу окна хранилища откроется список найденных элементов:



*Рис. 153. Панель с результатами поиска*

Используйте двойной щелчек мыши по найденной записи для установки курсора на соотвествующий раздел в дереве.

## Таблицы хранилища

Рассмотрим, как информация разных хранилищ размещаются в базе данных.

### GD\_COMPANYSTORAGE

Для каждой записи в таблице GD\_OURCOMPANY создается одна запись в таблице GD\_COMPANYSTORAGE, где сохраняются данные хранилища рабочей организации.

|  |  |  |
| --- | --- | --- |
| PK  FK NN | Поле Домен  Тип данных | Описание |
| PK FK  NN | COMPANYKEY DINTKEY  INTEGER | Уникальный идентификатор записи, он же ссылка на таблицу GD\_OURCOMPANY. |
| NN | DATA DBLOB4096  BLOB | Данные пользовательского хранилища. |
| NN | MODIFIED DTIMESTAMP  TIMESTAMP | Дата и время изменения данных. |

### GD\_GLOBALSTORAGE

Данные глобального хранилища находятся в таблице GD\_GLOBALSTORAGE, которая содержит всегда только одну запись.

|  |  |  |
| --- | --- | --- |
| PK  FK NN | Поле Домен  Тип данных | Описание |
| PK NN | ID  DINTKEY INTEGER | Уникальный идентификатор записи. |
| NN | DATA DBLOB4096  BLOB | Данные глобального хранилища. |
| NN | MODIFIED | Дата и время последней записи в таблицу. |

|  |  |  |
| --- | --- | --- |
|  | DTIMESTAMP  TIMESTAMP |  |

### GD\_USERSTORAGE

Для каждой учетной записи пользователя в таблице GD\_USERSTORAGE создается одна запись, которая содержит данные пользовательского хранилища.

|  |  |  |
| --- | --- | --- |
| PK FK  NN | Поле Домен  Тип данных | Описание |
| PK  FK NN | USERKEY  DINTKEY INTEGER | Уникальный идентификатор записи он же ссылка на таблицу GD\_USER. |
| NN | DATA DBLOB4096  BLOB | Данные пользовательского хранилища. |
| NN | MODIFIED DTIMESTAMP  TIMESTAMP | Дата и время изменения данных. |

### GD\_DESKTOP

Данные хранилища рабочего стола находятся в записи с данными о рабочем столе в таблице GD\_DESKTOP.

#### Решение проблемы быстродействия

Структура хранилища легко представляется в рамках реляционной модели. Параметр – это запись, которая ссылается на раздел. Тот, в свою очередь, -- на вышележащий раздел и т.д. Почему же Гедымин помещает данные хранилища целиком в одно БЛОБ поле? Ответ весьма прост. Удобство реляционной модели было пренесено в жертву производительности. Загрузка тысяч параметров, как отдельных записей, в датасет заняла бы десятки секунд при каждом запуске программы. А обращение к параметру с частотой сотни раз в секунду ввело бы в ступор

самую производительную систему50. В процессе работы Гедымина данные хранилища целиком располагаются в оперативной памяти ввиде системы объектов. При сохранении, данные параметров и разделов записываются в поток, который сжимается и помещается в базу. Одновременно этот же поток записывается во временный каталог на локальном жестком диске51. Во время загрузки программы из базы данных считывается время последнего изменения хранилища, если оно совпадает с временем, записанным в файле, то данные загружаются с локального диска. Как правило, хранилище записывается в базу при завершении работы программы, если оно было изменено. Для хранилища обычного пользователя, т.е. не Administrator, никаких проверок при записи не выполняется. Если несколько человек зайдут в программу под одним и тем же логином, то последний вышедший, затрет изменения всех остальных. Вот почему важно, чтобы каждый пользователь системы имел свою учетную запись. Очевидно, что мы не можем поступить аналогичным образом с глобальным хранилищем или учетной записью Administrator52. Вместо этого, при каждом обращении к хранилищу с сервера запрашивается время последней записи в базу данных. Если оно отличается от времени, которое мы получили при считывании, значит кто-то другой уже обновил данные и мы должны перечитать их. Гедымин следит, чтобы между последовательными запросами на синхронизацию прошло не менее 40 миллисекунд. Таким образом, при загрузке программы, когда сразу считываются десятки параметров, сервер не захлебнется от шквала запросов на синхронизацию. Синхронизацией можно управлять с помощью соответствующего параметра при вызове функций OpenFolder и CloseFolder. Если ваш код обращается к параметру глобального хранилища и вас не интересует его сиюминутная актуальность

50 Это не теоретическое предположение, а печальный опыт разработчиков платформы. Где-то, во времена версии 0.4, визуальные настройки хранились в базе в чистом реляционном виде. Когда к серверу подключалось несколько десятков пользователей, только открытие иного диалогового окна могло занять три-четыре минуты.

51 Вы можете проверить временный каталог на своем компьютере. Файлы глобального хранилища имеют расширение GSC, а пользовательского – USC. Имя файла содержит идентификатор базы данных и идентификатор пользователя. Например, g712985883.gsc – глобальное хранилище базы данных с идентификатором 712985883, g712985883\_147059155.usc – хранилище пользователя с идентификатором 147059155 из базы данных 712985883.

52 В силу специфики учетной записи Administrator мы не можем требовать, чтобы в любой момент времени только один пользователь был подключен к программе под этим логином.

(а в большинстве случаев так оно и есть), то всегда отключайте синхронизацию:

' Считываем значение строкового параметра

' из глобального хранилища без синхронизации с БД

' и помещаем его в переменную S Dim F, S

Set F = GlobalStorage.OpenFolder("MyOptions", True, False) S = F.ReadString("MyString", "")

GlobalStorage.CloseFolder F, False

## Работа с хранилищем из макросов

Для обращения к параметрам хранилища существуют три глобальных объекта: **GlobalStorage**, **UserStorage** и **CompanyStorage**. Начнем знакомство с простого примера. Запустим Гедымин и откроем окно **Редактора скрипт- объектов**. Создадим новый макрос с именем **"Test Storage"**, дадим скрипт- функции имя **"TestStorage"**, перейдем на вкладку **Скрипт** и введем следующий текст:

Option Explicit Sub TestStorage

Dim S

S = InputBox("Введите строковое значение:") If S = "" Then

Dim F

Set F = GlobalStorage.OpenFolder("\", False, False) F.DeleteFolder "MyOptions" GlobalStorage.CloseFolder F, True

Else

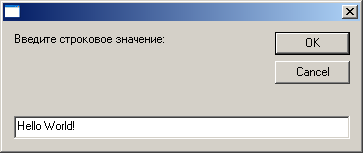
GlobalStorage.WriteString "\MyOptions", "Str", S End If

End Sub

Процедура **TestStorage** запрашивает у пользователя строку и записывает ее в раздел **MyOptions** глобального хранилища под именем **Str**. Если пользователь оставил поле пустым, то вышеуказанный раздел удаляется из хранилища.

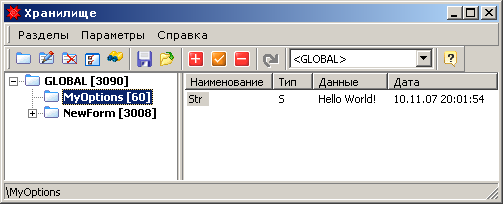
Запустим процедуру и введем строку "Hello World!", как показано на Рис.

154.



*Рис. 154. Ввод строки*

Откроем окно Хранилища, чтобы убедиться, что данные успешно записаны:



*Рис. 155. Строка в глобальном хранилище*

Обратите внимание, как для выполнения удаления мы сначала открыли корневой раздел, удалили из него папку MyOptions и, затем, закрыли его.

#### Объекты хранилища

Глобальные объекты GlobalStorage, UserStorage и CompanyStorage имеют типы TgsGlobalStorage, TgsUserStorage и TgsCompanyStorage соответственно. Диаграмма наследования классов показана ниже.

|  |  |  |
| --- | --- | --- |
| **TgsStorage** |  | **TgsIBStorage** |
|  |

*Рис. 156. Диаграмма наследования классов хранилища*

**TObject**

**TgsCompanyStorage**

**TgsGlobalStorage**

**TgsUserStorage**

Папка хранилища (TgsStorageFolder) и параметр (TgsStorageValue) наследуются от абстрактного базового класса TgsStorageItem (см Рис. 157).

**TgsStorageValue**

**TObject**

**TgsStorageItem**

**TgsStorageFolder**

*Рис. 157. Диаграмма наследования классов элементов хранилища*

На схеме не показаны классы для значений конкретных типов: TgsBooleanValue, TgsCurrencyValue, TgsDateTimeValue, TgsIntegerValue, TgsStreamValue и TgsStringValue. Все они наследуются от TgsStorageItem и за исключением внутренних механизмов они не привносят никакой дополнительной функциональности. По этой же причине не показан и класс TgsRootFolder – наследник TgsStorageFolder, представляющий корневую папку в дереве.

##### TgsStorage

Базовый класс для любого хранилища. Реализует основную его функциональность.

##### property DataString: String

Данные хранилища в строке. Обратите внимание, что данное свойство содержит не текстовое представление хранилища, а именно бинарные данные.

Пример:

Dim S

' Сохраняем хранилище в переменной

S = GlobalStorage.DataString ' Полностью очищаем хранилище GlobalStorage.Clear

' Восстанавливаем хранилище

GlobalStorage.DataString = S

##### property IsModified: Boolean

Показывает изменялись ли данные хранилища с момента его загрузки или последнего сохранения. Только для чтения.

##### property Name: String

Свойство только для чтения. Содержит наименование хранилища. Например, для объекта GlobalStorage свойство Name вернет строку "GLOBAL".

##### function BuildComponentPath(C: TComponent; Context: String): String

C Компонент.

Context Пустая строка или произвольная последовательность символов.

Как правило, используется в событиях OnSaveSettings/OnLoadSettings для создания полного имени раздела, куда записываются настройки компонента. Например, если метка L находится на форме dlgWindow типа TdlgWindow, то вызов функции:

UserStorage.BuildComponentPath(L, "")

Вернет строку "dlgWindow(TdlgWindow)\L(TLabel)". Если параметр Context не пустая строка, то он становится именем концевого раздела.

Пример использования функции:

' Как правило, код размещается в обработчике OnSaveSettings

...

' Сохраним высоту панели между запусками программы

Dim pnlMain, sPath

pnlMain = Sender.GetComponent("pnlMain")

sPath = UserStorage.BuildComponentPath(pnlMain, "") UserStorage.WriteInteger sPath, "Height", pnlMain.Height

...

##### procedure Clear

Очищает хранилище.

##### procedure CloseFolder (gsStorageFolder: TgsStorageFolder; SyncWithDatabase: Boolean)

gsStorageFolder Раздел хранилища. Результат выполнения функции OpenFolder.

SyncWithDatabase Если True, хранилище будет синхронизировано с базой данных после закрытия папки.

Применяется вместе с функцией OpenFolder. Пример использования приведен в описании функции OpenFolder.

##### function FolderExists(APath: String): Boolean

APath Полное имя раздела. Возвращает True, если такой раздел существует. **procedure LoadFromDatabase**

Загружает хранилище из базы данных. В обычных условиях нет

необходимости вызывать данный метод непосредственно.

##### procedure LoadFromFile(AFileName: String; AFileFormat: Variant)

AFileName Имя файла с данными хранилища. AFileFormat Формат данных: 0 – двоичный, 1 – текстовый.

Загружает хранилище из файла.

##### procedure LoadFromStream(S: TStream)

S Двоичный поток с данными. Загружает хранилище из потока.

##### procedure LoadFromStream2(S: TStringStream)

S Строковый поток данных. Загружает хранилище из потока текстового формата.

##### function OpenFolder(APath: String; CanCreate: Boolean; SyncWithDatabase: Boolean): TgsStorageFolder

APath Полное имя раздела.

CanCreate Если заданный раздел не существует и CanCreate = True, то функция создаст и вернет новый раздел.

SyncWithDatabase Если True, то перед открытием раздела хранилище будет синхронизировано с базой данных.

Открывает и возвращает объект раздел хранилища. После окончания использования раздел должен быть закрыт с помощью процедуры CloseFolder. Применяется для записи в раздел нескольких значений.

Пример:

' Такой код мог бы присутствовать в OnSaveSettings Dim F, pnlMain, sPath

pnlMain = Sender.GetComponent("pnlMain")

sPath = UserStorage.BuildComponentPath(pnlMain, "") F = UserStorage.OpenFolder(sPath, True, False) Call F.WriteInteger("Left", pnlMain.Left)

Call F.WriteInteger("Top", pnlMain.Top) Call F.WriteInteger("Width", pnlMain.Width)

Call F.WriteInteger("Height", pnlMain.Height) Call UserStorage.CloseFolder(F, False)

##### function ReadBoolean(APath: String; AValue: String; Default: Boolean): Boolean

APath Имя раздела.

AValue Имя параметра.

ADefault Значение по-умолчанию. Если заданный раздел не найден или параметр с указанным именем не существует, то функция вернет ADefault значение.

Считывает параметр булевского типа из хранилища. Если указанный параметр существует, но имеет не булевский тип, то его значение будет преобразовано к булевской величине.

Пример:

If UserStorage.ReadBoolean("\Options", "Magic", True) Then Application.MessageBox \_

"Магическое перемещение окон включено", "", vbOkOnly End If

##### function ReadCurrency(APath: String; AValue: String; Default: Currency): Currency

APath Имя раздела.

AValue Имя параметра.

ADefault Значение по-умолчанию. Если заданный раздел не найден или параметр с указанным именем не существует, то функция вернет ADefault значение.

Считывает параметр типа Currency из хранилища. Если указанный параметр существует, но имеет не денежный тип, то его значение будет преобразовано к типу Currency.

##### function ReadDateTime(APath: String; AValue: String; Default: Date): Date

APath Имя раздела.

AValue Имя параметра.

ADefault Значение по-умолчанию. Если заданный раздел не найден или параметр с указанным именем не существует, то функция вернет ADefault значение.

Считывает параметр типа дата и время из хранилища. Если параметр существует, но имеет тип, отличный от типа дата, то его значение будет преобразовано.

##### function ReadInteger(APath: String; AValue: String; Default: Integer): Integer

APath Имя раздела.

AValue Имя параметра.

ADefault Значение по-умолчанию. Если заданный раздел не найден или параметр с указанным именем не существует, то функция вернет ADefault значение.

Считывает целочисленный параметр из хранилища. Если параметр существует, но не является целым числом, то его значение будет преобразовано к целому.

##### function ReadStream(APath: String; AValue: String; S: TStream): Boolean

APath Имя раздела.

AValue Имя параметра.

ADefault Значение по-умолчанию. Если заданный раздел не найден или параметр с указанным именем не существует, то функция вернет ADefault значение.

Считывает двоичный поток данных из хранилища. Функция возвращает True, если данные считаны и успешно загружены в поток S и False – в противном случае.

##### function ReadString(APath: String; AValue: String; Default: String): String

APath Имя раздела.

AValue Имя параметра.

ADefault Значение по-умолчанию. Если заданный раздел не найден или параметр с указанным именем не существует, то функция вернет ADefault значение.

Считывает строковый параметр из хранилища. Если параметр существует, но не является строкой, то его значение будет преобразовано в строку в результате выполнения данной функции.

##### procedure SaveToDatabase

Сохраняет хранилище в базе данных. В обычных условиях у разработчика нет необходимости в непосредственном вызове данного метода.

##### procedure SaveToFile(AFileName: String; AFileFormat: Variant)

AFileName Имя файла, куда будет сохранено хранилище. AFileFormat Формат файла. 0 – двоичный, 1 – текстовый.

Сохраняет хранилище в указанном файле. **procedure SaveToStream(S: TStream)** S Поток.

Сохраняет хранилище в двоичном потоке.

##### procedure SaveToStream2(S: TStringStream)

S Текстовый поток.

Сохраняет хранилище в потоке, в текстовом формате.

##### function ValueExists(APath: String; AValue: String): Boolean

APath Имя раздела.

AValue Имя параметра.

Проверяет наличие параметра с именем AValue в папке APath.

Возвращает True в случае успеха и False – в противном случае.

##### procedure WriteBoolean(APath: String; AValueName: String; AValue: Boolean)

APath Полный путь к разделу. AValueName Имя параметра.

AValue Новое значение параметра.

Записывает значение в хранилище по указаному пути. Если параметр с таким именем уже существует, но его тип отличается от типа AValue, то новое значение будет преобразовано к типу параметра и в таком виде помещено в хранилище. Если раздел и/или параметр не существует, то он будет создан.

##### procedure WriteCurrency(APath: String; AValueName: String; AValue: Currency)

APath Полный путь к разделу. AValueName Имя параметра.

AValue Новое значение параметра.

Записывает значение в хранилище по указаному пути. Если параметр с таким именем уже существует, но его тип отличается от типа AValue, то новое значение будет преобразовано к типу параметра и в таком виде помещено в хранилище. Если раздел и/или параметр не существует, то он будет создан.

##### procedure WriteDateTime(APath: String; AValueName: String; AValue: Date)

APath Полный путь к разделу. AValueName Имя параметра.

AValue Новое значение параметра.

Записывает значение в хранилище по указаному пути. Если параметр с таким именем уже существует, но его тип отличается от типа AValue, то новое значение будет преобразовано к типу параметра и в таком виде помещено в хранилище. Если раздел и/или параметр не существует, то он будет создан.

##### procedure WriteInteger(APath: String; AValueName: String; AValue: Integer)

APath Полный путь к разделу. AValueName Имя параметра.

AValue Новое значение параметра.

Записывает значение в хранилище по указаному пути. Если параметр с таким именем уже существует, но его тип отличается от типа AValue, то новое значение будет преобразовано к типу параметра и в таком виде помещено в хранилище. Если раздел и/или параметр не существует, то он будет создан.

##### procedure WriteStream(APath: String; AValueName: String; AValue: TStream)

APath Полный путь к разделу. AValueName Имя параметра.

AValue Новое значение параметра.

Записывает значение в хранилище по указаному пути. Если раздел и/или параметр не существует, то он будет создан.

##### procedure WriteString(APath: String; AValueName: String; AValue: String)

APath Полный путь к разделу. AValueName Имя параметра.

AValue Новое значение параметра.

Записывает значение в хранилище по указаному пути. Если параметр с таким именем уже существует, но его тип отличается от типа AValue, то новое значение будет преобразовано к типу параметра и в таком виде помещено в хранилище. Если раздел и/или параметр не существует, то он будет создан.

##### TgsIBStorage

Интерфейс класса TgsIBStorage не добавляет каких-либо своих методов или свойств относительно TgsStorage. Данный класс реализует хранилище, данные которого записываются в БД.

##### TgsGlobalStorage

Глобальное хранилище. Данные записываются в таблицу GD\_GLOBALSTORAGE.

##### TgsUserStorage

Пользовательское хранилище. Данные находятся в таблице GD\_USERSTORAGE.

##### property UserKey: Integer

Идентификатор пользователя, которому принадлежит хранилище. Если программно изменить значение данного свойства, то загрузится хранилище указанного пользователя. Не забывайте после окончания работы, вернуть свойству исходное значение. Ниже приведен пример копирования опции "Показывать нули в таблице" от текущего пользователя всем пользователям системы.

Пример:

Option Explicit Sub DistrSetting

' В конце выполнения функции мы должны вернуть

' пользовательское хранилище в исходное состояние.

' Чтобы гарантированно дойти до конца функции,

' отключим обработку ошибок. On Error Resume Next

Dim I, R, B

gdcBaseManager.ExecSingleQueryResult \_ "SELECT id FROM gd\_user WHERE id <> :ID", \_ IBLogin.UserKey, R, Nil

If IsArray(R) Then

B = UserStorage.ReadBoolean("\Options", "ShowZero",\_ True)

For I = LBound(R, 2) To UBound(R, 2) UserStorage.UserKey = R(0, I) UserStorage.WriteBoolean "\Options", "ShowZero", B

Next

UserStorage.UserKey = IBLogin.UserKey End If

' Если что-то прошло не так, сообщим пользователю

' об ошибке.

If Err.Number <> 0 Then Application.MessageBox \_

"В процессе выполнения произошла ошибка: " &\_ vbNewLine & Err.Description, "Ошибка!", \_ vbCritical Or vbOkOnly

Else

Application.MessageBox "Значение скопировано успешно",\_

"", vbOkOnly End If

End Sub

##### TgsCompanyStorage

Хранилище компании. Данные записываются в таблицу GD\_COMPANYSTORAGE.

##### property CompanyKey: Integer

Идентификатор рабочей компании, которой принадлежит хранилище. Если программно изменить значение данного свойства, то загрузится хранилище указанной компании.

##### TgsStorageItem

**property DataSize: Integer**

Размер памяти, занимаемой элементом хранилища. Для папки учитываются все значения и все вложенные папки. Только для чтения.

##### property Name: String

Наименование элемента. Допускается изменение имени при условии, что оно будет уникальным в пределах своего раздела.

**property Parent: TgsStorageItem** Родительский элемент. Только для чтения. **property Path: String**

Полный путь к данному элементу. Только для чтения.

##### Storage: TgsStorage

Хранилище, которому принадлежит данный элемент. Только для чтения.

##### procedure Assign(Source: TgsStorageItem)

Копирует значение Source в текущий элемент.

##### function Find(AList: TStringList; ASearchString: String; ASearchOptions: String): Boolean

AList Список строк для размещения результата поиска. ASearchString Искомая строка. Поиск осуществляется по вхождению,

без учета регистра символов.

ASearchOptions Набор опций поиска. Строка может содержать множество из следующих значений: "GSTSOVALUE" – поиск среди наименований параметров, "GSTSOFOLDER" – поиск среди имен разделов, "GSTSODATA" – поиск среди значений параметров. В

качестве разделителя может использоваться любой символ, например, пробел.

Функция сканирует содержимое папки и добавляет к переданному списку полные имена (пути) элементов, удовлетворяющих критерию поиска. Возвращает True, если найден хотя бы один элемент. Так как переданный список не очищается, функция может быть вызвана многократно с объединением результатов поиска.

Пример:

' Найдем все элементы, содержащие в наименовании слово Check Dim F, S

Set S = Designer.CreateObject(Nil, "TStringList", "") Set F = GlobalStorage.OpenFolder("\", False, False)

If F.Find(S, "Check", "GSTSOVALUE GSTSOFOLDER") Then \_

Application.MessageBox S.Text, "Найдено:", vbOkOnly GlobalStorage.CloseFolder F, False Designer.DestroyObject S

##### function GetStorageName: String

Возвращает наименование хранилища, которому принадлежит данный элемент.

##### procedure LoadFromFile(FileName: String; FileFormat: Integer)

FileName Имя файла.

FileFormat Формат файла. 0 – двоичный, 1 – текстовый. Загружает элемент из файла.

##### procedure SaveToFile(FileName: String; FileFormat: Integer)

FileName Имя файла.

FileFormat Формат файла. 0 – двоичный, 1 – текстовый. Сохраняет элемент в файле.

##### TgsStorageValue

**property AsBoolean: Boolean**

Позволяет получить доступ к значению, как к булевскому типу. Если параметр содержит данные иного типа, они будут преобразованы. Если преобразование невозможно, будет сгенерирована ошибка.

##### property AsCurrency: Currency

Позволяет получить доступ к значению, как к денежному типу. Если параметр содержит данные иного типа, они будут преобразованы. Если преобразование невозможно, будет сгенерирована ошибка.

##### property AsDateTime: Date

Позволяет получить доступ к значению, как к типу даты и времени. Если параметр содержит данные иного типа, они будут преобразованы. Если преобразование невозможно, будет сгенерирована ошибка.

##### property AsInteger: Integer

Позволяет получить доступ к значению, как к целочисленному типу. Если параметр содержит данные иного типа, они будут преобразованы. Если преобразование невозможно, будет сгенерирована ошибка.

##### property AsString: String

Позволяет получить доступ к значению, как к строковому типу. Если параметр содержит данные иного типа, они будут преобразованы. Если преобразование невозможно, будет сгенерирована ошибка.

##### function GetStorageValueClass: String

Возвращает имя класса параметра хранилища. Возможные значения: TgsBooleanValue, TgsCurrencyValue, TgsDateTimeValue, TgsIntegerValue, TgsStreamValue, TgsStringValue.

##### function GetTypeID: Integer

Возвращает целочисленный идентификатор типа. Используется при сохранении данных в двоичный поток. Возможные значения:

1. Тип данных не известен;
2. Integer;
3. String;
4. Stream;
5. Boolean;
6. TDateTime;
7. Currency.

##### function GetTypeName: String

Возвращает символьный индентификатор типа. Используется при отображении данных в окне Хранилища и при сохранении в текстовом формате. Возможные значения:

I Integer;

S String;

St Stream;

B Boolean;

D TDateTime;

C Currency.

##### function ShowEditValueDialog: Integer

Открывает на экране диалоговое окно редактирования параметра (см.

Рис. 150). Возвращает mrOk или mrCancel.

##### TgsStorageFolder

**property Folders(Index: Integer): TgsStorageFolder**

Массив подразделов. Индекс начинается с нуля. **property FoldersCount: Integer** Количество подразделов.

##### property Values(Index: Integer): TgsStorageValue

Массив параметров. Индекс начинается с нуля. **property ValuesCount: Integer** Количество параметров.

##### function AddFolder(F: TgsStorageFolder): Integer

**procedure BuildTreeView(N: TTreeNode) procedure Clear**

Удаляет все содержимое папки.

##### function CreateFolder(AName: String): TgsStorageFolder

Создает и возвращает объект вложенной папки с именем AName. Если подраздел с такими именем уже существует, генерирует ошибку.

##### procedure DeleteFolder(AName: String)

Удаляет папку с именем AName.

##### function DeleteValue(AName: String): Boolean

Удаляет параметр с именем AName. Возвращает True в случае успешного удаления и False – если параметр с таким именем не найден.

##### procedure DropFolder

Удаляет объект папки со всем его содержимым. После вызова данного метода обращаться к экземпляру класса больше нельзя.

##### procedure ExtractFolder(F: TgsStorageFolder)

Удаляет папку F из списка вложенных папок текущего объекта. При этом сам объект F продолжает свое существование и может быть добавлен в другой раздел с помощью метода AddFolder.

##### function FindFolder(F: TgsStorageFolder; GoSubFolders: Boolean): Boolean

Возвращает результат True, если объект F является вложенной папкой текущего раздела. Если GoSubFolders = True, поиск распространяется рекурсивно на все подразделы.

##### function FolderByName(AName: String): TgsStorageFolder

Ищет подраздел с именем AName и возвращает его. Если такого имени не существует, функция вернет nil.

##### function FolderExists(AName: String): Boolean

Возвращает True, если раздел с именем AName существует.

##### function GetNextFolderName: String function GetNextValueName: String procedure LoadFromStream(S: TStream)

**function MoveFolder(NewParent: TgsStorageFolder): Boolean**

Перемещает папку в раздел, заданный параметром NewParent. Функция ворзвращает True, если перемещение прошло успешно.

**function OpenFolder(APath: String; CanCreate: Boolean): TgsStorageFolder**

**ReadBoolean(AValueName: String; Default: Boolean): Boolean**

**ReadCurrency(AValueName: String; Default: Currency): Currency**

**ReadDateTime(AValueName: String; Default: Date): Date**

**ReadInteger(AValueName: String; Default: Integer): Integer**

**ReadStream(AValueName: String; S: TStream) ReadString(AValueName: String; Default: String):**

**String**

**SaveToStream(S: TStream) ShowPropDialog**

Выводит на экран диалоговое окно со свойствами папки (см. Рис. 149).

##### ValueByName(AValueName: String): TgsStorageValue

Возвращает параметр с именем AValueName или nil, если заданное имя не найдено.

##### ValueExists(AValueName: String): Boolean

Возвращает True или False в зависимости от того, существует параметр с именем AValueName или нет.

##### WriteBoolean(AValueName: String; Default: Boolean)

Присваивает параметру с именем AValueName значение Default. Если параметр не существует, он будет создан. При необходимости, выполняется преобразование типов.

##### WriteCurrency(AValueName: String; Default: Currency)

Присваивает параметру с именем AValueName значение Default. Если параметр не существует, он будет создан. При необходимости, выполняется преобразование типов.

##### WriteDateTime(AValueName: String; Default: Date)

Присваивает параметру с именем AValueName значение Default. Если параметр не существует, он будет создан. При необходимости, выполняется преобразование типов.

##### WriteInteger(AValueName: String; Default: Integer)

Присваивает параметру с именем AValueName значение Default. Если параметр не существует, он будет создан. При необходимости, выполняется преобразование типов.

##### WriteStream(AValueName: String; S: TStream)

Записывает содержимое потока S в параметр AValueName. Если параметр не существует, он будет создан. Если параметр существует, но не является потоком (TgsStreamValue), то никакх действий не будет предпринято.

##### WriteString(AValueName: String; Default: String)

Присваивает параметру с именем AValueName значение Default. Если параметр не существует, он будет создан. При необходимости, выполняется преобразование типов.

##### WriteValue(AValue: TgsStorageValue)

Создает копию значения AValue. AValue должно принадлежать другому разделу, иначе возникнет конфликт дублирующегося имени.

## Заключение

Хранилище – это своего рода база данных внутри базы данных платформы, специализирующаяся на обработке многочисленных разнородных параметров, организованных в древовидную иерархическую структуру. Для минимизации нагрузки на сервер информация хранилища располагается в оперативной памяти в процессе работы программы и кэшируется на локальном жестком диске при выходе из нее. Механизм

синхронизации позволяет предотвратить потерю изменений при одновременной работе нескольких пользователей.

# Глава XVII. Безопасность

Подсистема безопасности платформы Гедымин предоставляет развитые средства разграничения доступа к данным и функциям системы, а так же позволяет организовать аудит действий пользователя и вести журнал совершенных изменений данных.

Следует заметить, что хотя абсолютно устойчивых к несанкционированному доступу систем не существует, грамотный администратор всегда может создать программно-аппаратный комплекс, затраты на взлом которого будут многократно превышать стоимость данных, полученных в результате такого взлома, т.е. сделают его экономически нецелесообразным.

Для начала, мы рассмотрим средства разграничения доступа, предоставляемые сервером Interbase, поскольку, во-первых, на них частично базируется подсистема безопасности платформы Гедымин, во-вторых, знание механизмов разграничения доступа предоставляемых стандартом SQL может пригодиться разработчику при организации прямого обращения к базе данных из сторонних программ. Если вы уверены в своих знаниях SQL, то можете лишь бегло ознакомиться с содержимым следующих параграфов, либо сразу перейти к главе «Разграничение прав доступа на платформе Гедымин».

## Средства разграничения доступа сервера Interbase

Сервер Interbase поддерживает механизмы определенные стандартом SQL-92 (включая роли) для разграничения прав пользователей для доступа к данным в базе.

Только пользователь, для которого создана учетная запись может осуществить подключение к серверу и базе данных. При этом необходимо указать имя учетной записи и пароль.

Учетные записи могут быть созданы при помощи: утилиты командной строки gsec;



графических утилит для работы с базами данных: IBExpert, QuickDesk, IBConsole и др.;

 вызовов соответствующих функций API Interbase (напрямую или используя промежуточные классы, например, из библиотеки Delphi компонент Interbase Express (сокращенно IBX).

Interbase использует единый список пользователей для всех баз данных, находящихся на одном физическом сервере. Информация об учетных

записях хранится в отдельной базе данных, в файле с именем isc4.gdb53, находящемся в корневом каталоге сервера54.

Хотя имя пользователя и его пароль могут иметь длинну до 32-х символов, значимыми являются только первые 8, т.е. пароли

«VeryLongPassword» и «VeryLong» для сервера идентичны. Для строки пароля различается регистр символов.

##### Учетная запись SYSDBA

Сразу после установки сервера Interbase существует только одна учетная запись с именем SYSDBA и паролем masterkey. Эта учетная запись обладает максимальными правами и позволяет выполнять любые действия как над данными в базе, так и над структурой базы данных. Учитывая

«всемогущество» записи SYSDBA важно изменить ее пароль сразу после установки сервера.

###### Предотвращение несанкционированного доступа

Существует достаточно простой способ получения доступа к информации в базе. Для этого достаточно скопировать файл базы данных на другой компьютер, проинсталлировать там сервер Interbase и подключиться к этой базе используя учетную запись SYSDBA и пароль по умолчанию. Вот почему важно исключить возможность несанкционированного доступа пользователей к файлу базы данных. Если используется операционная система Windows 2000 Server55, то сделать это достаточно просто, установив соответствующие права доступа на директорию, где находится файл базы данных или непосредственно на сам файл56. Однако, не следует забывать, что если будет похищен компьютер, являющийся сервером или винчестер, на котором располагается база данных, то злоумышленник всегда сможет подключить винчестер в другой компьютер, на котором установлена операционная система Windows 2000

53 В зависимости от клона сервера Interbase (Firebird, Yaffil) и его версии имя файла с информацией об учетных записях может различаться. При необходимости, обратитесь к соответствующей документации по работе с сервером базы данных.

54 Для сервера Yaffil, как правило, полный путь к базе данных с учетными записями выглядит следующим образом: c:\Program Files\Yaffil\isc4.gdb.

55 В равной степени сказанное можно отнести и к операционным системам Windows NT 4.0, Windows 2003 Server и Windows XP. Установка прав доступа на файл возможна только в файловой системе NTFS.

56 Достаточно предоставить права на данный файл (директорию) учетной записи SYSTEM и глобальной пользовательской группе Администраторы. Кроме файла базы данных необходимо защитить файл с информацией об учетных записях isc4.gdb.

Server и получить полный доступ к файлу базы данных. Противодействием здесь может быть:

 Размещение сервера в отдельном, хорошо защищенном и охраняемом помещении;

 Хранение файла базы данных на съемном носителе информации, который по окончании рабочего дня изымается и помещается в безопасное место;

 Шифрование файла базы данных средствами операционной системы57. Даже если злоумышленник получит доступ к такому файлу, на другом компьютере он не сможет прочитать его.

Очевидно, что следование одной или нескольким из перечисленных выше рекомендаций способно существенно снизить вероятность взлома базы данных и утечки конфеденциальной информации.

###### Предотвращение несанкционированного архивирования базы данных

Если злоумышленник не имеет прямого доступа к файлу базы данных на сервере, но знает пароль учетной записи SYSDBA или может войти в Гедымин под учетной записью Администратор или под учетной записью, входящей в группы Администраторы или Операторы архива, то он может выполнить архивное копирование базы данных, поместив архивный файл в любой доступный каталог на сервере. В последствии, злоумышленник может получить доступ к данным в базе, скопировав архив и разархивировав его на другом компьютере.

Для предотвращения утечки данных, держите пароль учетной записи SYSDBA (сервера Interbase) и Администратор (системы Гедымин) в тайне, а так же строго контролируйте членство в группах Администраторы и Операторы архива.

## Разграничение прав доступа на платформе Гедымин

Платформа Гедымин использует свою оригинальную систему разграничения прав доступа, которая частично базируется на механизмах сервера Interbase, и позволяет осуществлять:

 Аутентификацию пользователя;

57 Доступно в Windows 2000 Server и более новых версиях этой операционной системы при использовании файловой системы NTFS.

Разграничение прав доступа к функциям системы; Разграничение прав доступа к бизнес классам;



Разграничение прав доступа к отдельным бизнес объектам (записям в данных конкретного бизнес класса);



Аудит действий пользователя; Аудит изменений данных;

Блокировку изменения данных в указанном временном периоде.

Разграничение прав доступа осуществляется на уровне групп пользователей. В свойствах каждого пользователя присутствует поле INGROUP типа INTEGER, которое хранит битовую маску групп, участником которых он является. Все группы пронумерованы. Наличие установленного бита в определенной позиции маски означает, что пользователь является членом группы с номером, соответствующим номеру позиции. Например, значение 45 поля INGROUP, в двоичном исчислении представляется как 101101, что означает, что пользователь входит в группы с номерами: 1, 3, 4, 6. Нумерация битов начинается с единицы и осуществляется справа налево. Поскольку, в настоящее время58 тип данных INTEGER представляет собой 32-х битовое целое число, то количество групп ограничено 32-мя. Первые шесть групп с номерами от 1 до 6 являются системными и присутствуют изначально в эталонной базе данных. Остальные 26 групп могут быть созданы и использованы пользователем (настройщиком) по своему усмотрению.

#### Аутентификация пользователя

Для подключения к системе Гедымин надо иметь учетную запись и знать ее пароль. В отличие от сервера Interbase, все учетные записи Гедымина хранятся в самой базе данных в таблице GD\_USER. Таким образом, существенно облегчается процесс переноса файла базы с одного сервера на другой, что особенно актуально при развертывании распределенной базы данных.

При создании учетной записи в системе Гедымин автоматически создается учетная запись на сервере Interbase. Ее имя и пароль генерируются системой. Этой учетной записи предоставляется право на использование роли ADMINISTRATOR, которая изначально определена в каждой базе данных Гедымина. Роли ADMINISTRATOR предоставлены полные права на доступ ко всем таблицам и процедурам. При подключении

58 В будущем, при переходе на 64-битные вычисления, количество битов в типе INTEGER наверняка возрастет.

к серверу базы данных используется роль ADMINISTRATOR и учетная запись сервера Interbase, сопоставленная учетной записи платформы Гедымин. В процессе работы (за исключением момента аутентификации) Гедымин не использует средства SQL для разграничения доступа пользователей к таблицам и колонкам базы данных. Вместо этого используются внутренние механизмы платформы.

Важным элементом механизма безопасности платформы Гедымин является учетная запись сервера Interbase с именем STARTUSER и паролем startuser. Поскольку учетные записи Interbase не передаются вместе с базой данных, а хранятся в отдельной базе на сервере, то STARTUSER создается инсталлятором в процессе установки серверной части платформы. Если по какой-либо причине используется ручная установка или база данных размещается на компьютере, где уже установлен и сконфигурирован сервер, например, для работы с другими приложениями, то данную учетную запись необходимо создать вручную, либо с помощью утилиты командной строки gsec, либо с помощью любой доступной графической надстройки.

Единственным правом, которым обладает учетная запись STARTUSER является право выполнения процедуры аутентификации GD\_P\_SEC\_LOGINUSER. Таким образом, хотя имя и пароль этой учетной записи известны всем, подключение под ней не представляет никакой опасности59. Само сабой разумеется, что учетная запись STARTUSER не обладает правом использования роли Administrator.

В базе данных присутствует процедура с именем GD\_P\_SEC\_LOGINUSER, которая отвечает за аутентификацию пользователя. На вход этой процедуры передается наименование учетной записи платформы Гедымин и введенный пользователем пароль. Процедура возвращает код завершения (успех, или номер ошибки), а так же имя учетной записи Interbase, ее пароль и дополнительную информацию.

Рассмотрим подробно, как происходит загрузка платформы и ее подключение к серверу базы данных.

1. Запускается файл GEDEMIN.EXE;

59 Теоретически учетная запись STARTUSER может быть использована для подбора пароля методом перебора, но учитывая, что выполнение хранимой процедуры даже на производительном сервере занимает достаточный промежуток времени (порядка миллисекунды), а использование восьмисимвольного пароля, состоящего из цифр и букв английского алфавита дает порядка 368≈1013 вариантов, то перебор всего диапазона займет более 300 лет.

1. Определяется имя сервера и путь к файлу базы данных:
   1. Если среди параметров командной строки присутствует ключ /sn, то его значение используется в качестве имени сервера и пути к файлу базы данных;
   2. Если в командной строке имя сервера и путь к базе данных не заданы, то из системного реестра считывается информация о последнем подключении;
2. С использованием имени сервера и пути к файлу базы данных осуществляется попытка подключиться используя учетную запись STARTUSER и ее пароль;
3. Если возникла ошибка, то проверяется ее код:
   1. Если сервер не сконфигурирован, нет учетной записи STARTUSER, то Гедымин предлагает ее создать. При этом запрашивается пароль учетной записи SYSDBA. После успешного создания учетной записи STARTUSER попытка подключения повторяется.
   2. Если указанной базы данных нет на сервере, то выдается предупреждающее сообщение и открывается окно выбора базы данных.
   3. При возникновении других ошибок соответствующие сообщения выводятся на экран.
4. Если подключение под учетной записью STARTUSER прошло успешно, то:
   1. если в параметрах командной строки заданы имя пользователя и его пароль, то эти значения используются для аутентификации пользователя. Диалоговое окно запроса пароля на экран не выводится.
   2. если в параметрах командной строки не заданы ни имя пользователя, ни его пароль или задан только один параметр, то из системного реестра считывается имя учетной записи Гедымина, под которой последний раз было осуществлено подключение и на экран выводится окно для ввода пароля;
5. Если необходимо, на экран выводится диалоговое окно для ввода имени пользователя и пароля. В этот момент подключение к базе данных под учетной записью STATUSER по-прежнему активно.
   1. Если в течение трех минут пользователь не вводит имя пользователя и пароль, то диалоговое окно и подключение к базе данных закрываются. Пользователю предлагается либо завершить

выполнение программы, либо продолжить загрузку платформы без подключения к базе данных.

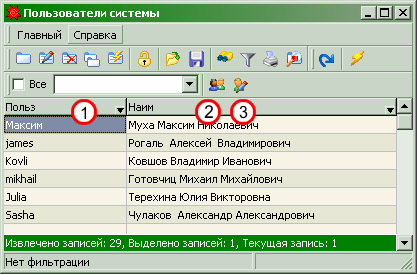
* 1. В процессе работы в диалоговом окне пользователь может выбрать другую базу данных из выпадающего списка, либо зарегистрировать новую базу данных. в любом случае, активное подключение закрывается и повторяются шаги 3-5.

1. Используя активное подключение вызывается процедура GD\_P\_SEC\_LOGINUSER на вход которой передаются: имя учетной записи Гедымин и введенный пользователем пароль;
2. Процедура проверяет введенную информацию. По заданному имени пользователя в таблице GD\_USER отыскивается запись:
   1. Если запись существует, и введенный пользователем пароль совпадает с тем, который хранится в записи, то возвращается имя пользователя Interbase, его пароль, а так же вспомогательная информация.
   2. Если переданы неверные данные или подключение невозможно, то возвращается соответствующий код ошибки. Например, такой учетной записи нет в базе данных, или запись заблокирована или введен неверный пароль и т.п.
3. Закрывается активное подключение к базе данных.
4. Происходит подлючение к базе данных с использованием учетной записи Interbase, ее пароля и роли ADMINISTRATOR.
   1. Если такой учетной записи Interbase нет на сервере, то, возможно, база данных была перенесена на другой сервер или сервер был переустановлен, т.е. файл с паролями isc4.gdb был заменен чистым файлом. Запрашиваем у пользователя пароль SYSDBA и пытаемся пересоздать учетную запись. Если получилось, отражаем изменения в таблице GD\_USER.
   2. Если подключение прошло успешно, то на этом процедура аутентификации завершена. Осуществляется загрузка платформы.

##### Просмотр списка учетных записей

Список учетных записей вызывается командой Пользователи из раздела

«Сервис»—«Администратор» Исследователя системы.



*Рис. 158. Список пользователей системы*

Обратите внимание, что на панели инструментов, кроме набора стандартных команд так же присутствуют:

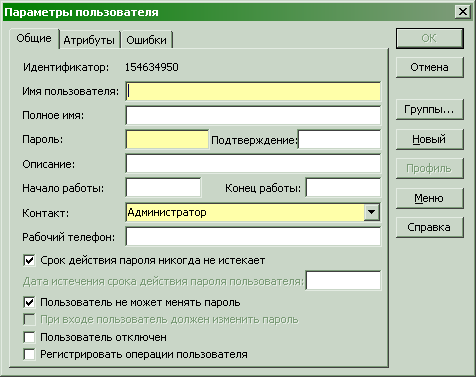
1. фильтр, для отбора пользователей, входящих в определенную группу,
2. команда просмотра списка групп, в которые входит выбранная учетная запись,
3. команда пересоздания учетных записей Interbase для всех учетных записей Гедымин.

Действие, выполняемое последней командой нуждается в пояснении. Выше, мы уже говорили о том, что каждой учетной записи Гедымин соответствует учетная запись сервера Interbase, под которой происходит подключение к базе данных. Поскольку, список пользователей сервера Interbase хранится отдельно от базы данных в файле isc4.gdb, то при переносе базы на другой сервер (или при полной переустановке сервера) необходимо для каждой учетной записи платформы Гедымин создать соответствующую учетную запись сервера Interbase. Проще всего это сделать войдя в систему под учетной записью Administrator (она использует учетную запись SYSDBA, которая, как правило, всегда присутствует на сервере Interbase), открыть окно Пользователи и выполнить команду

«Пересоздать всех пользователей Interbase».

##### Создание новой учетной записи

Для создания новой учетной записи необходимо вызвать соответствующую команду и заполнить поля в диалоговом окне.



*Рис. 159. Диалоговое окно создания новой учетной записи*

Поля, доступные при создании новой учетной записи:

|  |  |
| --- | --- |
| Имя пользователя | Наименование учетной записи  пользователя платформы Гедымин. |
| Полное имя | Полное имя пользователя. |
| Пароль | Пароль, который будет запрашиваться при  входе в систему. |
| Подтверждение пароля | Повторный ввод пароля, для исключения  ошибки. |
| Описание | Произвольный комментарий. |
| Начало работы | Начало временного интервала, в течение которого пользователю разрешено входить  в систему. |
| Конец работы | Окончание временного интервала, в течение которого пользователю разрешено входить в систему. Если оба поля пустые, то ограничений на время входа не  налагается. |
| Контакт | Связь учетной записи с записью в списке контактов (людей). Важно для каждой учетной записи завести отдельную запись в справочнике людей (сотрудников предприятия). При создании или изменение записей в базе данных  регистрируется кто произвел операцию. |

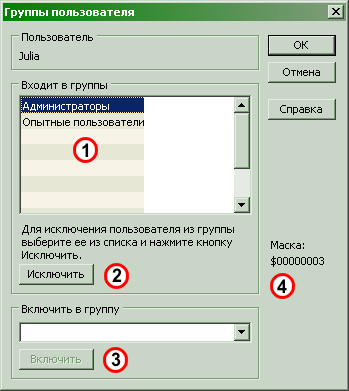
|  |  |
| --- | --- |
|  | Причем фиксируется не ссылка на учетную запись60, а ссылка на контакт, который связан с этой учетной записью. Поэтому, если несколько учетных записей ссылаются на одного и того же человека, то в последствии может возникнуть путаница  с определением кто и что изменял. |
| Рабочий телефон | Поле только для чтения. Заполняется автоматически телефонным номером, взятым из записи контакта, связанного с  данной учетной записью. |
| Срок действия пароля никогда не истекает | Если отмечено, то пароль действует всегда, пока существует данная учетная запись.  Если не отмечено, то необходимо указать дату, до которой действует введенный пароль. При очередном входе в систему, если срок действия пароля истек, пользователю будет предложено ввести  новый пароль. |
| Дата истечения срока действия пароля пользователя | Указывается дата, до которой действует пароль пользователя. Поле доступно только если флажок Срок действия пароля  никогда не истекает. |
| Пользователь не может менять пароль | Если флажок установлен, то пользователь не может самостоятельно изменить свой  пароль. |
| При входе пользователь  должен изменить пароль | Если флажок установлен, то при входе  пользователь должен ввести новый пароль. |
| Пользователь отключен | Если флажок установлен, то данный  пользователь не может войти в систему. |
| Регистрировать операции пользователя | Если флажок установлен и включен аудит действий пользователя, то все операции (действия) совершенные пользователем  регистрируются в системном журнале. |

60 Очевидно, что учетная запись — информация не постоянная. Сегодня человек работает на предприятии, а завтра уволился и системный администратор удалил его учетную запись из базы.

В правой части окна находится кнопка «Группы…» с помощью которой можно просмотреть список групп, в которые входит данный пользователь, изменить его.

##### Просмотр/изменение списка групп пользователя

Находясь в окне «Группы пользователя» можно просмотреть список групп, в которые входит выбранный пользователь, либо изменить его.



*Рис. 160. Добавление пользователя в группу*

Для того, чтобы исключить пользователя из группы, следует выбрать нужную группу в списке (1) и выбрать кнопку «Исключить» (2).

Для того, чтобы включить пользователя в группу, следует выбрать нужную группу из выпадающего списка внизу окна и выбрать кнопку

«Включить» (3).

В правой части окна выводится шестнадцатиричное (4) число, соответствующее битовой маске групп, в которые входит пользователь.

##### Системные группы пользователей

Ниже приводится список из шести системных групп пользователей, имеющих предопределенное значение. Системную группу нельзя удалить.

###### Администраторы

Имеют максимальные права доступа ко всем функциям и данным системы. Единственное, чего не может сделать пользователь, входящий в группу Администраторы — это изменять структуру базы данных и, соответственно, активизировать настройки. Для этого предназначена учетная запись Administrator. Идентификатор группы 1.

###### Опытные пользователи

Пользователи с расширенными полномочиями. Могут изменять отчеты в режиме просмотра, перед отправкой их на печать. Идентификатор группы 2.

###### Пользователи

Простые пользователи системы. Идентификатор группы 3.

###### Операторы архива

Операторы архивного копирования. На равне с администраторами имееют право создавать архивные копии базы данных и производить восстановление из архива. Идентификатор группы 4.

###### Операторы печати

Операторы печати имеют право изменять отчеты в режиме просмотра перед отправкой их на печать. Идентификатор группы 5.

###### Гости

Гости системы имеют наименьшие полномочия.

#### Разграничение доступа к функциям системы

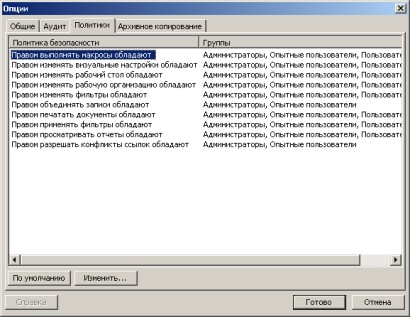
Доступ к некоторым функциям системы контролируется непосредственно в коде платформы. Например, открывать редактор скрипт-объектов или SQL редактор могут только участники группы Администраторы. Создавать архивные копии базы данных — Администраторы и Операторы архива и т.д. Более подробно, с предназначением каждой из системных групп можно ознакомиться в разделе «Системные группы пользователей».

Управлять доступом к остальным функциям можно с помощью политик групповой безопасности.

##### Политики групповой безопасности

Подобно тому как в операционной системе Windows можно назначить права как на отдельные файлы и папки, так и на функции операционной системы с помощью политик групповой безопасности, в Гедымине, так же можно разграничивать доступ к основным функциям платформы. Для этого предназначены политики групповой безопасности или просто политики безопасности.

Для просмотра списка политик необходимо открыть окно Опции в меню Сервис главного окна программы и перейти на вкладку Политики.



*Рис. 161. Политики групповой безопасности*

Как видно на рисунку вкладка содержит список доступных политик и список групп пользователей, на которых распространяется конкретная политика безопасности. Внизу списка располагаются две кнопки:

 По умолчанию — позволяет вернуть установки для данной политики к стандартным значениям, поставляемым вместе с изначальной базой данных;

 Изменить… — открывает окно, в котором можно указать на какие группы пользователей системы распространяется выбранная политика;

Доступны следующие политики групповой безопасности:

###### Правом выполнять макросы обладают

Только пользователи, входящие в указанные группы будут обладать правом на выполнение макросов из меню Макросы на форме просмотра.

В процессе настройки платформы может возникнуть ситуация, когда определенной группе пользователей необходимо дать доступ на выполнение только одного-двух макросов и запретить доступ ко всем остальным. Сделать это можно двумя способами:

1. Назначить права доступа для конкретных макросов в окне редактора скрипт объектов. В этом случае политику групповой безопасности задействовать не нужно;
2. Используя политику групповой безопасности запретить группе доступ к макросам в меню Макросы. Открыть на редактирование нужную форму и разместить дополнительную кнопку (кнопки) на панели инструментов для вызова нужных макросов. Присвоив событию OnClick этих кнопок следующий код:

Код…

Очевидно, что если первый способ более быстрый, то последний более гибкий и предоставляет гораздо больше возможностей. Например, в коде можно установить проверку не на группу пользователей, а на конкретную учетную запись, разрешив или запретив ей доступ на выполнение макроса.

###### Правом изменять визуальные настройки обладают

Данная политика определяет какие группы пользователей имеют право

на:

 Изменение внешнего вида формы в дизайнере (вызывается по коомбинации клавиш Ctrl-Alt-E);

 Изменение визуальных настроек таблицы (соответствующий мастер вызывается по нажатию на F10 в таблице);

 Копированию настроек формы от другого пользователя системы.

###### Правом изменять рабочий стол обладают

Данная политика определяет какие группы пользователей имеют право

на изменение рабочего стола, т.е. на закрытие или сворачивание форм, входящих в рабочий стол, создание новых рабочих столов либо удаление существующих.

###### Правом изменять рабочую организацию обладают

Как следует из названия политики, только указанные группы пользователей будут обладать правом выбрать рабочую организацию из выпадающего списка на главной форме Гедымина.

###### Правом изменять фильтры обладают

Только указанные группы пользователей будут иметь возможность создавать, удалять или изменять фильтры.

###### Правом объединять записи обладают

Только указанные группы пользователей будут обладать правом объединения записей.

###### Правом печатать документы обладают

Только указанные группы пользователей будут обладать правом отправлять документы на печать или экспортировать их в другие программы.

###### Правом применять фильтры обладают

Только указанные группы пользователей будут обладать правом выбрать и применить фильтр.

###### Правом просматривать отчеты обладают

Только указанные группы пользователей будут обладать правами на просмотр отчетов перед отправкой их на печать.

###### Правом разрешать конфликты ссылок обладают

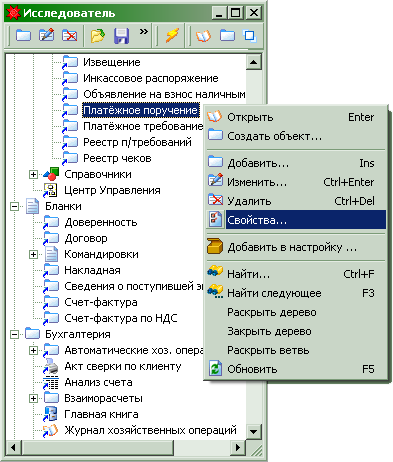
Только указанные группы пользователей будут обладать правом разрешать конфликты ссылок в базе данных. Конфликты ссылок возникают, когда удаляется запись и в базе данных присутствуют другие записи, ссылающиеся на удаляемую.

###### Правом работать с эквивалентом

Только пользователи указанных групп будут иметь возможность просматривать и изменять значения сумм в эквиваленте в документах складского учета.

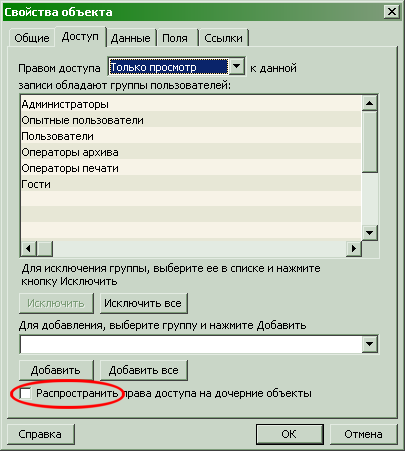
#### Разграничение доступа к бизнес классам

С помощью Исследователя системы можно разграничить уровни доступа для конкретного бизнес класса или группы классов. Для этого необходимо установить курсор на наименование класса и с помощью правой кнопки мыши открыть контекстное меню.



*Рис. 162. Команда Свойства*

В нем следует выбрать команду Свойства, как показано на рисунке выше. На экране появится диалоговое окно «Свойство объекта». Для установки прав доступа следует перейти на вкладку «Доступ».



*Рис. 163. Назначение прав доступа*

В верхней части вкладки находится выпадающий список с тремя уровнями доступа:



Только просмотр; Просмотр и изменение; Полный доступ;

В таблице, которая занимает основную часть вкладки отображается список групп пользователей обладающих указанным правом. Чтобы исключить группу из списка, следует установить на нее курсор и нажать кнопку «Исключить», которая располагается снизу таблицы. С помощью команды «Исключить все» можно быстро очистить список, оставив в нем только группу Администраторы, которая всегда обладает максимальными полномочиями.

Добавление группы в список осуществляется с помощью выпадающего списка групп и кнопок «Добавить» и «Добавить все». Первая кнопка добавляет выбранную группу, а вторая — все группы пользователей.

Обратите внимание на флаг «Распространить права доступа на дочерние объекты» в нижней части вкладки. Если флаг установлен, то права будут применены не только к выбранному объекту, но и ко всем вложенным объектам и папкам в Исследователе. Таким образом можно быстро настроить права доступа для группы классов. Например, вместо того, чтобы по отдельности устанавливать идентичные права доступа на

«Платежное поручение», «Платежное требование», «Приходный кассовый ордер» и т.п. Можно установить курсор на ветку «Банк и касса» и настроить права доступа для нее с распространением на все дочерние объекты.

Команды, для которых у текущего пользователя нет вообще никакого доступа, в Исследователе не отображаются. Если у пользователя нет доступа на некоторую папку, но есть хотя бы минимальные права на входящие в эту

папку команды, то они будут отображены в Исследователе, однако древовидная структура при этом будет нарушена. Обычно, такие команды будут выведены просто списком на самом верхнем уровне ирерархии. По этой причине, после настройки прав доступа в Исследователе, следует зайти в Гедымин под пользовательской учетной записью и убедиться, что окно Исследователя отображает корректную информацию.

**Назначение прав доступа на отдельные поля**

#### Разграничение доступа к данным

Настраивая доступ в Исследователе мы разрешаем или запрещаем определенные операции для всех записей одного класса. Но, что если для определенной группы требуется открыть доступ к части записей и закрыть к остальным? Гедымин предоставляет такую возможность, т.е. позволяет разграничивать права доступа на уровне отдельных записей в наборе данных.

###### Дескрипторы безопасности

Каким образом обеспечивается разграничение доступа для отдельных записей? Как известно, для каждого бизнес-класса существует таблица, которая содержит его данные. Если для этого класса поддерживается разграничение доступа на уровне записей, то в таблице будет присутствовать одно, два или три специальных поля, т.н. дескрипторы безопасности. Дескриптор безопасности — это целое число, содержащее битовую маску групп пользователей, обладающих определенным правом. Каким? Это зависит от наименования дескриптора безопасности:

AFULL Полный доступ;

ACHAG Доступ на просмотр и изменение данных (удаление и изменение прав доступа запрещены);

AVIEW Доступ только на просмотр данных.

Для того, чтобы скрыть от пользователя записи на просмотр которых у него нет прав, в SQL запрос для извлечения данных бизнес-класса добавляется следующее условие:

WHERE

G\_SEC\_TEST ( Z.AVIEW, - 1 ) <> 0

Второй параметр функции G\_SEC\_TEST — это целое число, представляющее битовую маску групп, в которые входит текущий пользователь. Так, например, если пользователь является только членом

группы с идентификатором 3 («Пользователи»), то в качестве параметра функции будет передаваться значение 2^(3-1)=2^2=4. Если текущий пользователь входит в группу «Администраторы», то в качестве параметра будет передаваться значение -1 (все 32 бита установлены).

Если заглянуть внутрь функции G\_SEC\_TEST, то мы увидим две очень простые операции:

G\_SEC\_TEST(X, Y) = (X OR 1) AND Y

Если с двоичным умножением (AND) все понятно — мы пытаемся определить входит ли пользователь хотя бы в одну группу из тех, которым разрешен доступ. То присутствие здесь оператора OR требует пояснения. Ранее, мы отмечали, что члены группы «Администраторы» всегда имеют доступ ко всем данным, поэтому перед выполнением побитового умножения, следует установить бит, соответствующий группе

«Администраторы». Теперь, даже если доступ установлен только для группы «Пользователи», а текущий пользователь входит только в группу

«Администраторы», функция G\_SEC\_TEST вернет значение отличное от 0 и доступ к записи будет открыт.

###### Инициализация дескрипторов безопасности

При создании новой записи дескрипторам безопасности присваиваются значения, установленные для этого класса в Исследователе. Инициализация полей происходит в методе OnNewRecord. Перекрыв данный метод, можно присвоить дескрипторам другие значения. Например, чтобы разрешить просмотр записи только пользователям из группы, в которую входит пользователь, создавший ее, следует после вызова наследованного метода вставить следующую строку:

…

Self.FieldByName(‘aview’).AsInteger = IBLogin.InGroup

…

Если добавить такую строку к коду метода OnNewRecord документа

«Счет-фактура» и разделить менеджеров на группы по региональному признаку, например, «Минск», «Брест», «Могилев» и т.д., то менеджеры каждого города будут видеть только свои счета.

###### Почему в Гедымине не используются представления для разграничения прав доступа?

У разработчика, хорошо знакомого с возможностями языка SQL может возникнуть вопрос: почему для разграничения доступа на уровне

отдельных колонок и на уровне записей в Гедымине не используются представления (view), как это рекомендовано стандартом SQL?

Ответов на такой вопрос может быть несколько:

1. Если использовать представления для разграничения прав доступа на колоноки таблицы, то каждый раз, когда администратор захочет разграничить права для той или иной группы придется создавать отдельное представление.
2. Как правило, данные бизнес объекта хранятся не в одной, а в нескольких таблицах. Если представление основано на запросе из нескольких таблиц, то оно не может обновляться (read-only). И для модификации данных придется либо создавать отдельные триггеры, либо реализовывать иные нетривиальные механизмы.

##### Изменение прав доступа с помощью SQL

Мы добавили новую группу пользователей и хотим открыть ей доступ ко всем документам. Можно, конечно, заходить в каждый из документов, выбирать все записи в списке и устанавливать для них права, а можно выполнить задание гораздо быстрее. Для этого следует напрямую подкорректировать значения дескрипторов безопасности в таблице GD\_DOCUMENT. С помощью функций BIN\_AND и BIN\_OR мы можем снимать или устанавливать нужные биты.

Пусть идентификатор добавленной группы равен 22. Битовая маска с одним установленным двадцать вторым битом соответствует целому значению 2^(22-1)=2^21=2097152.

Перейдем в редактор SQL и выполним следующую команду:

UPDATE GD\_DOCUMENT SET AVIEW=BIN\_OR(AVIEW, 2097152), ACHAG=BIN\_OR(ACHAG, 2097152), AFULL=BIN\_OR(AFULL, 2097152)

Если позже мы передумаем, то отозвать право доступа у группы можно с помощью команды:

UPDATE GD\_DOCUMENT SET AVIEW=BIN\_AND(AVIEW, BIN\_XOR(2097152, 0)), ACHAG=BIN\_AND(ACHAG, BIN\_XOR(2097152, 0)),

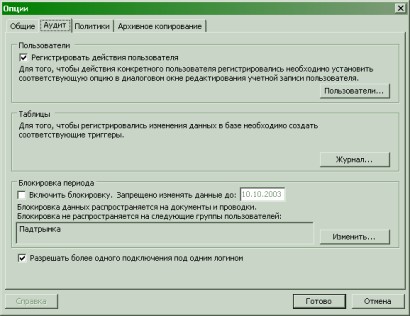
AFULL=BIN\_AND(AFULL, BIN\_XOR(2097152, 0))

#### Аудит действий пользователя

Механизмы разграничения прав доступа позволяют не дать пользователю доступ к определенным данным или право на выполнение определенных операций. Но что если и права вроде настроены правильно, а в системе происходит что-то неладное? Нужные документы «исчезают» или

«самопроизвольно» изменяются и все пользователи отрицают свою причастность к этому? Помочь может журнал событий, в котором регистрируются операции, выполненные пользователями и изменения данных в базе. Обе опции выключены по умолчанию. Для того, что бы включить аудит действий пользователя необходимо: 1) в главном окне Гедымина выбрать меню Сервис и в нем команду Опции, в появившемся окне перейти на вкладку Аудит и установить флажок Регистрировать действия пользователя; 2) перейти в список пользователей системы, нажав кнопку Пользователи, открыть на редактирование каждую учетную запись, для которой мы хотим активизировать аудит и установить флажок

«Регистрировать действия пользователя».



*Рис. 164. Аудит действий пользователя*

После этого, такие выполненные действия как: вход пользователя в систему, открытие и закрытие окон, создание, изменение, удаление объектов будут регистрироваться в журнале.

###### Просмотр журнала событий

Открыть журнал операций для просмотра можно через исследователь системы. Команда «Просмотр событий» находится в разделе «Сервис»—

«Администратор».

###### Очистка журнала событий

**Регистрация событий из макросов**

Запись дополнительных событий из макросов производится с помощью метода AddEvent объекта IBLogin.

#### Аудит изменения данных

Регистрация изменений данных может быть полезна в следующих случаях:



Необходимо установить контроль за действиями пользователей; Необходимо найти логическую ошибку в коде программы (настройки), которая проявляется в формировании неверных данных;

 Необходимо выполнить профилирование и оптимизацию программного кода, выявить наиболее часто выполняющиеся, неоптимальные участки;

 Необходимо отслеживать изменения в данных, производимые внешними программами.

###### Включение регистрации изменений

Для того, чтобы включить регистрацию изменений необходимо открыть Просмотр событий в разделе Администратор Исследователя. Далее в контекстном меню выбрать команду Создать триггеры.

Система предложит создать триггеры на все таблицы базы данных или на часть таблиц. Если выбрано второе, то откроется окно для выбора необходимых таблиц из списка.

Для каждой таблицы, изменения в данных которой будут фиксироваться, будут созданы три триггера: после вставки записи, после изменения записи и после удаления записи.

Триггеры будут заносить данные в таблицу GD\_JOURNAL. При этом поле SOURCE будет заполнено именем таблицы, а поле DATA будет содержать наименование операции: "Вставка записи", "Удаление записи", "Изменение записи". Для изменения записи дополнительно будет занесено какие поля изменились и каким образом. Изменения полей фиксируются для полей всех типов, кроме БЛОБ полей.

Если таблица, является главной таблицей бизнес-объекта, т.е. имеет поле ID, то для такой таблицы в GD\_JOURNAL, в поле OBJECTID будет заносится идентификатор записи (объекта) данные которого были изменены.

Триггеры не создаются для следующих таблиц:

1. системных таблиц (имя начинается с RDB$);
2. таблиц репликации (имя начинается с RPL);
3. таблицы GD\_JOURNAL;

###### Выключение регистрации изменений

Для того, чтобы отключить регистрацию изменений данных необходимо зайтив Просмотр событий в разделе Администратор Исследователя и выбрать в контекстном меню команду "Удалить триггеры".

###### Изменение структуры базы и регистрация

Перед изменением структуры базы данных необходимо отключить регистрацию изменений данных.

##### Блокировка периода

Блокировка периода распространяется на следующие таблицы:

ac\_entry бухгалтерские проводки; gd\_document документы;

inv\_card карточки складского учета; inv\_movement складское движение.

На каждую из этих таблиц в базе данных созданы по три триггера для отслеживания операций вставки, изменения и удаления записи. Триггеры не активны, если блокировка периода выключена и активизируются при ее включении.

В своем коде каждый триггер использует значения двух генераторов: gd\_g\_block и gd\_g\_block\_group. Первый генератор содержит приведенную к целому числу дату, до кототорой блокируются данные, а второй — битовую маску групп пользователей, на которых не распространяется блокировка.

###### Триггер блокировки периода

Поскольку, все двенадцать триггеров однотипны, мы приведем здесь только один из них для того, чтобы проиллюстрировать как работает механизм блокировки периода.

CREATE TRIGGER AC\_BI\_ENTRY\_BLOCK FOR AC\_ENTRY INACTIVE

BEFORE INSERT POSITION 28017

AS

DECLARE VARIABLE IG INTEGER; BEGIN

IF (CAST(NEW.entrydate AS INTEGER) < GEN\_ID(gd\_g\_block, 0)) THEN

BEGIN

IF (GEN\_ID(gd\_g\_block\_group, 0) = 0) THEN BEGIN

EXCEPTION gd\_e\_block;

END ELSE BEGIN

SELECT ingroup FROM gd\_user WHERE ibname = CURRENT\_USER

INTO :IG;

IF (BIN\_AND(GEN\_ID(gd\_g\_block\_group, 0), :IG) = 0)

THEN

BEGIN

EXCEPTION gd\_e\_block; END

END END

END

Обратите внимание на большой номер позиции триггера — 28017 — который гарантирует, что данный триггер будут выполняться после других триггеров изменяющих данные записи. Может возникнуть вопрос, почему бы в качестве номера позиции триггера не использовать максимально допустимое сервером Interbase число? Сделано это специально для того, чтобы оставить пространство для триггеров (например, участвующих в механизме репликации), которые должны выполняться еще позже, чем триггер контроля блокировки периода.

Алгоритм, реализованный в триггере можно описать следующим образом: сначала проверяется дата записи. Если она попадает в заблокированный период, то проверим заданы ли группы пользователей, на которых блокировка не распространяется. Если такие группы не заданы, то генерируется исключение. Если группы заданы, то из таблицы GD\_USER извлечем битовую маску групп, которым принадлежит текущий пользователь. Если ни одна из групп пользователя не входит во множество групп, на которых не распространяется блокировка, то генерируем исключение.

##### Обеспечение безопасности в случае локальной установки

При локальной инсталляции платформы Гедымин используется встроенный сервер Interbase, который допускает подключение под любой учетной записью. Точнее будет сказать, что при подключении к встроенному серверу учетная запись не проверяется вообще и любой подключившийся пользователь обладает максимальными правами доступа.

#### Сложные сценарии разграничения прав доступа

Как мы уже говорили выше, Гедымин позволяет заблокировать изменение документов, а так же бухгалтерских проводок и складского движения, оформленных до указанной даты. При этом, выборочно, могут быть заданы группы пользователей, на которых блокировка не распространяется. Встроенная функциональность системы не позволяет заблокировать период только для документов определенного типа. Попробуем преодолеть это ограничение. Рассмотрим следующую задачу:

##### Задача

На некотором предприятии, занимающемся оптовой торговлей товарами народного потребления, пользователи системы разделены на три группы:



Руководство, Бухгалтерия, Менеджеры.

Предприятие использует стандартные настройки из поставки «Оптово- розничный склад». В документообороте участвуют следующие документы:



Счет-фактура,

Накладная на отпуск товара на сторону.

Конечно, наш пример достаточно условен. Реальное предприятие использует гораздо больше видов документов, но наша задача — научиться разграничивать права доступа и для ее выполнения будет вполне достаточно и счетов-фактур с накладными.

Необходимо разграничить права доступа следующим образом: Руководство имеет право только просматривать все документы; Бухгалтерия имеет полный доступ ко всем документам;



 Менеджеры имеют полный доступ к документам в текущем месяце. В более ранних периодах они имеют только доступ на изменение к счетам-фактурам и не имеют никакого доступа к накладным.

##### Решение

Блокировать изменение документов можно как на уровне базы данных, так и на уровне Гедымина. В первом случае необходимо создать тригеры на таблицы документов, проводок и складских остатков и складского движения, которые будут вызываться перед выполнением операций изменения, добавления или удаления данных и, при нарушении определенного условия, блокировать их. Во втором случае, следует перекрыть методы у бизнес-класса документа, для того, чтобы предотвратить нежелательное изменение данных. Каждый способ имеет свои положительные и отрицательные стороны. Реализация блокировки на уровне базы данных более трудоемкая задача, так как требует создания двенадцати тригеров: по одному тригеру для каждой из трех операций изменения данных для четырех таблиц, содержащих документы, бухгалтерские проводки складские остатки и складское движение61. Кроме этого, в случае реализации блокировки только на уровне базы данных, нет никакой возможности взаимодействия с пользователем до того, как он приступит к редактированию документа или в процессе этого редактирования. Т.е. пользователь сможет открыть диалоговое окно, отредактировать данные и только в момент сохранения их на экран будет выведено сообщение о том, что данный документ заблокирован и не может быть изменен. Сильной стороной блокировки на уровне базы данных является то, что данные будут защищены вне зависимости от того, как они изменяются: через методы бизнес-объекта, с помощью выполнения SQL запросов в макросах Гедымина или через внешнюю программу, такую, например, как IBExpert или EMS Interbase Manager. При реализации блокировки на уровне Гедымина мы сможем предупредить пользователя о том, что он может только просматривать данные и запретить нажатие кнопки «Ок» в диалоговом окне. Но, с другой стороны, не сможем запретить изменение данных напрямую из макроса или внешней программы.

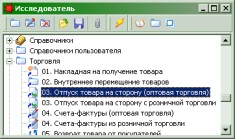
61 Стоит заметить, что в версии сервера Firebird 2.0 будут предусмотрены т.н. универсальные тригеры, вызываемые на все операции изменения для заданной таблицы. В этом случае, очевидно, количество необходимых тригеров для решения поставленной задачи может быть уменьшено с 12 до 4.

Для решения данной задачи мы выберем смешанный вариант, реализовав код блокировки как в тригерах в базе данных, так и в коде макросов системы Гедымин.

Внимательно ознакомившись с условиями задачи мы можем увидеть, что разграничение прав доступа для групп Руководство (только просмотр) и Бухгалтерия (полный доступ) не требут дополнительного программирования и может быть реализовано имеющимися средствами платформы.

###### Разграничение прав для групп Руководство и Бухгалтерия

Откроем Исследователь системы. Найдем в нем раздел Торговля и установив курсор на документ, именуемый как «03. Отпуск товара на сторону (оптовая торговля)».



*Рис. 165. Вызов команды в Исследователе*

Нажмем правую кнопку мыши и из появившегося меню выберем команду «Свойства…». В появившемся диалоговом окне перейдем на вкладку «Доступ» и настроим права следующим образом:

 Только просмотр — Администраторы, Руководство, Бухгалтерия и Менеджеры;

 Просмотр и изменение — Администраторы, Бухгалтерия, Менеджеры;

 Полный доступ — Администраторы, Бухгалтерия и Менеджеры.

Расстановку прав доступа рекомендуется производить именно в таком порядке: от менее сильных прав к более сильным.

Аналогично настроим права доступа для документа «04. Счета-фактуры (оптовая торговля)».

###### Ограничение доступа для группы Менеджеры

Займемся теперь менеджерами. Задачу разграничения прав доступа для этой группы разобьем на следующие подзадачи:

 Запретить добавление и удаление счетов-фактур за пределами текущего месяца;

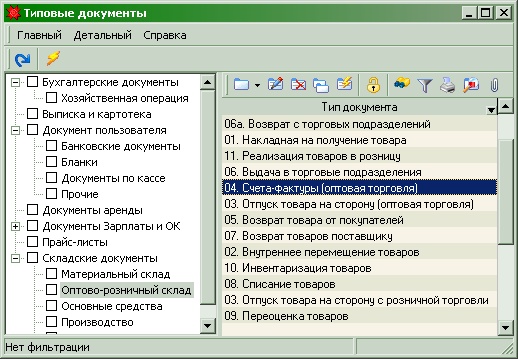
 Запретить добавление, изменение и удаление накладных за пределами текущего месяца;

 Запретить просмотр накладных за пределами текущего месяца.

Первые два пункта из нашего списка можно решить, наложив ограничения в тригерах на таблице GD\_DOCUMENT.

Прежде чем перейти к созданию тригеров определим идентификаторы типов документов и групп пользователей. Для получения идентификаторов типов перейдем в Исследователь и в разделе «Сервис» выберем команду

«Типовые документы». В древовидном списке, в левой части окна найдем раздел «Оптово-розничный склад». В правой части, установим курсор на позицию «04. Счета-фактуры (оптовая торговля)» как показано на следующем рисунке.



*Рис. 166. Выбор типового документа*

Нажмем правую кнопку мыши и выберем команду «Свойства…». Выпишем значение идентификатора записи. Аналогичную операцию проделаем для документа «03. Отпуск товара на сторону (оптовая торговля)». У нас получились следующие значения:



Счет-фактура — 147043282;

Накладная — 147043395.

Обратите внимание, что значения идентификаторов на каждой базе данных будут отличаться от приведенных выше! Поэтому, выполняя наш пример на своей базе данных вы получите другие значения. Неизменными от базы к базе будут РУИДы этих записей, но об их использовании мы поговорим отдельно.

Для получения идентификаторов групп пользователей перейдем в раздел «Исследователь»—«Сервис»—«Администратор» и выберем команду

«Группы пользователей». Так же, как и для документов, воспользуемся командой «Свойства…» из контекстно-зависимого меню. Наши значения идентификаторов:



Руководство — 7;

Менеджеры — 9;

Как и в случае с идентификаторами типов документов, значения полученные на вашем экземпляре базы данных скорее всего будут отличаться от полученных нами.

Собственно, использовать непосредственно идентификаторы групп в тригерах мы не сможем. Нам нужна битовая маска, которую мы вычислим следующим образом:

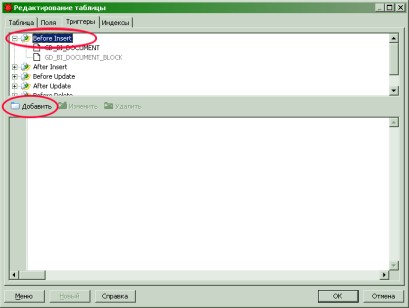
27-1+29-1=26+28=64+256=320

В рассматриваемой нами задаче необходимо запретить менеджерам и представителям группы руководство вставку и удаление счетов-фактур и накладных за пределами текущего месяца. Кроме этого, руководству запрещается изменение счетов и накладных за пределами текущего месяца, а менеджерам запрещается изменение только накладных, но разрешается изменение счетов.

Создадим три тригера на таблице GD\_DOCUMENT, которые будут вызываться, соответственно, перед вставкой, изменением и удалением записи. Если дата документа выходит за пределы текущего месяца и документ является счетом-фактурой или накладной и пользователь входит хотя бы в одну из «запрещенных» групп, то сгенерируем исключение (EXCEPTION). Обратите внимание, что мы не будем создавать своего объекта исключения, а воспользуемся GD\_E\_BLOCK — объектом присуствующим в эталонной конфигурации базы данных.

Для создания тригера необходимо открыть Исследователь системы. Далее последовательно откроем раздел «Сервис» и, находящийся в нем раздел «Атрибуты». Вызовем команду «Таблицы». На экране откроется список таблиц базы данных. Найдем в нем GD\_DOCUMENT и откроем данную таблицу на редактирование в диалоговом окне. Перейдем на вкладку «Триггеры». В верхней части вкладки установим курор на позицию

«Before Insert» и нажмем кнопку «Добавить».



*Рис. 167. Создание нового триггера*

На экране откроется окно создания нового триггера.



*Рис. 168. Текст триггера*

Введем текст триггера и нажмем «Ок». Вышеуказанные операции повторим последовательно для создания триггеров перед изменением и перед удалением записи. Ниже приводятся исходные коды всех трех триггеров. Будьте внимательны! Они не идентичны. В триггере перед обновлением записи мы проверяем пользователя только на вхождению в группу Руководство (битовая маска 256), а в триггере перед удалением мы используем мета-переменную OLD, а не NEW.

**Триггер Before Insert**

AS

DECLARE VARIABLE D DATE; DECLARE VARIABLE G INTEGER;

BEGIN

D = CAST(’01.’ || EXTRACT(MONTH FROM CURRENT\_DATE) || ‘.’

||

EXTRACT(YEAR FROM CURRENT\_DATE) AS DATE);

IF (NEW.documentdate < :D) THEN BEGIN

IF (NEW.documenttypekey IN (147043282, 147043395)) THEN BEGIN

SELECT FIRST 1 ingroup FROM gd\_user WHERE ibname = CURRENT\_USER

INTO :G;

IF (BIN\_AND(:G, 320) <> 0) THEN BEGIN

EXCEPTION gd\_e\_block; END

END END

END

**Триггер Before Update**

AS

DECLARE VARIABLE D DATE; DECLARE VARIABLE G INTEGER;

BEGIN

D = CAST(’01.’ || EXTRACT(MONTH FROM CURRENT\_DATE) || ‘.’

||

EXTRACT(YEAR FROM CURRENT\_DATE) AS DATE);

IF (NEW.documentdate < :D) THEN BEGIN

IF (NEW.documenttypekey IN (147043282, 147043395)) THEN BEGIN

SELECT FIRST 1 ingroup FROM gd\_user WHERE ibname = CURRENT\_USER

INTO :G;

IF (BIN\_AND(:G, 64) <> 0) THEN BEGIN

EXCEPTION gd\_e\_block; END

IF ((BIN\_AND(:G, 256) <> 0)

AND (NEW.documenttypekey = 147043395)) THEN BEGIN

EXCEPTION gd\_e\_block; END

END END

END

**Триггер Before Delete**

AS

DECLARE VARIABLE D DATE; DECLARE VARIABLE G INTEGER;

BEGIN

D = CAST(’01.’ || EXTRACT(MONTH FROM CURRENT\_DATE) || ‘.’

||

EXTRACT(YEAR FROM CURRENT\_DATE) AS DATE);

IF (OLD.documentdate < :D) THEN BEGIN

IF (OLD.documenttypekey IN (147043282, 147043395)) THEN BEGIN

SELECT FIRST 1 ingroup FROM gd\_user WHERE ibname = CURRENT\_USER

INTO :G;

IF (BIN\_AND(:G, 320) <> 0) THEN BEGIN

EXCEPTION gd\_e\_block; END

END END

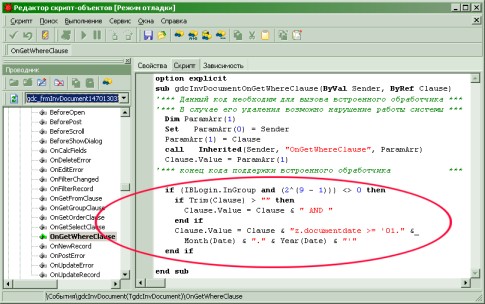
END

Создание триггеров в базе данных произойдет только после того, как будет закрыто по кнопке «Ок» окно редактирования таблицы. При этом на экран будет выведено окно с ходом выполнения соответствующих SQL команд.

Осталось только скрыть от менеджеров накладные за пределами текущего месяца. Для этого, при формировании запроса для извлечения из базы данных списка накладных мы будем проверять входит ли текущий пользователь в группу «Менеджеры» и если да, то добавлять в часть WHERE запроса условие для отбора записей только после первого числа текущего месяца.

Реализуем вышеописанный алгоритм:

1. Откроем форму просмотра накладных;
2. Переведем ее в режим дизайнера нажав одновременно Ctrl-Alt-E на клавиатуре;
3. Найдем на форме компонент gdcInvDocument и выделим его;
4. В Инспекторе объектов перейдем на вкладку События;
5. Найдем в списке событие OnGetWhereClause и создадим обработчик для него;
6. В сгенерированный системой код обработчика добавим свои строки, как показано на рисунке ниже.



*Рис. 169. Текст обработчика события*

Полный текст обработчика события приведен ниже. Добавленый нами код достаточно тривиален. Мы проверяем не установлен ли в битовом наборе групп в которые входит текущий пользователь бит, соответствующий группе «Менеджеры», если установлен, то добавляем к условиям секции WHERE ограничение на дату документа. При этом, если секция WHERE уже содержала некоторые условия, то присоединим новое условие через логический оператор AND.

option explicit

sub gdcInvDocumentOnGetWhereClause(ByVal Sender, ByRef Clause)

'\*\*\* Данный код необходим для вызова встроенного обработчика

\*\*\*

'\*\*\* В случае его удаления возможно нарушение работы системы

\*\*\*

Dim ParamArr(1)

Set ParamArr(0) = Sender ParamArr(1) = Clause

call Inherited(Sender, "OnGetWhereClause", ParamArr) Clause.Value = ParamArr(1)

'\*\*\* конец кода поддержки встроенного обработчика

\*\*\*

if (IBLogin.InGroup and 2^(9 - 1)) <> 0 then if Trim(Clause) > "" then

Clause = Clause & " AND " end if

Clause = Clause & "z.documentdate >= '01." &\_ Month(Date) & "." & Year(Date) & "'"

end if end sub

###### Тестирование

Осталось протестировать, созданный нами код. Для этого следует по- очередно войти в Гедымин под учетными записями Руководителя, Бухгалтера и Менеджера и попробовать создать, изменить и удалить документ как в рамках текущего месяца и за более ранние даты. Если все было сделано правильно, то при попытке выполнения недозволенной операции на экран должно быть выдано сообщение о возникновении исключения.

# Приложения

## Список горячих клавиш

**Метапеременные в SQL запросах**

Метапеременная - это строковое выражение, имеющее определенный формат. Метапаременная может быть указана в любом месте SQL запроса. Перед выполнением запроса система Гедымин вместо метапеременных вставит в текст запроса соответствующие значения.

В системе Гедымин можно использовать следующие метапеременные в SQL запросах:

COMPANYKEY CONTACTKEY INGROUP RUID HOLDINGLIST

##### COMPANYKEY

Идентификатор текущей (рабочей) компании.

###### Синтаксис:

<COMPANYKEY/>

**Пример использования:**

данный запрос извлечет из базы данных все банковские счета текущей организации:

select \* from gd\_companyaccount where companykey =

<COMPANYKEY/>

##### CONTACTKEY

Идентификатор текущего контакта (т.е. идентификатор контакта, соответствующего учетной записи текущего пользователя).

**Синтаксис:**

<CONTACTKEY/>

##### INGROUP

Битовый набор групп, в которые входит текущий пользователь.

**Синтаксис:**

<INGROUP/>

##### RUID

Возвращает идентификатор записи в текущей базе данных по ее РУИДу.

**Синтаксис:**

<RUID XID="xid\_number" DBID ="dbid\_number"/>

##### HOLDINGLIST

Возвращает список ключей компаний, входящих в текущую рабочую организацию, если она является холдингом. Список так же содержит ключ самой рабочей организации. Если текущая рабочая организация не является холдингом, то возвращается ее идентификатор.

**Синтаксис:**

<HOLDINGLIST/>