

Генератор отчетов FastReport 3.0

Руководство пользователя

Редакция 1.01

Copyright (c) 1998-2004, Fast Reports Inc.

Содержание

Дизайнер.....	5
Клавиши управления.....	6
Управление мышью.....	6
Панели инструментов.....	7
Панель режимов дизайнера.....	7
Панель инструментов "Стандартная".....	8
Панель инструментов "Текст".....	9
Панель инструментов "Прямоугольник".....	10
Панель инструментов "Выравнивание".....	10
Опции дизайнера.....	11
Параметры отчета.....	12
Параметры страницы.....	13
Построение отчетов.....	15
Компонент TfrxReport.....	15
Объекты отчета.....	15
Отчет "Hello, World!".....	16
Изучаем объект "Текст".....	18
HTML-тэги в объекте "Текст".....	21
Отображение выражений с помощью объекта "Текст".....	21
Бэнды в FastReport.....	23
Бэнды-данные.....	25
Компонент TfrxDBDataSet.....	25
Отчет "Список клиентов".....	26
Отображение полей БД с помощью объекта "Текст".....	29
Псевдонимы.....	30
Переменные.....	31
Объект "Рисунок".....	33
Отчет с картинками.....	34
Отображение многострочного текста.....	36
Разрыв данных.....	38
Обтекание объектов текстом.....	40
Печать данных в виде таблицы.....	42
Печать этикеток.....	45
Child-бэнды.....	47
Смещение объектов.....	49
Отчет с двумя уровнями данных (master-detail).....	50
Связывание данных.....	52
Заголовок и подвал данных.....	54
Отчет с группами.....	55
Другие особенности групп.....	58
Нумерация записей.....	60
Агрегатные функции.....	61
Итоги по странице и по отчету.....	64
Вставка агрегатной функции.....	65

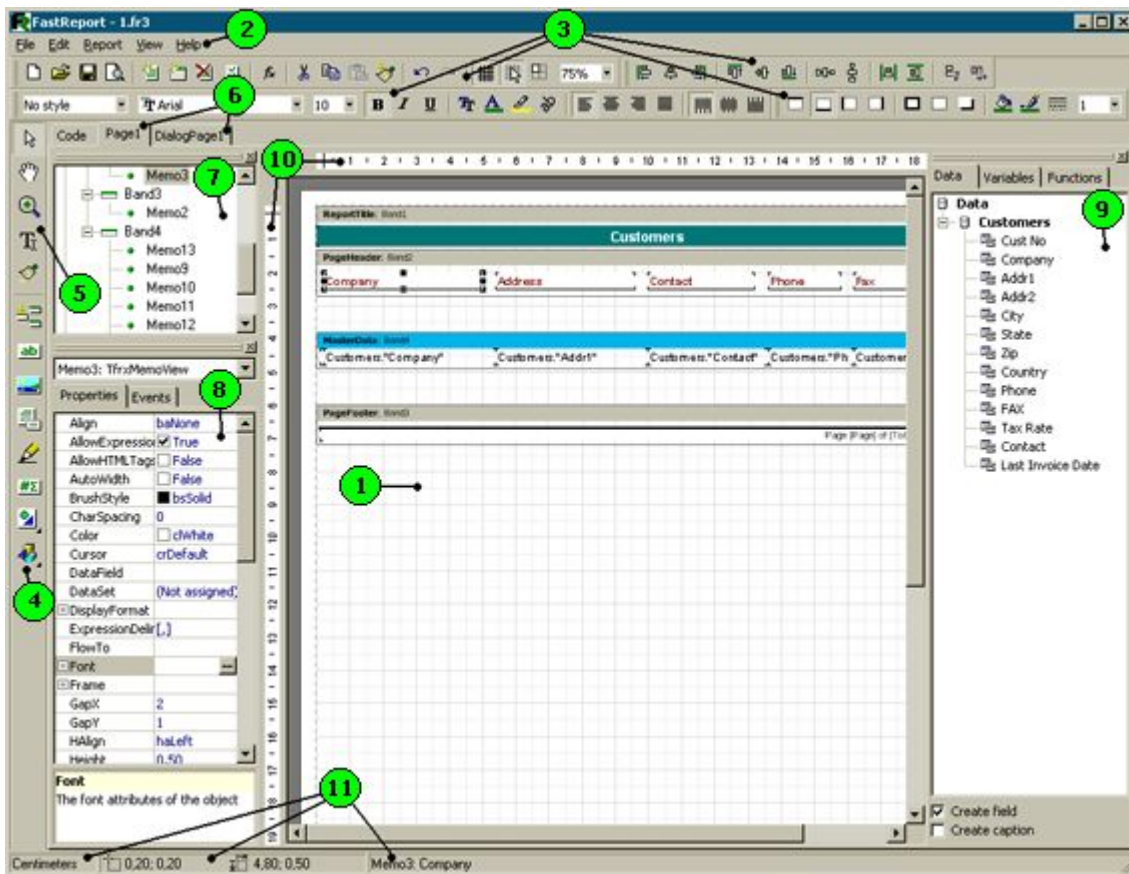
Особенности вызова агрегатной функции.....	66
Форматирование значений.....	67
Форматирование по месту.....	68
Условное выделение.....	70
Выделение строк через одну.....	71
Многостраничные отчеты.....	72
Вложенные отчеты.....	74
Вывод вложенных отчетов рядом.....	75
Ограничения на использование вложенных отчетов.....	75
Опция "Печатать на родителе" (PrintOnParent).....	76
Перекрестные (cross-tab) отчеты.....	78
Строим кросс-отчет.....	79
Использование функций.....	83
Сортировка значений.....	84
Таблица с составными заголовками.....	85
Подбор ширины ячеек.....	87
Выделение значений цветом.....	88
Управление кросс-таблицей из скрипта.....	89
Управление размером строк/колонок.....	93
Заполнение таблицы вручную.....	93
Построение диаграмм.....	96
Ограничение количества значений в диаграмме.....	99
Некоторые полезные настройки.....	100
Диаграмма с фиксированными данными.....	101
Скрипт.....	102
Первое знакомство.....	103
Структура скрипта.....	105
Скрипт "Hello, World!".....	106
Использование объектов в скрипте.....	107
Обращение к переменным из списка переменных отчета.....	108
Обращение к полям БД.....	108
Использование агрегатных функций в скрипте.....	108
Вывод значения переменной в отчете.....	109
События.....	109
Пример использования события OnBeforePrint.....	111
Печать итоговой суммы по группе в заголовке группы.....	113
Событие OnAfterData.....	116
Служебные объекты.....	117
Объект Report.....	117
Объект Engine.....	117
Объект Outline.....	118
Применение объекта Engine.....	119
Якоря.....	122
Применение объекта Outline.....	123
Событие страницы OnManualBuild.....	127
Создание объектов в скрипте.....	130

Диалоговые формы.....	132
Элементы управления.....	132
Отчет "Hello, World!".....	134
Ввод параметров и передача их в отчет.....	135
Взаимодействие элементов управления.....	136
Компоненты доступа к данным.....	137
Описание компонентов.....	137
TfrxDBLookupComboBox.....	138
TfrxBDETable.....	139
TfrxBDEQuery.....	140
TfrxBDEDataBase.....	141
Построение отчетов.....	142
Простой отчет типа "Список".....	143
Отчет с запросом параметров.....	144

Дизайнер

Компонент снабжен встроенным дизайнером, который можно вызвать в design-time двойным щелчком мыши на компоненте TfrxReport. Дизайнер предоставляет пользователю удобные средства для разработки внешнего вида отчета и позволяет сразу выполнить предварительный просмотр. Интерфейс дизайнера выполнен на современном уровне с использованием панелей инструментов (toolbars), расположение которых можно изменять по своему вкусу. Информация о расположении панелей запоминается в реестре, и при следующем запуске восстанавливается. Также в реестре запоминаются остальные настройки дизайнера.

Дизайнер доступен из среды Delphi в design-time. Для использования дизайнера в скомпилированном проекте необходимо использовать компонент TfrxDesigner из палитры компонентов FastReport, либо включить в список **uses** модуль frxDesign. Использование дизайнера в run-time дает возможность пользователю настраивать вид отчета, а также редактировать готовый отчет.



Цифрами на рисунке обозначены:
1 – рабочее поле дизайнера;

- 2 – строка меню;
- 3 – панели инструментов;
- 4 – панель объектов;
- 5 – панель режимов работы дизайнера;
- 6 – закладки страниц отчета;
- 7 – окно "Дерево отчета";
- 8 – окно "Инспектор объектов";
- 9 – окно "Дерево данных". Из этого окна можно перетаскивать элементы на лист отчета;
- 10 – линейки. При перетаскивании линейки на лист отчета образуется выносная линия, к которой могут прилипнуть объекты;
- 11 – строка состояния.

Клавиши управления

Клавиши	Описание
Ctrl+O	Команда меню "Файл Открыть..."
Ctrl+S	Команда меню "Файл Сохранить"
Ctrl+P	Команда меню "Файл Предварительный просмотр"
Ctrl+Z	Команда меню "Правка Отменить"
Ctrl+Y	Команда меню "Правка Повторить"
Ctrl+C	Команда меню "Правка Копировать"
Ctrl+V	Команда меню "Правка Вставить"
Ctrl+X	Команда меню "Правка Вырезать"
Ctrl+A	Команда меню "Правка Выделить все"
Стрелки	Перемещение по объектам
Del	Удаление выделенных объектов
Enter	Вызов редактора выделенного объекта
Shift+стрелки	Изменение размеров выделенных объектов
Ctrl+стрелки	Перемещение выделенных объектов
Alt+стрелки	Выделенный объект прилипает к ближайшему в выбранном направлении.

Управление мышью






Действие	Описание
Левая кнопка	Выбор объекта; вставка нового объекта; перемещение и изменение размеров объекта (объектов). Изменить масштаб выделенных объектов можно, потянув мышкой за красный квадратик на нижнем правом углу группы выделенных объектов.
Правая кнопка	Контекстное меню объекта (объектов), над которым находится указатель мыши.

Двойной щелчок левой кнопкой	Вызов редактора объекта. Если двойной щелчок сделать на пустом месте листа, то вызывается диалоговое окно "Параметры страницы".
Колесо мыши	Прокрутка листа отчета.
Shift + левая кнопка	Если объект уже выделен, то снимает с него выделение, иначе объект выделяется. Выделение остальных объектов не изменяется.
Ctrl + левая кнопка	При нажатии и перемещении мыши рисуется рамка; после отпускания кнопки мыши выделяются все объекты, попавшие в рамку. Такого же эффекта можно добиться, если щелкнуть левой кнопкой мыши на пустом месте листа, и, не отпуская кнопки, потянуть мышь до нужной позиции.
Alt + левая кнопка	Если выделен объект "Текст", редактирует его содержимое на месте.

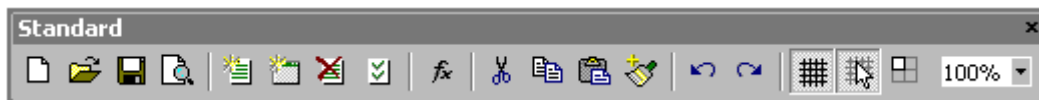
Панели инструментов

Панель режимов дизайнера



Панель объединена с панелью объектов и имеет следующие кнопки:

Иконка	Название	Описание
	Выбор объекта	Обычный режим работы, в котором указатель мыши позволяет выбирать объекты, изменять их размеры и пр.
	Рука	При нажатии левой кнопки мыши позволяет таскать лист отчета.
	Лупа	При однократном нажатии левой кнопки мыши увеличивает масштаб на 100%, правой кнопки – уменьшает масштаб на 100%. Если нажать левую кнопку и, не отпуская, тянуть мышь, то увеличивает выделенную область.
	Редактор текста	При щелчке на объекте "Текст" позволяет редактировать его содержимое прямо на листе отчета. Если нажать левую кнопку и, не отпуская, тянуть мышь, то на выделенном месте создается объект "Текст" и запустится его редактор.
	Копирование формата	Кнопка становится активной, если выбран объект "Текст". При нажатии левой кнопки мыши на объекте "Текст" копирует в объект форматирование, которое имеет ранее выделенный объект "Текст".

Панель инструментов "Стандартная"




Иконка	Название	Описание
	Новый отчет	Создает новый пустой отчет.
	Открыть отчет	Открывает существующий отчет из файла. Клавиатурный аналог - Ctrl+O.
	Сохранить отчет	Сохраняет отчет в файл. Клавиатурный аналог - Ctrl+S.
	Предварительный просмотр	Выполняет построение отчета и его предварительный просмотр. Клавиатурный аналог - Ctrl+P.
	Новая страница	Добавляет новую страницу в отчет.
	Новая диалоговая форма	Добавляет новую диалоговую форму в отчет.
	Удалить страницу	Удаляет текущую страницу.
	Свойства страницы	Вызывает диалог со свойствами страницы.
	Переменные	Вызывает редактор переменных отчета.
	Вырезать	Вырезает выделенные объекты в буфер обмена. Клавиатурный аналог - Ctrl+X.
	Копировать	Копирует выделенные объекты в буфер обмена. Клавиатурный аналог - Ctrl+C.
	Вставить	Вставляет объекты из буфера обмена. Клавиатурный аналог - Ctrl+V.
	Образец форматирования	Задаёт образец форматирования объекта "Текст". Выберите объект "Текст" и нажмите эту кнопку. Все последующие вставляемые объекты "Текст" будут иметь такое же форматирование, как и образец. Для сброса форматирования щелкните мышью на пустом месте страницы и нажмите эту кнопку.
	Отменить	Отменяет последнюю операцию. Клавиатурный аналог - Ctrl+Z.
	Повторить	Повторяет последнюю отмененную операцию. Клавиатурный аналог – Ctrl+Y.
	Показать сетку	Показывает сетку на странице. Шаг сетки можно задать в опциях дизайнера.
	Выравнивать объекты по сетке	При перемещении или изменении размеров объектов координаты/размеры будут изменяться скачкообразно в соответствии с шагом сетки.

	Расположить в узлах сетки	Изменяет размеры/расположение выделенных объектов таким образом, чтобы они размещались в узлах сетки.
	Масштаб	Задает масштаб страницы отчета.

Панель инструментов "Текст"



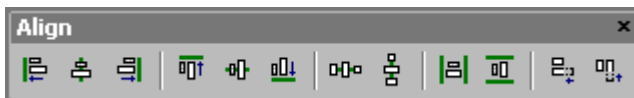
Иконка	Название	Описание
No style	Стиль	Позволяет выбрать стиль. Чтобы определить список стилей, вызовите пункт меню "Отчет Стили...".
Arial	Шрифт	Позволяет выбрать название шрифта из выпадающего списка. Помнит пять последних использованных шрифтов.
8	Размер шрифта	Позволяет выбрать размер шрифта из выпадающего списка. Размер можно также ввести вручную.
B	Утолщение	Устанавливает/снимает утолщение шрифта.
<i>I</i>	Наклон	Устанавливает/снимает наклон шрифта.
<u>U</u>	Подчеркивание	Устанавливает/снимает подчеркивание шрифта.
	Цвет шрифта	Выбирает цвет шрифта из выпадающего списка.
	Условное выделение	Показывает диалог с атрибутами выделения для выбранного объекта "Текст".
ab	Поворот текста	Позволяет выбрать поворот текста.
	Выравнивание влево	Устанавливает выравнивание текста влево.
	Выравнивание по центру	Устанавливает выравнивание текста по центру.
	Выравнивание вправо	Устанавливает выравнивание текста вправо.
	Выравнивание по ширине	Устанавливает выравнивание текста равномерно по ширине.
	Выравнивание по верхнему краю	Устанавливает выравнивание текста по верхнему краю.
	Выравнивание по высоте	Устанавливает выравнивание текста по высоте.
	Выравнивание по нижнему краю	Устанавливает выравнивание текста по нижнему краю.

Панель инструментов "Прямоугольник"



Иконка	Название	Описание
	Верхняя линия	Включает/выключает верхнюю линию рамки.
	Нижняя линия	Включает/выключает нижнюю линию рамки.
	Левая линия	Включает/выключает левую линию рамки.
	Правая линия	Включает/выключает правую линию рамки.
	Все линии	Включает все линии рамки.
	Нет линий	Выключает все линии рамки.
	Тень	Включает/выключает тень.
	Цвет фона	Выбирает цвет фона из выпадающего списка.
	Цвет линии	Выбирает цвет линии из выпадающего списка.
	Стиль линии	Выбирает стиль линии из выпадающего списка.
2	Толщина линии	Выбирает толщину линии из выпадающего списка.

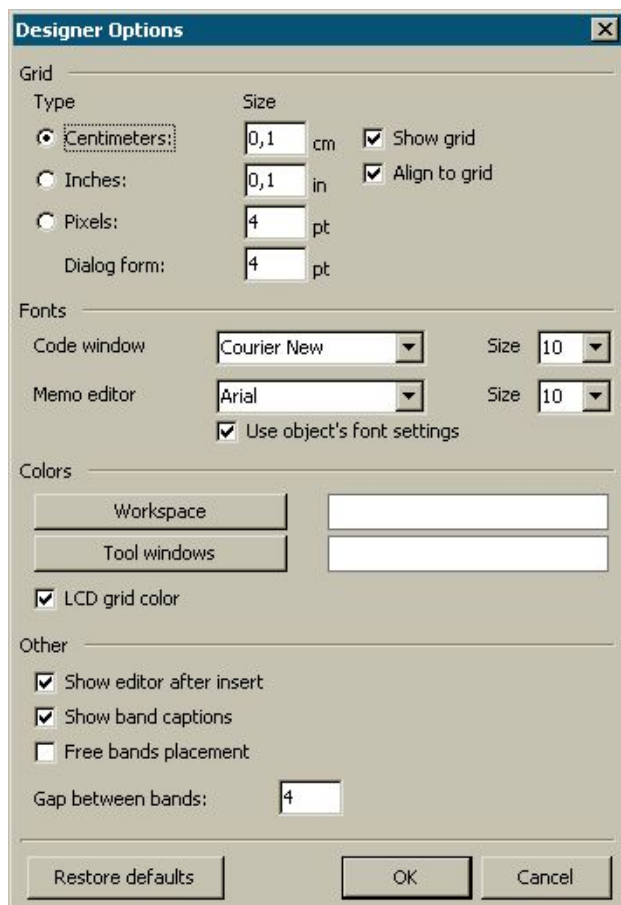
Панель инструментов "Выравнивание"



Иконка	Описание
	Выровнять левые края.
	Центрировать по горизонтали.
	Выровнять правые края
	Выровнять верхние края.
	Центрировать по вертикали.
	Выровнять нижние края.
	Расположить равномерно по ширине.
	Расположить равномерно по высоте.
	Центрировать по горизонтали в окне.
	Центрировать по вертикали в окне.
	Установить ту же ширину, что и у первого выделенного объекта.
	Установить ту же высоту, что и у первого выделенного объекта.

Опции дизайнера

Чтобы установить опции дизайнера, воспользуйтесь командой меню "Вид|Опции...".



Здесь можно задать используемые единицы измерения (сантиметры, дюймы, пиксели), указать шаг сетки для каждой единицы измерения. Переключить единицы измерения также можно в самом дизайнере, сделав двойной щелчок в левой части строки состояния, там, где отображаются текущие единицы измерения.

Также можно указать, надо ли показывать сетку и выравнивать объекты по сетке. Это также можно сделать с помощью кнопок на панели инструментов "Стандартная" в дизайнере.

Вы можете настроить шрифт для окна редактора кода и для редактора объекта "Текст". При включенной опции "Использовать установки шрифта объекта" шрифт в окне редактора текста будет соответствовать шрифту редактируемого объекта.

Если белый фон рабочего поля дизайнера и служебных окон вас не устраивает, можно сменить его, воспользовавшись кнопками "Цвет листа" и "Цвет служебных окон". Опция "Цвета LCD" слегка увеличивает контрастность линий сетки, что позволяет их лучше видеть на жидкокристаллических дисплеях.

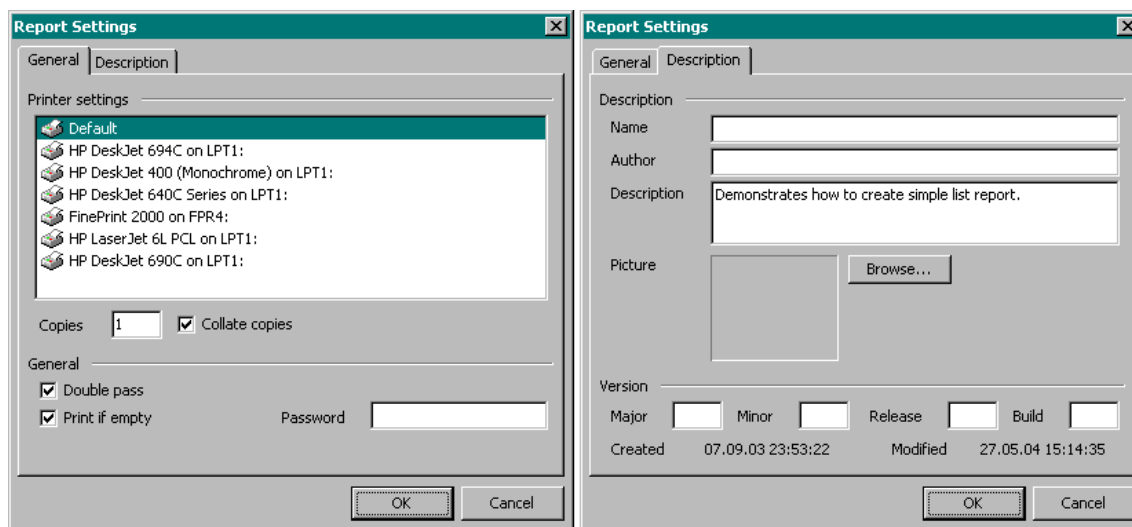
Опция "Редактирование после вставки" управляет процессом вставки новых объектов. Если опция включена, каждый раз при вставке объекта будет показываться его редактор. При вставке большого количества пустых объектов опцию лучше отключать.

Отключив опцию "Показывать заголовки бэндов", можно отключать заголовки у бэндов в целях экономии свободного места на странице. При этом название бэнда пишется внутри него.

Опция "Свободное расположение бэндов" отключает привязку бэндов к листу. По умолчанию эта опция отключена, и бэнды автоматически группируются на странице согласно их назначению. Между бэндами устанавливается промежуток, заданный в поле "Промежуток между бэндами".

Параметры отчета

Окно с параметрами отчета доступно из меню "Отчет|Опции...". Диалог имеет две страницы:



Вы можете привязать отчет к одному из принтеров, установленных в системе. Это значит, что печать отчета будет по умолчанию идти на выбранный принтер. Это полезно в случае, если в системе имеется несколько разных принтеров - текстовые документы можно привязать к монохромному принтеру, документы с графикой - к цветному. В списке принтеров имеется "Принтер по умолчанию". При его выборе отчет не будет привязан к какому-либо принтеру, и печать будет производиться на принтер, установленный по умолчанию.

Вы также можете указать, сколько копий отчета печатать и надо ли делать разбор по копиям. Заданные в этом диалоге значения будут показаны в окне "Опции печати".

Если флажок "Делать два прохода" установлен, формирование отчета будет осуществляться в два этапа. На первом проходе отчет формируется, осуществляется его разбивка на страницы, но результат нигде не сохраняется. На втором проходе происходит обычное формирование отчета с сохранением результата в потоке.

Для чего же нужно делать два прохода? Наиболее часто эта опция применяется в случае, если в отчете имеется упоминание об общем количестве страниц в нем, т.е. информация вида "Страница 1 из 15". Общее количество страниц подсчитывается на первом проходе и доступно через системную переменную TOTALPAGES. Наиболее частая ошибка - попытка применить эту переменную в однопроходном отчете, в этом случае она возвращает 0.

Другая область применения - выполнение каких-либо вычислений на первом проходе и отображение результатов на втором. Например, в случае, когда необходимо отобразить в заголовке группы сумму, которая обычно подсчитывается и отображается в подвале группы. Такого рода вычисления связаны с использованием встроенного языка FR.

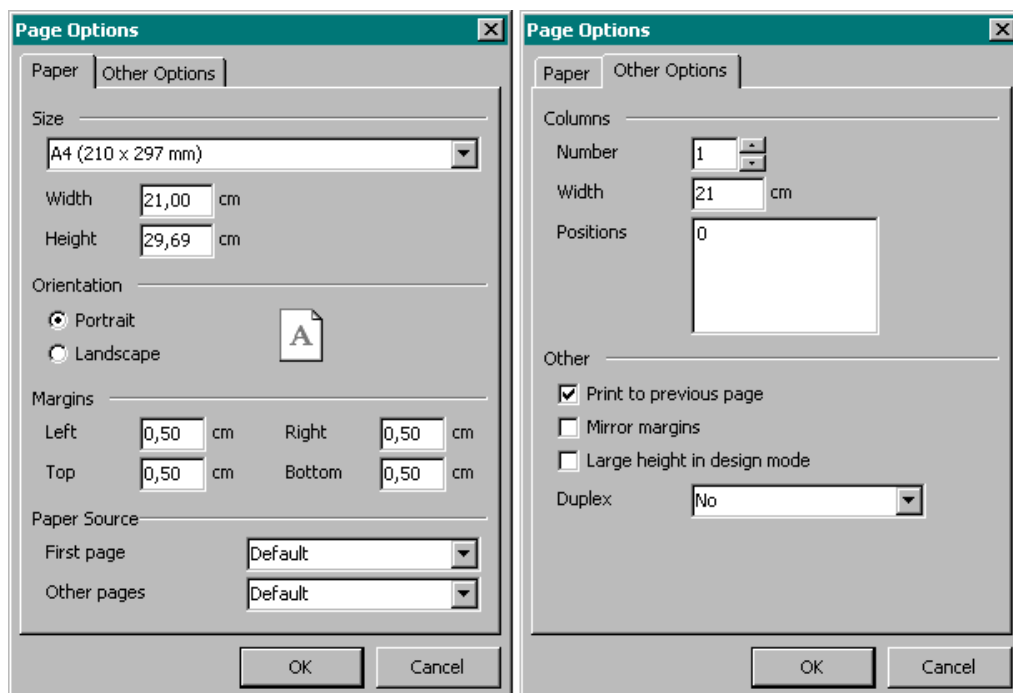
Флажок "Печатать, если отчет пуст" позволяет строить отчет, не содержащий ни одной строки данных. Если эту опцию отключить, пустые отчеты не будут строиться.

Поле "Пароль" позволяет задать пароль, который будет запрошен у пользователя при открытии отчета.

Элементы управления на второй странице диалога позволяют задать описание отчета.

Параметры страницы

Параметры страницы доступны через меню "Файл|Параметры страницы..." либо при двойном щелчке мышью на пустом месте страницы. Диалог имеет две страницы:



На первой странице диалога можно выбрать размер и ориентацию бумаги, а также задать поля. В выпадающих списках "Источник бумаги" можно выбрать лоток принтера для первой страницы и остальных страниц отчета.

На второй странице диалога можно указать количество колонок для печати многоколоночных отчетов. Текущие установки отображаются в дизайнера.

Флажок "Печатать на предыдущем листе" позволяет начать печать страницы, начиная со свободного места на предыдущем листе. Эта опция может применяться в случае, если шаблон отчета состоит из нескольких листов либо при печати пакетных (композитных) отчетов.

Опция "Зеркальные поля" меняет местами правое и левое поля страницы для четных страниц при просмотре или печати отчета.

Опция "Большая высота в дизайнера" увеличивает высоту страницы в несколько раз. Это может быть полезным, если на странице размещено много бэндов. Реальная высота страницы при построении отчета при этом не изменяется.

Построение отчетов

В этой главе мы рассмотрим возможности основных компонентов и объектов FastReport, составляющих основу любого отчета - таких, как TfrxReport, TfrxDBDataSet, TfrxMemoView, TfrxBand. Мы научимся строить простые отчеты, содержащие данные из таблицы БД.

Компонент TfrxReport

Основу любого отчета составляет компонент TfrxReport. Каждый такой компонент может содержать только один отчет. Компонент имеет все необходимое для загрузки отчета, его построения, предварительного просмотра и печати. Рассмотрим наиболее важные методы TfrxReport.

```
function LoadFromFile(const FileName: String; ExceptionIfNotFound: Boolean = False): Boolean;
```

загружает форму отчета из файла с указанным именем.

```
procedure SaveToFile(const FileName: String);
```

записывает текущую форму отчета в файл с указанным именем.

```
procedure DesignReport;
```

вызывает дизайнер отчета.

```
procedure ShowReport(ClearLastReport: Boolean = True);
```

строит отчет и показывает его в окне предварительного просмотра.

В этой главе нам понадобится только метод ShowReport. Единственный параметр этого метода определяет, надо ли очистить старый отчет перед построением нового. Этот параметр по умолчанию равен True, что нас вполне устраивает.

Объекты отчета

В FastReport пустой отчет представлен в виде листа бумаги. На любое место листа можно положить объекты, которые могут отображать разную информацию (текст, графика) и определять внешний вид отчета. Кратко опишем назначение объектов FastReport, входящих в стандартную поставку:














Объект	Иконка	Описание
Бэнд		Позволяет задать область отчета с определенным поведением.
Текст		Отображает одну или несколько строк текста внутри прямоугольной области.

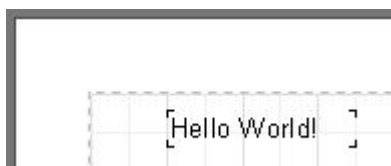
Рисунок		Отображает графический файл формата BMP, JPEG, ICO, WMF, EMF.
Линия		Отображает горизонтальную или вертикальную линию.
Служебный текст		Отображает служебную информацию (дата, время, номер страницы), а также агрегатные значения.
Вложенный отчет		Позволяет вставить дополнительный отчет внутрь основного.
Фигура		Объекты категории "Рисование" представляют собой различные геометрические фигуры – наклонная линия, прямоугольник, прямоугольник с круглыми углами, эллипс, треугольник, ромб.
Диаграмма		Отображает данные в виде диаграмм различных типов (круговая диаграмма, гистограмма и т.п.).
RichText		Отображает форматированный текст в формате Rich text (RTF).
CheckBox		Отображает значок - галочку или крестик.
Штрихкод		Отображает данные в виде штрихкода (доступно около десятка разных типов штрихкодов).
OLE		Может отображать любой объект, используя технологию OLE.
Градиент		Отображает градиентную заливку.

Основные объекты, с которыми вам придется работать больше всего - это объекты "Бэнд" и "Текст". Далее по ходу этой главы вы подробно ознакомитесь с их возможностями.

Отчет "Hello, World!"

Итак, создаем новый проект в Delphi, выбрав File|New application. На форму проекта кладем компонент TfrxReport из закладки FastReport. Это все, что необходимо для построения нашего первого отчета.

Отчет будет содержать всего одну надпись "Hello, World!". Открываем дизайнер, дважды щелкнув мышью на компоненте TfrxReport (также можно выбрать пункт "Design Report..." из контекстного меню компонента). На панели объектов дизайнера находим кнопку с объектом "Текст" и нажимаем ее. Перемещаем указатель мыши на нужное место на листе, и опять нажимаем клавишу мыши. Объект вставлен.



Сразу же на экране появляется окно редактора текста; если оно не появилось (это задается в настройках дизайнера), сделайте двойной щелчок мышью на объекте. Впишите текст "Hello, World!" и нажмите кнопку ОК.



Отчет готов. Для его просмотра выберите пункт Файл|Предварительный просмотр, или нажмите кнопку на панели инструментов. Вы увидите окно просмотра с единственной страницей отчета, которая содержит надпись "Hello, World!". Полученный отчет можно распечатать, сохранить в файл или экспортировать в один из поддерживаемых форматов.



Всю работу мы проделали в Delphi IDE, не написав ни строчки кода. Если же мы хотим запустить проект, надо кое-что добавить. Поместите на форму проекта кнопку и в ее обработчике OnClick напишите:

```
frxReport1.ShowReport;
```

Теперь при запуске нашего приложения и нажатии кнопки будет построен и показан отчет.

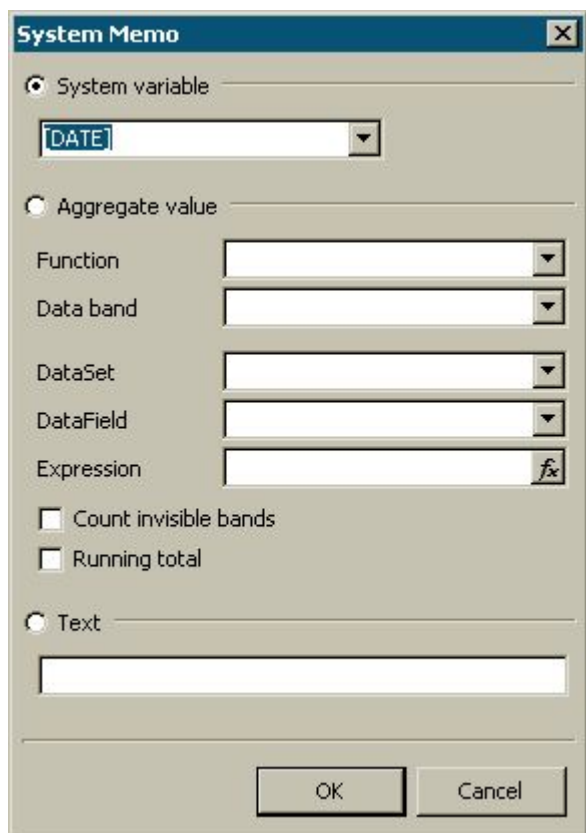
Немного усложним наш первый отчет. Пусть текст "Hello, World!" будет выведен жирными буквами на желтом фоне и взят в рамочку. Также рядом с текстом выведем текущую дату.

Снова открываем дизайнер FastReport и выделяем мышкой объект с текстом. Обратите внимание, что при этом некоторые кнопки на панелях инструментов становятся активными. Находим кнопку "Жирный" на панели инструментов "Текст" и нажимаем ее. Чтобы включить рамку, нажимаем кнопку "Все линии рамки". При желании можно отключить некоторые линии рамки,

воспользовавшись кнопками , задать цвет линии, ее толщину и стиль. На той же панели инструментов находим кнопку "Цвет заливки"  и выбираем желтый из выпадающего списка.

Вывести дату проще всего с помощью объекта "Служебный текст". Добавляем его на страницу точно так же, как мы сделали это с первым объектом. В окне редактора выбираем "Системная переменная" и в выпадающем списке ниже -

"[DATE]".



Закрываем редактор кнопкой OK, запускаем отчет и смотрим, что получилось.



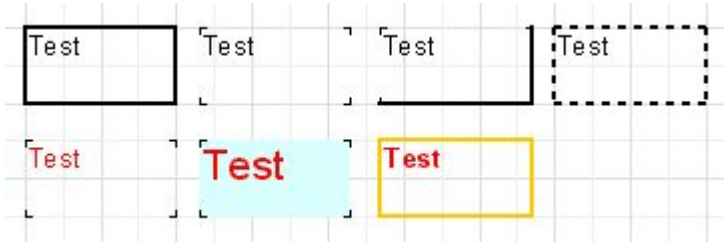
Изучаем объект "Текст"

Объект "Текст" обладает очень широкими возможностями. Мы уже видели, что он умеет отображать текст, рамку, заливку. Текст может быть отображен любым шрифтом, любого размера, цвета и стиля. Все настройки делаются визуально с помощью панелей инструментов:





Вот некоторые примеры оформления текста:




Познакомимся с другими возможностями этого основного объекта. Для испытаний создадим новый объект "Текст" и поместим в него 2 строки:

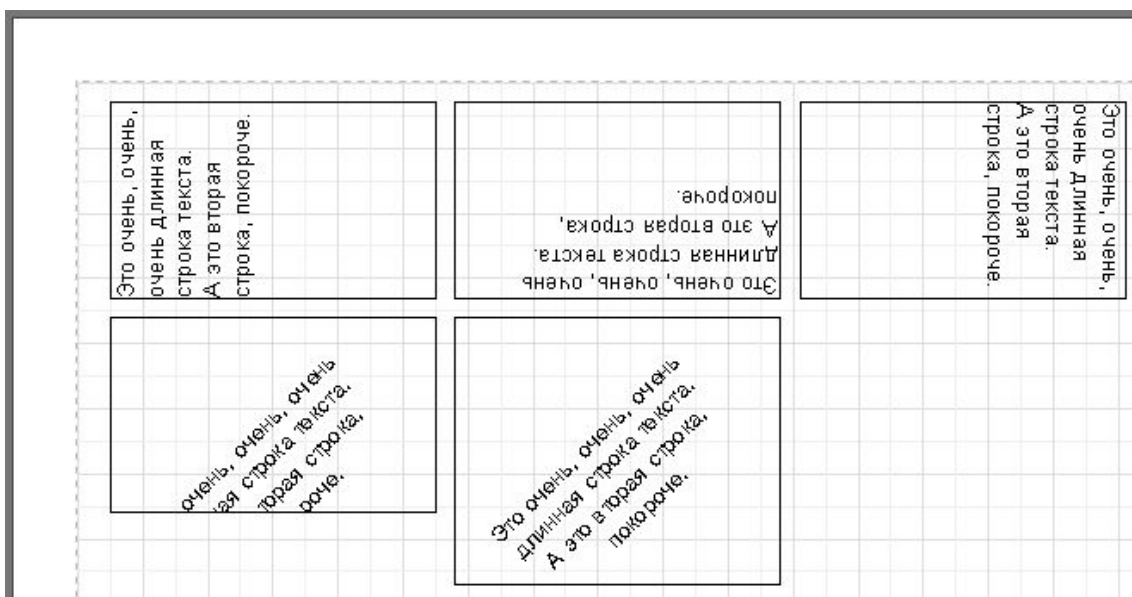
*Это очень, очень, очень длинная строка текста.
А это вторая строка, покороче.*

Включим рамку у объекта и с помощью мыши растянем его до размеров 9см x 3см. Мы видим, что объект умеет показывать не только однострочный текст, но и несколько строк. Теперь уменьшим ширину объекта до 5см. Видно, что длинные строки не уместились в объекте и были перенесены по словам. Это работает свойство объекта WordWrap, или "Перенос по словам". Если отключить его (в инспекторе или через контекстное меню объекта), то длинные строки просто будут обрезаны.

Теперь проверим, как работает выравнивание текста внутри объекта. Кнопки выравнивания расположены на панели инструментов "Текст" и позволяют независимо задать выравнивание текста по горизонтали и по вертикали. Обратите внимание на кнопку "Выравнивание по ширине" - она позволяет выровнять параграф по обоим краям объекта. При этом должна быть включена опция "Перенос слов".



Весь текст может быть повернут на любой угол в пределах 0..360 градусов. Кнопка  на панели инструментов "Текст" позволяет быстро повернуть текст на 45, 90, 180 и 270 градусов. Если нужно повернуть текст на какое-либо другое значение, воспользуйтесь инспектором объектов. Свойство Rotation задает нужный угол. При повороте на значения, отличные от 90, 180, 270, текст может вылезти за пределы объекта, как в нашем случае (см. рис.). Чтобы текст полностью уместился, немного увеличим высоту объекта.



Коротко остановимся на некоторых оставшихся свойствах объекта "Текст", которые влияют на его внешний вид. Большинство из этих свойств доступны только из инспектора объектов:

- BrushStyle - тип заливки объекта;
- CharSpacing - расстояние в пикселах между символами;

- GapX, GapY - отступы текста от левой и верхней границ объекта, в пикселах;
- LineSpacing - расстояние в пикселах между строками;
- ParagraphGap - отступ первой строки параграфа, в пикселах.

HTML-тэги в объекте "Текст"

Да, этот объект "понимает" некоторые простейшие тэги HTML. Тэги могут располагаться внутри текста объекта. По умолчанию тэги отключены; чтобы их включить, пометьте пункт "HTML тэги" в контекстном меню объекта или включите свойство "AllowHTMLTags" в инспекторе объектов. Вот список поддерживаемых тэгов:

 - жирный текст
 <i> - наклонный текст
 <u> - подчеркнутый текст
 <strike> - зачеркнутый текст
 <sub> - подстрочный текст
 <sup> - надстрочный текст
 - цвет шрифта

Небогато, но этого достаточно для большинства применений. Менять размер и имя шрифта нельзя - иначе пришлось бы значительно усложнять модуль рендеринга текста в FastReport.

Продemonстрируем применение тэгов на примерах.

текст жирный текст <i>наклонный текст</i> <i>жирный и
 наклонный</i>
 $E = mc^2$
 $A_{sub} 1 = B^{sup} 2$
 это обычный текст, а это красный
 это обычный текст, а это оранжевый



Отображение выражений с помощью объекта "Текст"

Одна из самых главных особенностей этого универсального объекта - это возможность отображения не только статического текста, но и выражений. Причем, выражения могут располагаться в объекте вперемешку с текстом. Рассмотрим

простой пример.

Недавно мы сделали отчет, который печатал строку "Hello, World!" и выводил текущую дату. Для этого нам пришлось поместить в отчет два объекта. В один мы поместили текст приветствия, в другой - системную переменную DATE. Однако, мы можем обойтись одним объектом "Текст" для вывода и строки, и даты. Для этого надо поместить в объект примерно такую строку:

Hello, World! Today is [DATE].

Если запустить отчет на построение, мы увидим приблизительно следующее:

Hello, World! Today is 01.01.2004.

Что произошло? В процессе построения отчета FastReport встретил в тексте выражение, заключенное в квадратные скобки, вычислил его и вставил полученное значение обратно в текст, убрав, разумеется, скобки. Объект "Текст" может содержать любое количество выражений, смешанных с обычным текстом. В скобки можно заключать и одиночные переменные, и выражения, например, $[1+2*(3+4)]$. В выражениях можно использовать константы, переменные, функции, поля БД. Мы рассмотрим эти возможности позже, по ходу главы.

Итак, FastReport автоматически распознает имеющиеся в тексте выражения по квадратным скобкам. А что, если наш текст содержит квадратные скобки, и мы не хотим, чтобы они были восприняты как выражения? Например, если нам нужно вывести такой текст:

$a[1] := 10$

FastReport воспримет [1] как выражение и выведет следующее:

$a1 := 10$

что нас, естественно, не устраивает. Один из способов избежать подобной ситуации - отключить распознавание выражений. Просто выключите свойство AllowExpressions ("Выражения" в контекстном меню), и все выражения, имеющиеся в тексте, будут проигнорированы. В нашем примере FastReport покажет то, что нам нужно:

$a[1] := 10$

Иногда требуется, чтобы текст содержал и выражения, и просто текст с квадратными скобками, например:

$a[1] := [myVar]$

Отключение выражений позволит вывести квадратные скобки там, где они нужны, но при этом отключит и обработку выражений. В этом случае FastReport позволяет задать другой набор символов, обозначающих выражение. За это отвечает свойство объекта `ExpressionDelimiters`, которое по умолчанию равно "[.]". В нашем случае можно использовать для выражений не квадратные, а угловые скобки:

```
a[1] := <myVar>
```

При этом свойству `ExpressionDelimiters` надо установить значение "<,>". Как видно, запятая разделяет открывающие символы и закрывающие. Есть одно ограничение - нельзя задавать одинаковые открывающие и закрывающие символы, т.е. "%,%%" работать не будет. Можно задать несколько символов, например "<%,%%>". При этом наш пример будет выглядеть так:

```
a[1] := <%myVar%>
```

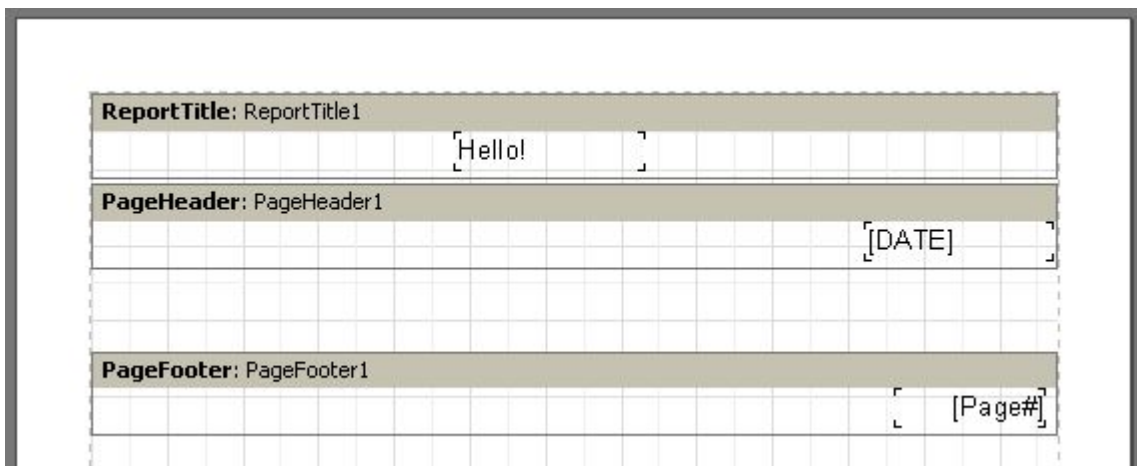
Бэнды в FastReport

Слово "бэнд" (band) по-английски означает "полоска". Бэнды применяются для логической группировки объектов. Так, разместив объект на бэнде типа "Заголовок страницы", мы тем самым говорим FastReport, что данный объект надо вывести на каждой странице готового отчета вверх. Аналогичным образом бэнд "Подвал страницы" выводится внизу каждой страницы, со всеми лежащими на нем объектами. Продемонстрируем это небольшим примером. Сделаем отчет, который содержит надпись "Hello!" вверху страницы, текущую дату вверху справа и номер страницы внизу справа.

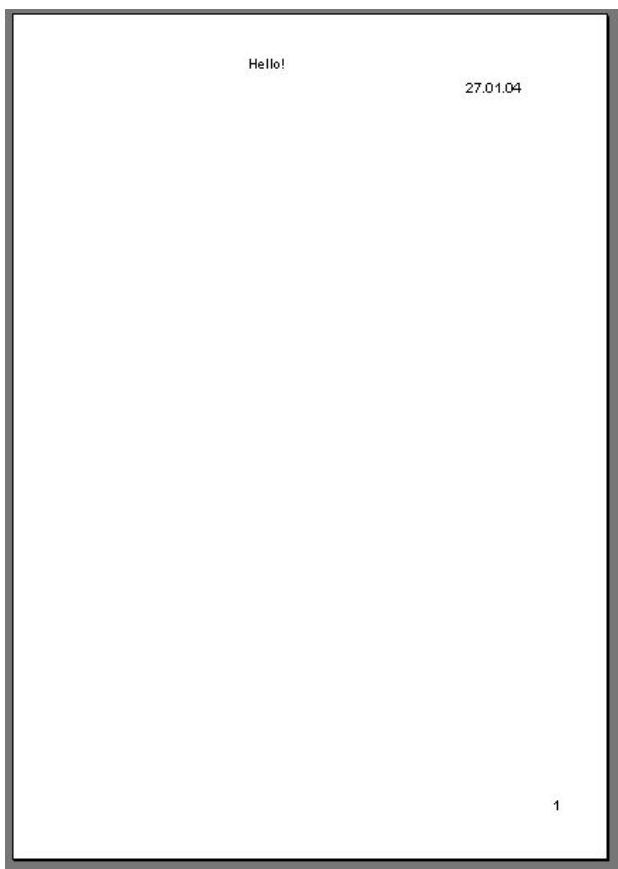
Зайдите в дизайнер FastReport и нажмите кнопку "Новый отчет" на панели инструментов. Вы увидите шаблон отчета, который уже содержит три бэнда: "Заголовок отчета", "Данные 1 уровня" и "Подвал страницы". Пока удалим бэнд "Данные 1 уровня" - щелкните мышкой на любом свободном месте внутри бэнда или на его заголовке и удалите его с помощью клавиши `Delete` или через контекстное меню. Теперь добавим новый бэнд - "Заголовок страницы". Для этого на панели объектов щелкните кнопку "Вставить бэнд"  и из открывшегося списка выберите "Заголовок страницы". Мы видим, что на страницу добавился новый бэнд. При этом имеющиеся бэнды сместились ниже. Дизайнер FastReport автоматически размещает бэнды на странице таким образом, чтобы вверху находились бэнды-заголовки, после них - бэнды-данные, и ниже всех - бэнды-подвалы.

Теперь размещаем объекты. На бэнд "Заголовок страницы" помещаем объект "Системный текст" и в его редакторе выбираем "Системная переменная", "[DATE]" (напомним, что дату можно вывести и с помощью обычного объекта "Текст", набрав в его редакторе текст "[DATE]"). На бэнд "Заголовок отчета"

помещаем объект "Текст", который будет содержать текст "Hello!". А на бэнде "Подвал страницы", как мы видим, уже размещен нужный нам объект, отображающий номер страницы.



Запустим отчет на выполнение и увидим, что объекты в готовом отчете разместились именно так, как нам нужно.



Итак, за размещение объектов в нужном месте отчета отвечают бэнды. В



зависимости от типа бэнда мы можем расположить объект сверху или внизу страницы, на первой странице, на последней странице. Основные бэнды, которые могут нам понадобиться в большинстве отчетов, работают следующим образом:

- бэнд "Заголовок страницы" выводится в самом верху на каждой странице;
- бэнд "Подвал страницы" выводится в самом низу на каждой странице;
- бэнд "Заголовок отчета" выводится на первой странице отчета сверху, но после бэнда "Заголовок страницы";
- бэнд "Подвал отчета" выводится в самом конце отчета, на свободном месте.

Бэнды-данные

Итак, мы плавно подошли к самому интересному - возможности выводить на печать данные из таблиц БД или запросов. Что такое таблица в данном случае? Это неопределенное количество строк (записей), каждая из которых содержит определенное количество колонок (полей). Для печати такого рода информации FastReport использует особый тип бэндов - бэнды-данные. Это бэнды с названиями "Данные xxx уровня". Чтобы напечатать всю таблицу или некоторые ее поля, необходимо добавить такой бэнд в отчет, подключить его к таблице и разместить на нем объекты с полями, которые мы хотим распечатать. При построении отчета FastReport повторит печать бэнда столько раз, сколько записей в нашей таблице. При этом, если закончилось свободное место на странице, будут сформированы новые страницы отчета.

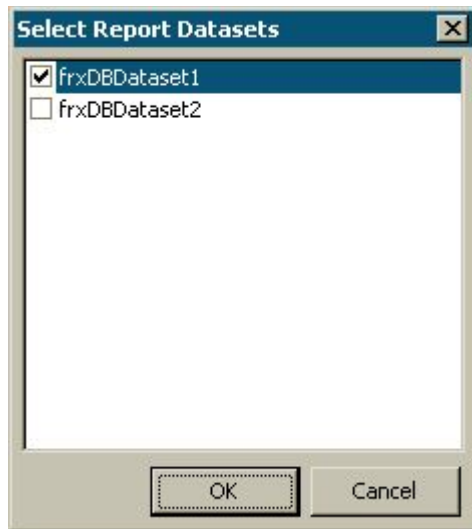
Компонент TfrxDBDataSet

Для подключения таблицы (или другого источника данных) к бэнду применяется компонент-коннектор TfrxDBDataSet  из палитры компонент FastReport. Этот компонент выполняет роль посредника между источником данных и ядром FastReport. Компонент отвечает за навигацию по записям и обращение к полям. Это позволило не привязывать ядро FastReport к какой-либо библиотеке доступа к данным. FastReport может одновременно работать как с BDE, IB_Objects (с их нестандартной реализацией, несовместимой с TDataSet), так и с любой другой библиотекой, либо вообще получать данные из источника, не связанного с БД, например, из массива или файла. Компонент TfrxDBDataSet предназначен для работы с источниками данных, совместимыми с TDataSet (это BDE, ADO, IBX и подавляющее большинство других библиотек). Для работы с IB_Objects предназначен компонент TfrxIBODataSet, для работы с прочими источниками данных (массив, файл и т.п.) - компонент TfrxUserDataSet .

Пользоваться компонентом TfrxDBDataSet очень просто. Чтобы связать его с источником данных, настройте свойство DataSet (подключается непосредственно к таблице или запросу) или DataSource (подключается к компоненту TDataSource).

Оба способа подключения равноценны, просто первый позволяет обойтись без компонента TDataSource.

Чтобы компонент и связанные с ним данные стали доступны в отчете, надо явно указать, какие источники данных используются в отчете. Для этого в дизайнера FastReport выберите пункт меню "Отчет|Данные..." и в открывшемся окне пометьте галочками нужные источники.



Отчет "Список клиентов"

Наш второй отчет будет значительно сложнее первого - он будет содержать данные из таблицы БД - список клиентов некоторой фирмы. Для этого воспользуемся демонстрационной базой данных, которая идет в комплекте с Delphi – DBDEMOS. Создадим новый проект в Delphi. На форму положим компонент TTable и настроим его свойства:

```
DatabaseName = 'DBDEMOS'  
TableName = 'Customer.db'
```

Для того, чтобы работать с таблицей из FastReport, добавим компонент TfrxDBDataSet и настроим его свойство:

```
DataSet = Table1
```

Наконец, положим на форму основной компонент FastReport - TfrxReport. Зайдем в дизайнер и нажмем кнопку "Новый отчет", чтобы FastReport автоматически создал пустой шаблон с тремя бэндами - "Заголовок отчета", "Данные 1 уровня" и "Подвал страницы". Чтобы наша таблица стала видна в FastReport, необходимо разрешить ее использование. Для этого выберем пункт меню "Отчет|Данные..." и пометим нашу таблицу (она сейчас единственная в

списке) галочкой. После того, как мы закрыли диалоговое окно, таблица и ее поля стали видны в служебном окне "Данные".

Приступим к созданию формы отчета. На бэнд "Заголовок отчета" положим объект "Текст" с текстом "Список клиентов". Бэнд "Данные 1 уровня" подключим к нашему источнику данных. Это можно сделать тремя способами:

- сделать двойной щелчок на бэнде;
- выбрать пункт "Редактировать..." из контекстного меню бэнда;
- щелкнуть на свойстве DataSet в инспекторе объектов.

Теперь разместим на бэнде четыре объекта, которые будут отображать номер клиента, его наименование, телефон и факс. Сделаем это разными способами, чтобы продемонстрировать широкие возможности дизайнера FastReport. Первый объект "Текст" положим на бэнд и наберем в нем текст "[frxDBDataSet1."CustNo"]". Это самый неудобный способ, т.к. приходится ссылку на поле писать вручную, и мы можем легко ошибиться. Чтобы облегчить вставку таких ссылок в текст, можно воспользоваться конструктором выражений - кнопка его вызова расположена на панели инструментов в редакторе объекта "Текст". Для вставки нашего поля нажмем эту кнопку и дважды щелкнем на нужном элементе в открывшемся диалоге. Нажав кнопку ОК, закрываем диалог и видим, что поле вставлено в текст.

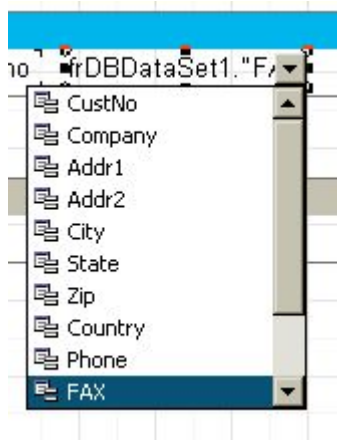
Второй способ вставки поля БД в отчет похож на тот, что широко применяется в среде Delphi - мы сделаем это с помощью настройки свойств в инспекторе объектов. Положим на бэнд второй объект, в редакторе ничего писать не будем. В инспекторе настроим свойства объекта:

```
DataSet = frxDBDataSet1  
DataField = 'Company'
```

Т.к. оба свойства представляют собой список, нам достаточно выбрать нужные значения мышкой.

Третий способ - drag&drop нужного поля из служебного окна "Данные" в отчет. Это самый простой и наглядный способ. Схватите мышкой поле "Phone" и перетащите его на бэнд. Единственное, что надо сделать в нашем случае - это отключить флажок "Создать заголовок" в нижней части окна "Данные", иначе вместе с нужным полем мы создадим лишний в данном случае объект, содержащий название поля.

Наконец, четвертый способ. Поместите пустой объект "Текст" на бэнд. Теперь подведите указатель мыши к объекту. В правой части объекта вы увидите изображение кнопки со стрелкой вниз, как на раскрывающихся списках. Это и есть раскрывающийся список полей БД. Нажмите на кнопку и выберите из списка поле "FAX". Вы можете пользоваться этой возможностью, когда бэнд подключен к данным.



Итак, наш отчет готов:

ReportTitle: ReportTitle1			
[Список клиентов]			
MasterData: MasterData1			
[frDBDataSet1]	[frDBDataSet1."Company"]	[frDBDataSet1."Pho"]	[frDBDataSet1."FAX"]
PageFooter: PageFooter1			
[Page#]			

Нажмем кнопку предварительного просмотра и посмотрим, что получилось.

Список клиентов

1221	Kauai Dive Shoppe	808-555-0269	808-555-0278
1231	Unisco	809-555-3915	809-555-4958
1351	Sight Diver	357-6-876708	357-6-870943
1354	Cayman Divers World Unlimited	011-5-697044	011-5-697064
1356	Tom Sawyer Diving Centre	504-798-3022	504-798-7772
1380	Blue Jack Aqua Center	401-609-7623	401-609-9403
1384	VIP Divers Club	809-453-5976	809-453-5932
1510	Ocean Paradise	808-555-8231	808-555-8450
1513	Fantastique Aquatica	057-1-773434	057-1-773421
1551	Marmot Divers Club	416-698-0399	426-698-0399
1560	The Depth Charge	800-555-3798	800-555-0353
1563	Blue Sports	610-772-6704	610-772-6898
1624	Makai SCUBA Club	317-649-9098	317-649-6787
1645	Action Club	813-870-0239	813-870-0282
1651	Jamaica SCUBA Centre	011-3-697043	011-3-697043
1680	Island Finders	713-423-5675	713-423-5676

Отображение полей БД с помощью объекта "Текст"

Как мы видели, объект "Текст" способен, помимо статического текста и выражений, также отображать данные из БД. Причем мы можем делать это двумя способами: поместить ссылку на поле БД в текст объекта либо подключить объект к нужному полю с помощью свойств DataSet, DataField. Первый способ хорош тем, что позволяет нам в одном объекте вывести и содержимое поля, и какой-нибудь поясняющий текст. Например, так:

Контактное лицо: [frxDBDataSet1."Contact_Person"]

Как видно, для ссылок на поле БД применяется специальный синтаксис: имя_набора_данных."имя_поля". Как имя набора, так и имя поля может содержать пробелы. Не допускается наличие пробела между точкой и кавычкой.

В текст объекта можно помещать не только ссылку на поле. Мы можем произвести какие-нибудь вычисления с полем:

*Длина в см: [<frxDBDataSet1."Length_in"> * 2.54]*

Обратите внимание на использование квадратных и угловых скобок. Напомним, что квадратные скобки по умолчанию используются для обозначения выражений, имеющих в тексте объекта. Вместо квадратных скобок может быть использована пара любых других открывающих/закрывающих последовательностей, если это требуется (см. "Отображение выражений с

помощью объекта "Текст"). Угловые же скобки используются внутри выражения для обозначения переменных FastReport и полей БД. По логике, мы должны были бы писать

Контактное лицо: [<frxDBDataSet1."Contact_Person">]

вместо

Контактное лицо: [frxDBDataSet1."Contact_Person"]

но обе формы записи верны, т.к. FastReport допускает отсутствие угловых скобок в случае, если выражение содержит только одну переменную/поле БД. Однако, если в выражении несколько членов, то скобки обязательны:

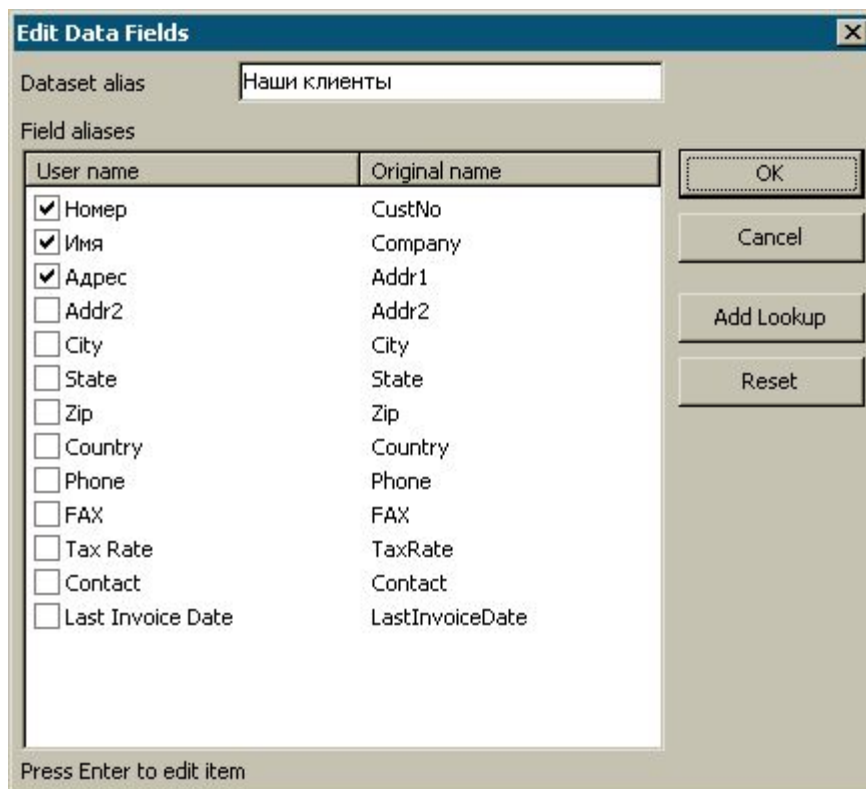
*Длина в см: [<frxDBDataSet1."Length_in"> * 2.54]*

Это одно из существенных отличий от ранних версий FastReport, где вместо угловых скобок везде используются квадратные. Это сделано по той причине, что все выражения обрабатываются скриптовым языком, где квадратные скобки используются для обозначения множеств или открытых массивов.

Псевдонимы

В предыдущем отчете мы использовали источник данных с именем frxDBDataSet1 и полями CustNo, Company, Phone, FAX. Соответственно, в отчет приходилось вставлять нечто вроде "[frxDBDataSet1."CustNo"]". Понятно? Не очень. Так и хочется переименовать источник данных в "Наши клиенты", а поле - в "Номер". Однако, "frxDBDataSet1" - это имя компонента, которое не может содержать русские буквы. А "CustNo" - это имя поля, его тоже напрямую не переименуешь, без реструктуризации базы данных. Выход есть, он заключается в использовании так называемых псевдонимов, или алиасов. И у источника данных, и у поля есть вторые имена - псевдонимы, которые можно легко изменить (оригинальные имена при этом, разумеется, не меняются). Если у имени есть алиас, то именно он используется в FastReport. В противном случае используется оригинальное имя.

Переименовать источник данных и его поля в FastReport очень просто. Это делается из среды Delphi. Сделайте двойной щелчок на компоненте TfrxDBDataSet, и вы увидите редактор алиасов. Здесь можно изменить имя источника данных, имена его полей и выбрать только те поля, которые нам необходимы в отчете. Переименуем источник и поля (см. рис):



Заметим, что алиас самого источника можно поменять и без использования редактора алиасов - для этого измените свойство `UserName` компонента `TfrxDBDataSet`.

Теперь нам необходимо исправить и сам отчет, т.к. имена полей изменились. Чтобы поменять названия полей в объектах, проще всего воспользоваться четвертым способом, рассмотренным в главе "Отчет "Список клиентов": наведите мышку на объект "Текст", чтобы в правой части объекта появилась кнопка, нажмите на кнопку и выберите необходимое поле из списка. Как видим, теперь названия источника данных и его полей более чем понятны.

Остается добавить, что работу по назначению алиасов лучше сделать в самом начале, до построения отчета. Это позволит избежать последующего переименования полей в отчете.

Переменные

Кроме использования псевдонимов, есть еще один способ, позволяющий задать более понятные имена полям БД, и не только им. Используя переменные, определенные в отчете, можно сопоставить переменной имя поля БД, а также любое выражение. Для работы с переменными в FastReport выберите пункт меню "Отчет|Переменные..." или нажмите кнопку "Переменные" на панели инструментов.

Список переменных в FastReport имеет двухуровневую структуру. Первый уровень - это категории, второй - сами переменные. Разбивка переменных на категории сделана для удобства пользования в случае, если список переменных велик. В списке должна существовать как минимум одна категория, т.е. переменные не могут располагаться на верхнем уровне. Кроме того, категории нужны только для логической группировки переменных и в отчет не вставляются. Поэтому, давая имя переменной, не забывайте, что оно должно быть уникально - две одинаковые переменные в разных категориях создать нельзя.

Продemonстрируем использование переменных на примере. Допустим, у нас есть два источника данных: первый - frxDBDataSet1 с полями "CustNo" и "Name" и второй - frxDBDataSet2 с полями "OrderNo" и "Date". Мы можем сопоставить полям такой список переменных:

Клиенты

Номер клиента

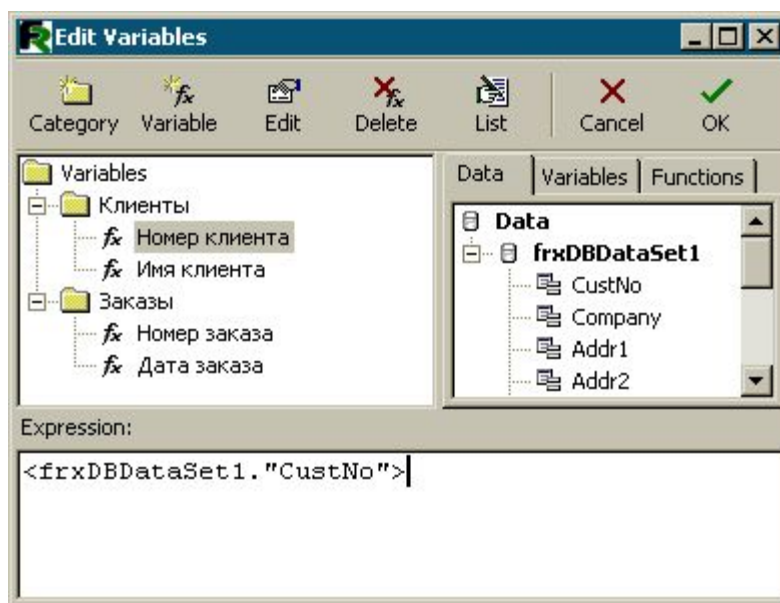
Имя клиента

Заказы

Номер заказа

Дата заказа

где "Клиенты" и "Заказы" - это две категории. Зайдем в редактор переменных и, пользуясь кнопками "Новая категория", "Новая переменная" и "Редактировать", создадим необходимую структуру. Чтобы сопоставить переменные полям БД, выберем переменную и дважды щелкнем на нужном поле БД в правой части окна. При этом ссылка на поле БД поместится в нижнюю часть окна. Выражение в нижней части окна - это и есть значение переменной, при необходимости его можно отредактировать вручную. Категории сопоставлять ничему не надо.



После того, как список переменных создан, закроем редактор переменных. Теперь надо вставить переменные в отчет. В отличие от вставки полей БД, вариантов здесь гораздо меньше. Мы можем либо вставить переменную в текст объекта вручную, набрав текст "[Номер клиента]", либо перетащить переменную из служебного окна "Данные" в нужное место отчета. Во втором случае надо переключиться на закладку "Переменные" в этом окне.

Объект "Рисунок"

Следующий объект, который мы рассмотрим - это объект "Рисунок". Он также довольно часто используется в отчетах. С помощью объекта вы можете вставить в отчет логотип вашей фирмы, фотографию сотрудника или любую другую графическую информацию. Объект способен отображать графику в формате BMP, JPEG, ICO, WMF, EMF.

Давайте рассмотрим возможности объекта. Создайте пустой отчет и поместите на лист отчета объект "Рисунок". В редакторе объекта (если он не открылся автоматически, сделайте двойной щелчок мышью на объекте) мы можем загрузить рисунок из файла или очистить имеющийся в объекте рисунок. Загрузите любой подходящий рисунок и нажмите кнопку ОК.



В контекстном меню объекта мы увидим следующие опции (в скобках - соответствующие названия свойств в инспекторе объектов):

- Авторазмер (AutoSize)

- Растягивание (Stretch) - включено по умолчанию
- Центровка (Center)
- Сохранять пропорции (KeepAspectRatio) - включено по умолчанию

Включив опцию "Авторазмер" мы увидим, что объект принял размеры, соответствующие находящемуся в нем рисунку. Иногда такая возможность бывает полезна, если надо отображать рисунки разных размеров. По умолчанию эта опция выключена, что подходит для большинства случаев.

Опция "Растягивание" включена по умолчанию, что заставляет рисунок растягиваться внутри объекта. Изменяйте размеры объекта мышкой, и вы увидите, что размер картинки все время соответствует размеру объекта. Если опцию отключить, то рисунок будет отображаться в исходных размерах. Это поведение отличается от опции "Авторазмер" тем, что размеры объекта не подгоняются под размер рисунка, т.е. объект можно сделать больше рисунка или меньше.

Опция "Центровка" позволяет отцентрировать рисунок внутри объекта.

Опция "Сохранять пропорции" включена по умолчанию и выполняет очень полезную задачу: не позволяет пропорциям рисунка искажаться при изменении размеров объекта. Эта опция работает только в паре с опцией "Растягивание". При любом изменении размеров объекта нарисованный круг останется кругом, а не превратится в овал. При этом растянутый рисунок занимает не весь внутренний объем объекта, а только часть, необходимую для отображения картинки в правильных пропорциях. Если опцию отключить, то картинка растянется на весь объем объекта, и, если размеры объекта не соответствуют исходным пропорциям картинки, картинка исказится.

Отчет с картинками

Объект "Рисунок", как и многие объекты в FastReport, умеет отображать данные из БД. Подключение объекта к нужному полю БД осуществляется с помощью свойств DataSet, DataField в инспекторе объектов. В отличие от объекта "Текст", это единственный способ подключить объект к данным.

Продemonстрируем все вышесказанное примером отчета, который будет содержать изображения рыб вместе с их названиями. Для этого нам опять потребуется демонстрационная база данных DBDEMOS, идущая в комплекте с Delphi.

Создадим пустой проект в Delphi. Положим на форму компонент TTable и настроим его свойства:

```
DatabaseName = 'DBDEMOS'  
TableName = 'Biolife.db'
```

Для того, чтобы работать с таблицей из FastReport, добавим компонент TfrxDBDataSet и настроим его свойства:

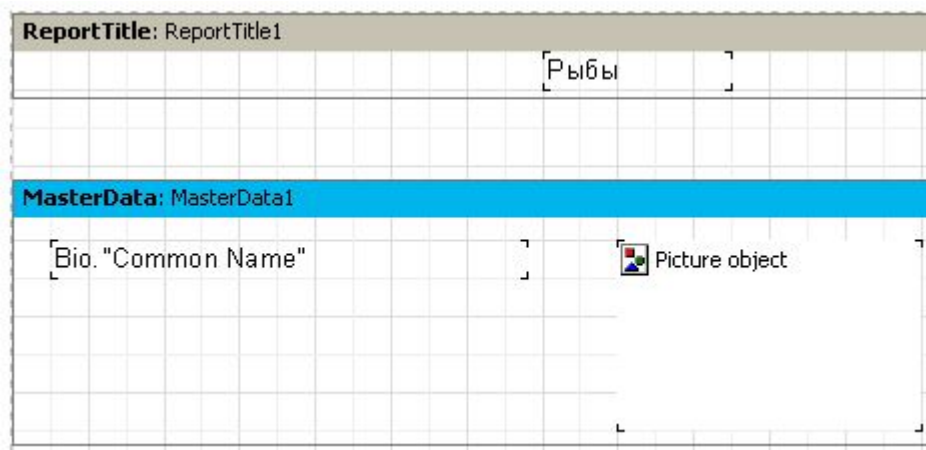
```
DataSet = Table1  
UserName = 'Bio'
```

Наконец, положим на форму компонент TfrxReport. Зайдем в дизайнер и нажмем кнопку "Новый отчет", чтобы FastReport автоматически создал пустой шаблон. Подключим таблицу к отчету в окне "Отчет|Данные...".



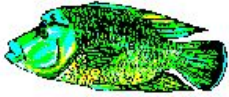
Приступим к созданию формы отчета. На бэнд "Заголовок отчета" положим объект "Текст" с текстом "Рыбы". Бэнд "Данные 1 уровня" подключим к источнику данных (сделаем двойной щелчок на бэнде и выберем "Bio" из списка). Высоту бэнда увеличим до 3см, чтобы уместить картинку. На бэнд положим объект "Текст" и подключим его к полю "CommonName" одним из способов, описанных выше. Рядом положим объект "Рисунок" и подключим его к полю "Graphic". Для этого в инспекторе объектов настроим свойства:

```
DataSet = Bio  
DataField = 'Graphic'
```

напомним, что оба этих свойства - типа "список", поэтому нужные значения можно выбрать с помощью мыши. Чтобы уместить картинку, растянем объект до размеров 4 x 2.5см.






Все, отчет готов (см. рис):

Рыбы	
Clown Triggerfish	
Red Emperor	
Giant Maori Wrasse	

Отображение многострочного текста

Вернемся к предыдущему примеру с рыбами. В таблице Biolife.db есть поле "Notes", которое содержит подробное описание каждой рыбы. Давайте модернизируем наш отчет, добавив в него это поле.

На первый взгляд все просто: добавляем на бэнд с данными объект "Текст", подключаем его к полю "Notes" и устанавливаем размеры объекта - 8 x 2.5см. Запускаем отчет на выполнение и видим, что получилось не совсем то, чего мы ожидали:

Рыбы		
Clown Triggerfish	Also known as the big spotted triggerfish. Inhabits outer reef areas and feeds upon crustaceans and mollusks by crushing them with powerful teeth. They are voracious eaters, and divers report seeing the clown triggerfish devour beds of pearl oysters.	
Red Emperor	Called seaperch in Australia. Inhabits the areas around lagoon coral reefs and sandy bottoms.	
Giant Maori Wrasse	The red emperor is a valuable food fish and considered a great sporting fish that fights with fury when hooked. The flesh of an old fish is just This is the largest of all the wrasse. It is found in dense reef areas, feeding on a wide variety of mollusks, fishes, sea urchins, crustaceans, and other invertebrates. In spite of its immense size, divers find it a very wary fish.	

Однако, FastReport всего лишь сделал то, что его просили сделать. Поле "Notes" содержит многострочный текст, размер которого может варьироваться. А

наш объект "Текст", отображающий информацию из этого поля, имеет фиксированный размер. Вот некоторые строки и не влезли в объект и были обрезаны. Как поступить в данной ситуации?



Можно, конечно, подобрать размеры объекта с запасом или уменьшить размер шрифта. Однако, это приведет к неэкономному использованию места на листе: одни рыбы имеют длинное описание, другие - короткое. В FastReport есть средства, позволяющие решить эту проблему.

Речь идет о возможности бэнда подбирать свою высоту таким образом, чтобы уместить все имеющиеся в нем объекты. Для этого надо всего лишь включить свойство "Растягивание" (Stretch). Однако это не все - объект с длинным текстом и сам должен уметь растягиваться. Объект "Текст" умеет это делать.

Объект может автоматически подбирать свою высоту или ширину, чтобы полностью уместить имеющийся в нем текст. Для этого служат свойства "Автоширина" (AutoWidth) и "Растягивание" (StretchMode). Свойство "Автоширина" подбирает ширину объекта таким образом, чтобы уместились все строки, без переносов слов. Этот режим удобен, когда объект содержит единственную строку текста. Свойство "Растягивание" позволяет подобрать высоту объекта так, чтобы поместился весь текст. Ширина объекта при этом не меняется. Это свойство является перечислением, и вы можете выбрать один из режимов в инспекторе объектов:

smDontStretch - не растягивать объект, значение по умолчанию;
smActualHeight - растянуть объект, чтобы уместился весь текст;
smMaxHeight - растянуть объект, чтобы его нижняя граница совпала с нижней границей бэнда, на котором находится объект. Этот режим мы рассмотрим чуть позже.

Сейчас нас интересует свойство "Растягивание" объекта "Текст". Включите его в контекстном меню объекта, либо установите значение свойства StretchMode = smActualHeight. Также включите свойство "Растягивание" у бэнда. Запустим отчет и убедимся, что теперь все работает как надо.

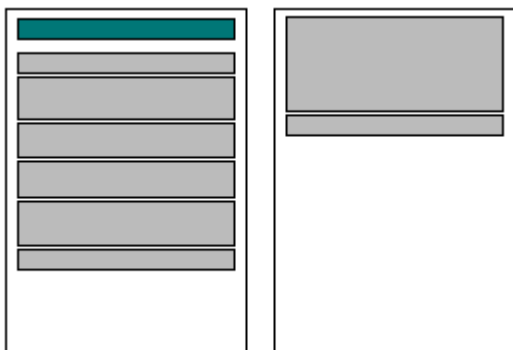
Рыбы		
Clown Triggerfish	<p>Also known as the big spotted triggerfish. Inhabits outer reef areas and feeds upon crustaceans and mollusks by crushing them with powerful teeth. They are voracious eaters, and divers report seeing the clown triggerfish devour beds of pearl oysters.</p> <p>Do not eat this fish. According to an 1878 account, "the poisonous flesh acts primarily upon the nervous tissue of the stomach, occasioning violent spasms of that organ, and shortly afterwards all the muscles of the body. The frame becomes rocked with spasms, the tongue thickened, the eye fixed, the breathing laborious, and the patient expires in a paroxysm of extreme suffering."</p> <p>Not edible.</p>	
Red Emperor	<p>Range is Indo-Pacific and East Africa to Samoa. Called seaperch in Australia. Inhabits the areas around lagoon coral reefs and sandy bottoms.</p> <p>The red emperor is a valuable food fish and considered a great sporting fish that fights with fury when hooked. The flesh of an old fish is just as tender to eat as that of the very young.</p>	

Как видим, при построении отчета FastReport заполняет объекты данными, растягивает объекты со включенной опцией "Растягивание" и потом подбирает высоту бэнда таким образом, чтобы уместить все объекты. Если опция "Растягивание" у бэнда отключена, то подбор высоты бэнда не производится, и бэнд выводится с той высотой, что была установлена в дизайнере. Если мы попробуем отключить эту опцию, мы увидим, что объекты с длинным текстом по-прежнему растягиваются, а бэнды - нет, что приводит к наложению текста. Ведь очередной бэнд выводится сразу после предыдущего.

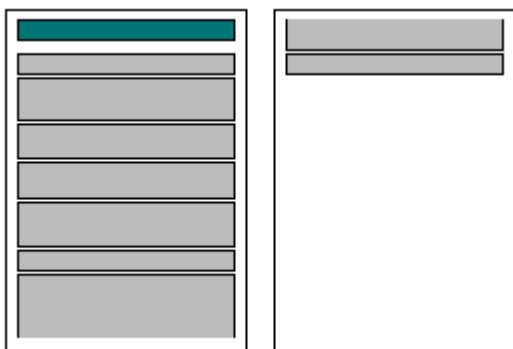
Разрыв данных

Обратим внимание на одну особенность отчета с рыбами: на некоторых страницах внизу остается много пустого места. Почему это происходит? Когда отчет строится, ядро FastReport заполняет свободное место листа бэндами. После вывода каждого бэнда текущая позиция смещается все ниже и ниже. Когда FastReport обнаруживает, что места для вывода очередного бэнда не хватает (его высота больше, чем высота оставшегося места на листе), то формируется новая страница и вывод бэндов продолжается на ней. И так до тех пор, пока есть записи в наборе данных.

Наш отчет как раз содержит объект с большим количеством текста, поэтому высота бэндов получается довольно большая. И если большой бэнд не помещается на страницу, он переносится на следующую, а внизу страницы остается много неиспользованного места. Это видно на следующем рисунке:



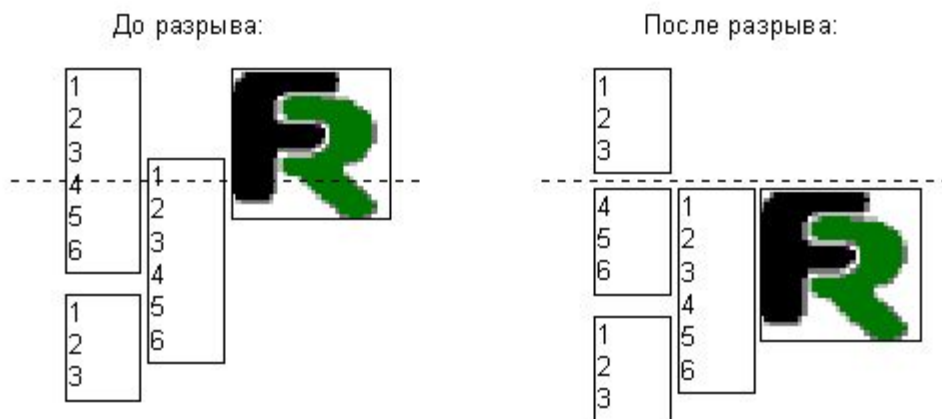
Чтобы рациональнее использовать бумагу, воспользуемся возможностью FastReport разбивать содержимое бэндов на части. Все, что нужно - это включить опцию "Разрыв" (AllowSplit) у бэнда "Данные 1 уровня". Мы видим, что пустого места внизу страниц отчета значительно прибавилось:



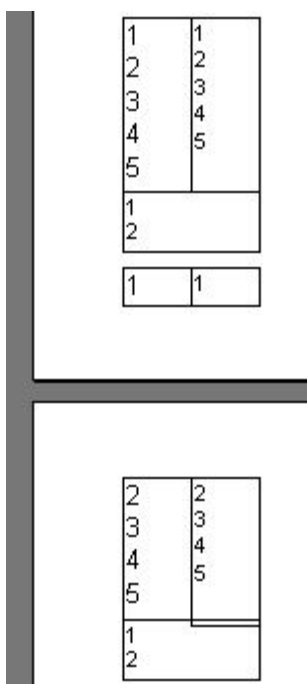
Как работает разрыв бэнда? В FastReport есть несколько объектов, которые поддерживают эту возможность. Это объекты "Текст", "Линия" и "RichEdit". Они могут быть "разорваны", остальные объекты - нет. Когда FastReport сталкивается с необходимостью выполнить разрыв, он делает следующее:

- выводит неразрываемые объекты, которые полностью помещаются на свободном месте;
- частично выводит разрываемые объекты (текстовые объекты выводятся таким образом, чтобы в объекте поместилось целое число строк);
- формирует новую страницу и продолжает вывод объектов;
- если неразрываемый объект не помещается на свободное место, он переносится на следующий лист, при этом все объекты, лежащие под ним, также смещаются;
- процесс продолжается до тех пор, пока не будут полностью выведены все объекты бэнда.

Алгоритм разрыва станет понятен, если взглянуть на рисунок:



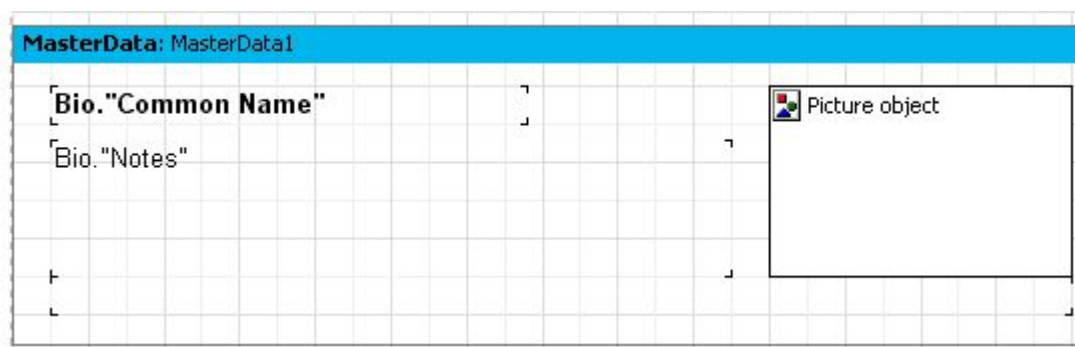
Следует отметить, что алгоритм разрыва не обеспечивает 100% качества получаемого отчета. Поэтому используйте эту опцию аккуратно, если объекты на разрываемом бэнде сгруппированы сложным образом и к тому же имеют разный размер шрифта. Вот пример того, что может получиться:



Обтекание объектов текстом

В некоторых случаях при оформлении отчета бывает необходимо сделать обтекание объектов (зачастую рисунков) текстом. Продемонстрируем такую возможность FastReport на примере с рыбами.

Добавим в отчет еще один объект "Текст" и расположим объекты следующим образом:



У объекта Bio."Notes" выключим растягивание, а у нижнего объекта, наоборот, включим. Чтобы текст "перетекал" из объекта Bio."Notes" в нижний объект, у объекта Bio."Notes" надо настроить свойство FlowTo. Это свойство настраивается в инспекторе объектов и имеет тип "выпадающий список". Из этого списка надо выбрать имя нижнего объекта. Результат будет выглядеть следующим образом:

Clown Triggerfish

Also known as the big spotted triggerfish. Inhabits outer reef areas and feeds upon crustaceans and mollusks by crushing them with powerful teeth. They are voracious eaters, and divers report seeing the clown triggerfish devour beds of pearl oysters.



Do not eat this fish. According to an 1878 account, "the poisonous flesh acts primarily upon the nervous tissue of the stomach, occasioning violent spasms of that organ, and shortly afterwards all the muscles of the body. The frame becomes rocked with spasms, the tongue thickened, the eye fixed, the breathing laborious, and the patient expires in a paroxysm of extreme suffering."

Not edible.

Range is Indo-Pacific and East Africa to Samoa.

Red Emperor

Called seaperch in Australia. Inhabits the areas around lagoon coral reefs and sandy bottoms.



The red emperor is a valuable food fish and considered a great sporting fish that fights with fury when hooked. The flesh of an old fish is just as tender to eat as that of the very young.

Range is from the Indo-Pacific to East Africa.

При построении отчета, когда текст не помещается в верхний объект, его невместившаяся часть переносится в нижний. Так как объекты расположены вокруг рисунка, создается эффект обтекания рисунка текстом.

Внимание: для правильной работы обтекания основной объект должен быть вставлен в отчет раньше, чем связанный! Если ваш отчет работает неправильно, выделите связанный объект и перенесите его на передний план командой меню "Правка|На передний план".

Печать данных в виде таблицы

Часто бывает необходимо отобразить отчет в виде таблицы с обрамлением. Один из примеров такого отчета – это прайс-лист. Чтобы построить такой отчет в FastReport, надо всего лишь включить обрамление у объектов, лежащих на бэнде "Данные". Рассмотрим несколько вариантов обрамления на примере тестового отчета.

Создадим пустой проект в Delphi. Положим на форму компонент TTable и настроим его свойства:

```
DatabaseName = 'DBDEMOS'  
TableName = 'Biolife.db'
```

Для того, чтобы работать с таблицей из FastReport, добавим компонент TfrxDBDataSet и настроим его свойства:

```
DataSet = Table1  
UserName = 'Bio'
```

Создадим отчет следующего вида:

PageHeader: PageHeader1		
MasterData: MasterData1		
[Bio."Specie"]	[Bio."Common Name"]	[Bio."Length"]
ReportSummary: ReportSummary1		

Разместим объекты на бэнде встык, а также уменьшим высоту бэнда до минимального размера.

Первый и самый простой тип таблицы – с полным обрамлением. Для этого надо у каждого объекта включить все линии рамки:

90020	Clown Triggerfish	50
90030	Red Emperor	60
90050	Giant Maori Wrasse	229
90070	Blue Angelfish	30
90080	Lunartail Rockcod	80

Следующий тип оформления – только горизонтальные или только вертикальные линии – делается аналогично, у объектов включается горизонтальное или вертикальное оформление.

90020	Clown Triggerfish	50
90030	Red Emperor	60
90050	Giant Maori Wrasse	229
90070	Blue Angelfish	30
90080	Lunartail Rockcod	80

Наконец, чтобы сделать только наружное оформление таблицы, надо слегка видоизменить отчет:

PageHeader: PageHeader1		
MasterData: MasterData1		
[Bio."Specie"]	[Bio."Common Name"]	[Bio."Length"]
ReportSummary: ReportSummary1		

Как видно, мы добавили два объекта "Текст" и включили линии рамки у крайних объектов на дата-бэнде. В результате отчет будет выглядеть следующим образом:

90020	Clown Triggerfish	50
90030	Red Emperor	60
90050	Giant Maori Wrasse	229
90070	Blue Angelfish	30
90080	Lunartail Rockcod	80
90090	Firefish	38
90100	Omate Butterflyfish	19
90110	Swell Shark	102
90120	Bat Ray	56
90130	California Moray	150
90140	Lingcod	150
90150	Cabezon	99
90160	Atlantic Spadefish	90
90170	Nurse Shark	400
90180	Spotted Eagle Ray	200
90190	Yellowtail Snapper	75
90200	Redband Parrotfish	28
90210	Great Barracuda	150
90220	French Grunt	30
90230	Dog Snapper	90
90240	Nassau Grouper	91
90250	Bluehead Wrasse	15
90260	Yellow Jack	90
90270	Redtail Surperch	40
90280	White Sea Bass	150
90290	Rock Greenling	60
90300	Senoiita	25
90310	Surf Smelt	25

Все вышеприведенные примеры содержали бэнды, которые имели фиксированный размер. Но как вывести таблицу, если бэнд растягиваемый? Покажем это на примере. Добавим в наш отчет новое поле – многострочный текст из Bio.Notes. Как мы уже знаем, надо включить свойство "Растягивание" у этого объекта и бэнда, на котором оно лежит. В этом случае высота бэнда будет подбираться в зависимости от количества текста в объекте "Текст". Мы получим отчет следующего вида:

90020	Clown Triggerfish	50	Also known as the big spotted triggerfish. Inhabits outer reef areas and feeds upon crustaceans and mollusks by crushing them with powerful teeth. They are voracious eaters, and divers report seeing the clown triggerfish devour beds of pearl oysters. Do not eat this fish. According to an 1878 account, "the poisonous flesh acts primarily upon the nervous tissue of the stomach, occasioning violent spasms of that organ, and shortly afterwards all the muscles of the body. The frame becomes racked with spasms, the tongue thickened, the eye fixed, the breathing laborious, and the patient expires in a paroxysm of extreme suffering." Not edible. Range is Indo-Pacific and East Africa to Samoa.
90030	Red Emperor	60	Called seaperch in Australia. Inhabits the areas around lagoon coral reefs and sandy bottoms.

Немного не то, что нам нужно – хотелось бы, чтобы рамки соседних объектов тоже растягивались. FastReport позволяет легко решить эту проблему. Для построения подобных отчетов достаточно включить у всех объектов, которые должны быть растянуты, свойство "Растягивание вниз" (или StretchMode = smMaxHeight в инспекторе объектов). При этом ядро FastReport сначала считает

максимальную высоту бэнда, затем "дотягивает" объекты с включенной опцией до нижнего края бэнда. Т.к. вместе с объектом растягивается и его рамка, в результате вид отчета меняется:

90020	Clown Triggerfish	50	Also known as the big spotted triggerfish. Inhabits outer reef areas and feeds upon crustaceans and mollusks by crushing them with powerful teeth. They are voracious eaters, and divers report seeing the clown triggerfish devour beds of pearl oysters. Do not eat this fish. According to an 1878 account, "the poisonous flesh acts primarily upon the nervous tissue of the stomach, occasioning violent spasms of that organ, and shortly afterwards all the muscles of the body. The frame becomes racked with spasms, the tongue thickened, the eye fixed, the breathing laborious, and the patient expires in a paroxysm of extreme suffering." Not edible. Range is Indo-Pacific and East Africa to Samoa.
90030	Red Emperor	60	Called seaperch in Australia. Inhabits the areas around lagoon coral reefs and sandy bottoms.

Печать этикеток

В отличие от табличных отчетов, данные в отчетах типа "этикетка" располагаются друг под другом. Рассмотрим пример подобного отчета, который выводит данные о рыбах (см. предыдущий пример) в виде этикеток. Отчет имеет следующую структуру:

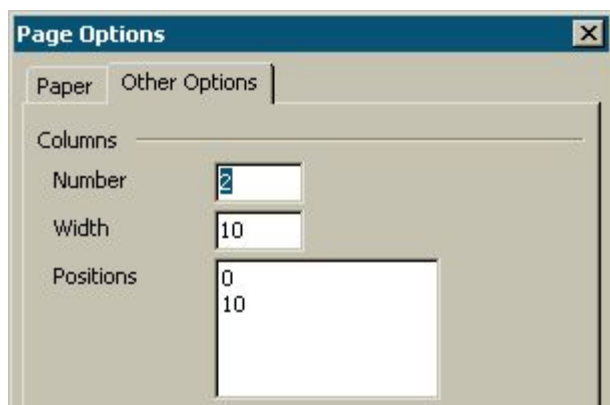
MasterData: Band3			
Номер:	[Bio."Species No"]		
Категория:	[Bio."Category"]		
Название:	[Bio."Common Name"]		
Длина, см:	[Bio."Length (cm)"]		

Если запустить отчет на выполнение, получим следующее:

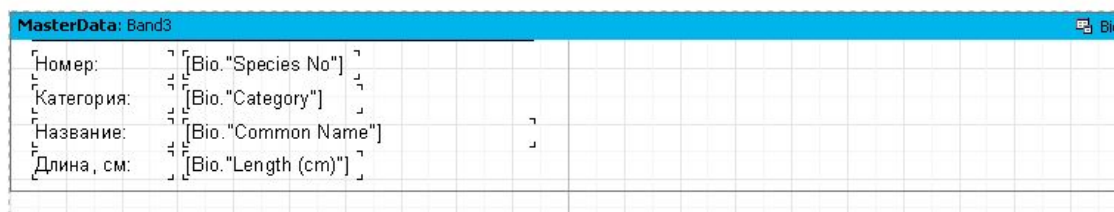
Номер:	90020
Категория:	Triggerfish
Название:	Clown Triggerfish
Длина, см:	50
Номер:	90030
Категория:	Snapper
Название:	Red Emperor
Длина, см:	60

Как видно, остается много неиспользованного места в правой части листа. Чтобы заполнить лист целиком, можно задать в настройках страницы отчета

количество колонок, в которых будут выводиться данные. Для этого сделайте двойной щелчок на пустом месте страницы, или вызовите пункт меню "Файл|Параметры страницы...".



Здесь можно задать количество колонок, ширину и позицию каждой колонки. В нашем случае достаточно указать количество = 2, остальные параметры FastReport подберет сам. Границы колонок показываются в дизайнера тонкой вертикальной линией:



MasterData: Band3	
Номер:	[Bio."Species No"]
Категория:	[Bio."Category"]
Название:	[Bio."Common Name"]
Длина, см:	[Bio."Length (cm)"]

При этом печать отчета будет происходить следующим образом. FastReport будет выводить бэнд "Данные 1 уровня" до тех пор, пока на странице не закончится свободное место. После этого сформируется не новая страница, как в обычном отчете, а новая колонка на этой же странице, и вывод бэндов продолжится сверху. Но теперь все объекты будут смещены вправо на ширину колонки. Так будет продолжаться до тех пор, пока не будет выведено заданное количество колонок. После этого FastReport сформирует новую страницу и продолжит выводить данные с первой колонки.

Наш отчет с двумя колонками будет выглядеть следующим образом:

Номер:	90020	Номер:	90140
Категория:	Triggerfish	Категория:	Cod
Название:	Clown Triggerfish	Название:	Lingcod
Длина, см:	50	Длина, см:	150
Номер:	90030	Номер:	90150
Категория:	Snapper	Категория:	Sculpin
Название:	Red Emperor	Название:	Cabezon
Длина, см:	60	Длина, см:	99

Есть еще один способ задать количество колонок – это свойство `Columns` у всех дата-бэндов. Оно позволяет задать количество колонок для отдельного бэнда, а не для всей страницы, как в предыдущем примере. При этом данные будут выводиться не "сверху вниз, потом слева направо", а "слева направо, потом сверху вниз".

В нашем примере отключим колонки у страницы (установим их количество = 1) и укажем 2 в свойстве `Columns` у бэнда. FastReport покажет штриховыми линиями границы колонок. Изменяя свойство `ColumnWidth` (ширина колонки), добьемся нужных размеров колонок:

MasterData: Band3		Bio	
Номер:	[Bio."Species No"]		
Категория:	[Bio."Category"]		
Название:	[Bio."Common Name"]		
Длина, см:	[Bio."Length (cm)"]		

Построенный таким образом отчет будет отличаться от предыдущего только тем, что данные будут выведены в порядке "слева направо, потом сверху вниз".

Child-бэнды

Рассмотрим случай, когда одна из строк в отчете типа "этикетка" может иметь переменный размер. Чтобы смоделировать ситуацию на нашем примере, уменьшим ширину объекта `Bio."Common Name"` до 2.5см и включим у него опцию "Растягивание". Также включим растягивание у бэнда "Данные 1 уровня". Также включим все линии рамки у всех объектов, чтобы лучше был виден принцип растягивания. Получится отчет следующего вида:

Номер:	90020
Категория:	Triggerfish
Название:	Clown
Длина, см:	Triggerfish
	50

Номер:	90030
Категория:	Snapper
Название:	Red Emperor
Длина, см:	60

Мы видим, что объект в первом случае содержит длинный текст и поэтому он растянулся на две строки. При этом лежащий под ним объект, привязанный к полю Bio."Length (cm)", сместился ниже. Произошло это потому, что по умолчанию все объекты имеют включенное свойство "Смещение" (или ShiftMode = smAlways в инспекторе объектов). Такие объекты смещаются вниз, если над ними есть растягиваемый объект (объект "Текст" с включенным свойством "Растягивание"). Высота, на которую смещается объект, зависит от того, насколько сильно растягивается лежащий над ним объект.

Однако в нашем случае это неприемлемо – нам нужно, чтобы объект с текстом "Длина, см:" также смещался. Для этого в FastReport есть специальный тип бэнда – Child-бэнд, или дочерний бэнд. Он привязывается к основному бэнду и выводится после него. Модифицируем наш отчет:

MasterData: MasterData1	
Номер:	Bio."Species No"
Категория:	Bio."Category"
Название:	Bio."Common"
Child: Child1	
Длина, см:	Bio."Length (cm)"

Для того, чтобы связать основной бэнд с дочерним, у бэнда "Данные 1 уровня" установим в инспекторе объектов свойство Child = Child1. Теперь каждый раз при печати основного бэнда будет выводиться и дочерний:

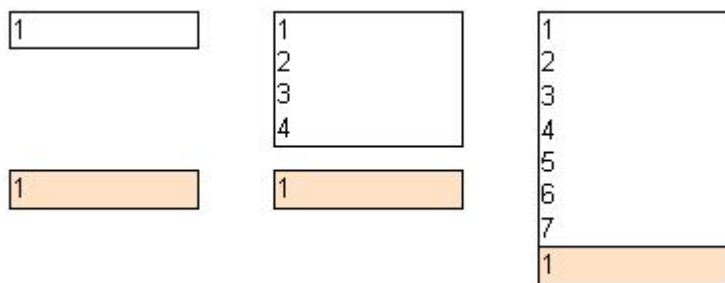
Номер:	90020
Категория:	Triggerfish
Название:	Clown Triggerfish
Длина, см:	50

Номер:	90030
Категория:	Snapper
Название:	Red Emperor
Длина, см:	60

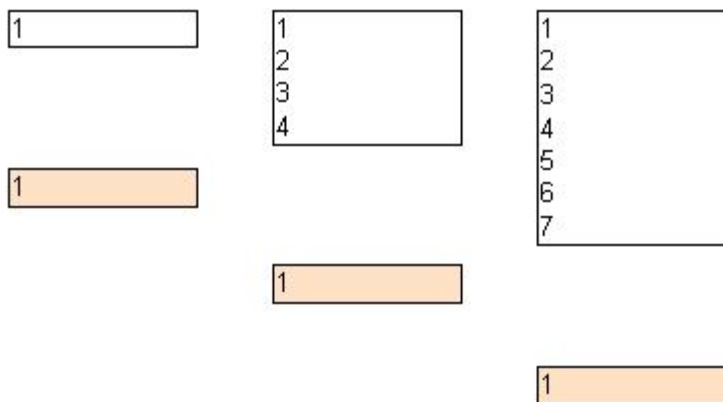
Как видно, теперь заголовок печатается там, где нужно. Для того, чтобы избежать переноса child-бэнда на следующую страницу (т.е. отрыва его от основного бэнда), установите у основного бэнда свойство "Не отрывать child" (KeepChild в инспекторе объектов).

Смещение объектов

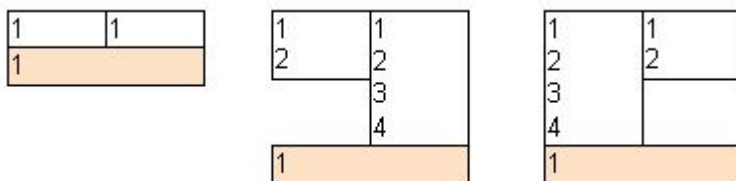
Мы уже видели, как работает свойство "Смещение". Рассмотрим другой режим работы смещения – "Смещение при перекрытии". В инспекторе объектов этому режиму соответствует значение свойства ShiftMode = smWhenOverlapped. При этом смещение объекта будет происходить только в том случае, если лежащий сверху объект при растягивании перекрыл данный объект. На рисунке ниже представлено три случая. Как мы видим, нижний объект со включенной опцией "Смещение при перекрытии" смещается только в последнем случае, когда в верхнем объекте много текста и он перекрывает нижний.



Если же включить опцию "Смещение", то нижний объект будет смещаться в любом случае:



В некоторых случаях это позволяет реализовать довольно сложную логику отрисовки объектов, особенно если один объект лежит сразу над несколькими. Так, в следующем примере оба верхних объекта содержат растягиваемый текст, а у нижнего объекта включена опция "Смещение при перекрытии". Независимо от количества текста в верхних объектах, нижний объект всегда будет выведен вплотную к тому объекту, который содержит больше текста:



Если же в этом примере у нижнего объекта включить опцию "Смещение", то нижний объект сместится дважды, т.к. он находится под двумя объектами, и образуется ненужный в данном случае зазор.

Отчет с двумя уровнями данных (master-detail)

До сих пор мы рассматривали отчеты, в которых присутствовал только один дата-бэнд – "Данные 1 уровня". Это давало возможность печатать данные из одной таблицы БД. FastReport позволяет печатать отчеты, содержащие до 6 уровней данных (можно и больше, используя объект "Вложенный отчет", но об этом позже). В реальных приложениях редко приходится печатать отчеты с большой вложенностью данных; как правило, ограничиваются 1-3 уровнями.

Рассмотрим создание двухуровневого отчета. Он будет содержать данные из таблиц DBDEMOS: Customer.db и Orders.db. Первая таблица – это список клиентов, вторая – список заказов, сделанных клиентами. Таблицы содержат данные следующего вида:

Customer:

CustNo	Company
1221	Kauai Dive Shoppe
1231	Unisco
1351	Sight Diver
....	

Orders:

OrderNo	CustNo	SaleDate
1003	1351	12.04.1988
1023	1221	01.07.1988
1052	1351	06.01.1989
1055	1351	04.02.1989
1060	1231	28.02.1989
1123	1221	24.08.1993
....		

Как видно, вторая таблица содержит список всех заказов, сделанных всеми компаниями. Чтобы получить список заказов, сделанных конкретной компанией, из таблицы следует отобрать записи, у которых поле CustNo = номеру выбранной компании. Отчет, построенный на таких данных, будет выглядеть следующим образом:

1221	Kauai Dive Shoppe
1023	01.07.1988
1123	24.08.1993
1231	Unisco
1060	28.02.1989
1351	Sight Diver
1003	12.04.1988
1052	06.01.1989
1055	04.02.1989

Приступим к созданию отчета. Создадим новый проект в Delphi, на форму положим два компонента TTable, компонент TDataSource, два компонента TfrxDBDataSet и один TfrxReport. Настроим компоненты следующим образом:

Table1:
DatabaseName = 'DBDEMOS'
TableName = 'Customer.db'

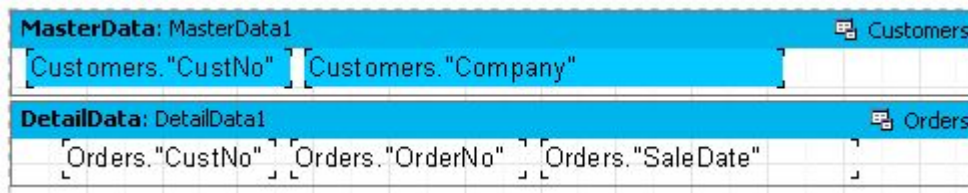
Table2:
DatabaseName = 'DBDEMOS'
TableName = 'Orders.db'

```
DataSource1:  
DataSet = Table1
```

```
frxDBDataSet1:  
DataSet = Table1  
UserName = 'Customers'
```

```
frxDBDataSet2:  
DataSet = Table2  
UserName = 'Orders'
```

В дизайнере отчета подключим наши источники данных в окне "Отчет|Данные...". Положим на страницу бэнды "Данные 1 уровня" и "Данные 2 уровня":



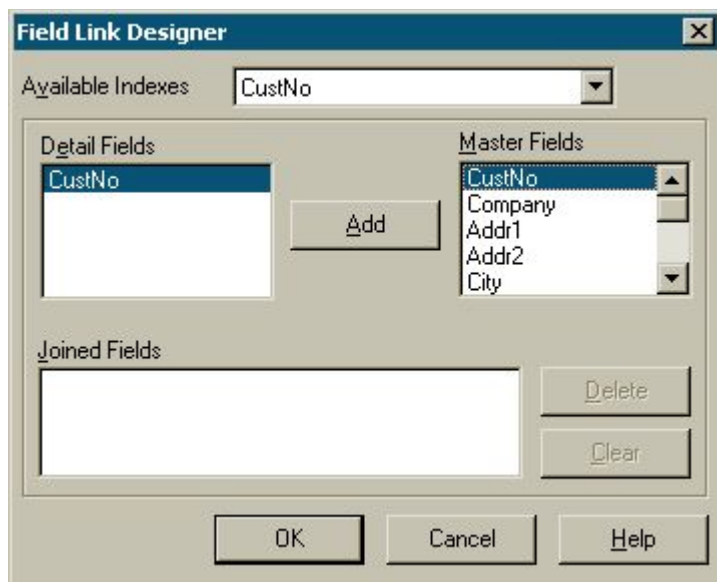
MasterData: MasterData1		Customers
Customers.CustNo	Customers.Company	

DetailData: DetailData1			Orders
Orders.CustNo	Orders.OrderNo	Orders.SaleDate	

Обратите внимание – бэнд "Данные 1 уровня" должен располагаться выше! Если разместить его под бэндом "Данные 2 уровня", FastReport сообщит об ошибке при запуске отчета.

Связывание данных

Если сейчас запустить отчет, мы увидим, что список заказов одинаковый для каждого клиента и содержит все записи из таблицы Orders.db. Это произошло потому, что мы не включили фильтрацию записей в таблице Orders. Вернемся к нашим источникам данных. У компонента Table2 установим свойство *MasterSource* = *DataSource1*. Таким образом мы установили связь "главный-подчиненный". Теперь надо задать условие фильтрации записей в подчиненном источнике. Для этого вызовите редактор свойства MasterFields у компонента Table2:



Нам надо связать два поля CustNo в обоих источниках. Для этого выберите индекс CustNo в списке сверху, выберите поля и нажмите кнопку "Add". Связка полей переместится в нижнее окно. После этого закройте редактор кнопкой ОК.

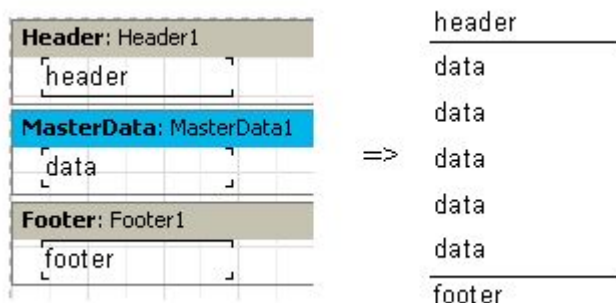
При запуске отчета FastReport сделает следующее. Выбрав очередную запись из главной таблицы (Customer), он установит фильтр на подчиненную таблицу (Orders). В таблице останутся только те записи, которые удовлетворяют условию `Orders.CustNo = Customer.CustNo`. Т.е. для каждого клиента будут показаны только его заказы:

1221	Kauai Dive Shoppe	
1221	1023	01.07.88
1221	1076	16.12.94
1221	1123	24.08.93
1221	1169	06.07.94
1221	1176	26.07.94
1221	1269	16.12.94
1231	Unisco	
1231	1060	28.02.89
1231	1073	15.04.89
1231	1102	06.06.92
1231	1160	01.06.94

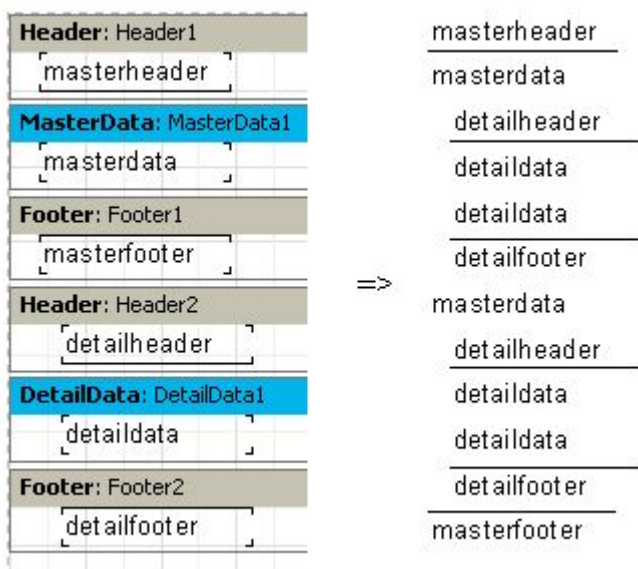
Аналогичным образом можно строить отчеты, содержащие до 6 уровней данных.

Заголовок и подвал данных

Дата-бэнды могут иметь заголовок и подвал. Заголовок выводится перед печатью дата-бэнда, подвал выводится после печати последнего дата-бэнда. Вот пример того, как работают заголовки и подвалы при печати простого отчета:



Рассмотрим печать заголовков и подвалов на примере отчета master-detail:



Как видно, заголовок печатается перед началом печати всех записей дата-бэнда. Для бэнда "Данные 1 уровня" это происходит один раз в начале отчета, а для бэнда "Данные 2 уровня" – каждый раз при печати очередной группы бэндов, привязанных к бэнду "Данные 1 уровня". Подвал же печатается после того, как напечатаны все записи бэнда.

Однако, используя свойство дата-бэнда `FooterAfterEach` (или пункт контекстного меню "Footer после каждой записи"), можно вывести подвал после каждой строки данных. Это может оказаться полезным при печати отчетов типа

master-detail. Предыдущий пример с включенным у бэнда "Данные 1 уровня" свойством FooterAfterEach будет выглядеть так:

```

masterheader
masterdata
  detailheader
  detaildata
  detaildata
  detailfooter
masterfooter
masterdata
  detailheader
  detaildata
  detaildata
  detailfooter
masterfooter

```

Отчет с группами

В предыдущем примере мы строили двухуровневый отчет на основе данных из двух таблиц. FastReport позволяет построить аналогичный отчет на основе одного набора данных, сформированного особым способом.

Для этого необходимо составить запрос на языке SQL, который вернет данные из обеих таблиц, сгруппированные по определенному условию. В нашем случае условие – соответствие полей CustNo в обеих таблицах. SQL-запрос может выглядеть следующим образом:

```

select * from customer, orders
where orders.CustNo = customer.CustNo
order by customer.CustNo

```

Строка "order by" нужна для сортировки записей по полю CustNo. Запрос вернет данные примерно следующего вида:

CustNo	Company	...	OrderNo	SaleDate
1221	Kauai Dive Shoppe		1023	01.07.1988
1221	Kauai Dive Shoppe		1123	24.08.1993
1231	Unisco		1060	28.02.1989
1351	Sight Diver		1003	12.04.1988
1351	Sight Diver		1052	06.01.1989
1351	Sight Diver		1055	04.02.1989

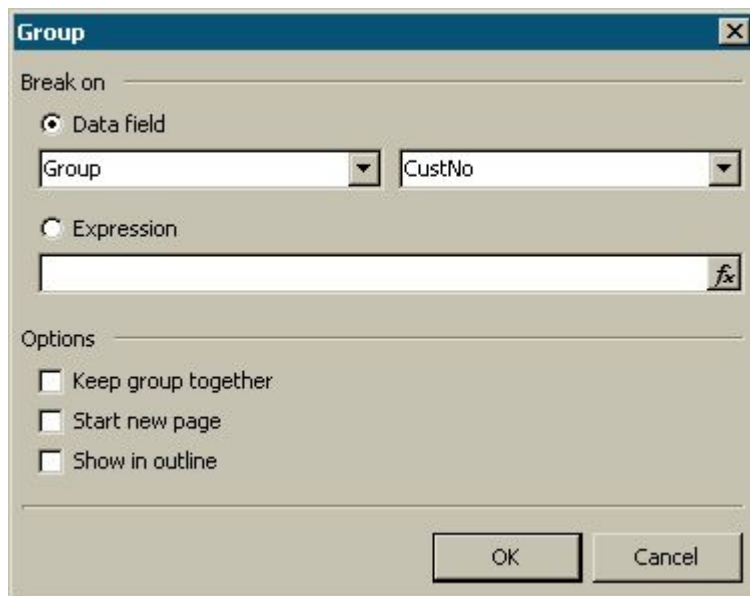
Как на основе этих данных построить многоуровневый отчет? В FastReport для этого есть специальный бэнд – "Заголовок группы". У бэнда задается условие (значение поля БД или выражение), при смене которого происходит вывод бэнда. Продемонстрируем это на примере.

Создадим новый проект в Delphi, на форму положим компоненты TQuery, TfrxReport, TfrxDBDataSet. Настроим их следующим образом:

```
Query1:
DatabaseName = 'DBDEMOS'
SQL =
select * from customer, orders
where orders.CustNo = customer.CustNo
order by customer.CustNo
```

```
frxDBDataSet1:
DataSet = Query1
UserName = 'Group'
```

Зайдем в дизайнер и подключим наш источник данных к отчету. Добавим в отчет два бэнда: "Заголовок группы" и "Данные 1 уровня". В редакторе бэнда "Заголовок группы" укажем условие – поле данных Group.CustNo:



Дата-бэнд привяжем к источнику данных Group и разместим объекты следующим образом (обратите внимание, что заголовок группы должен располагаться над дата-бэндом):

GroupHeader: GroupHeader1		Group."CustNo"
Group."CustNo"	Group."Company"	
MasterData: MasterData1		Group
Group."OrderNo"	Group."SaleDate"	

При запуске получится отчет следующего вида:

1221	Kauai Dive Shoppe
1269	16.12.94
1023	01.07.88
1176	26.07.94
1076	16.12.94
1123	24.08.93
1169	06.07.94
1231	Unisco
1173	16.07.94
1178	02.08.94

Как видно, бэнд "Заголовок группы" выводится только в том случае, когда поле, к которому он подключен, меняет свое значение. В остальных случаях выводится связанный с группой дата-бэнд. Если сравнить этот отчет с отчетом master-detail, который мы строили ранее, то видно, что номера заказов здесь не отсортированы по возрастанию. Это легко исправить, изменив текст запроса SQL:

```
select * from customer, orders
where orders.CustNo = customer.CustNo
order by customer.CustNo, orders.OrderNo
```

Аналогичным образом можно строить отчеты с вложенными группами, при этом количество вложений не ограничено. Таким образом, отчеты с группами имеют ряд преимуществ над отчетами типа master-detail:

- требуется только одна таблица (запрос) для всего отчета;
- число уровней вложенности данных неограничено;
- возможность дополнительной сортировки данных;
- более рациональное использование ресурсов СУБД – запрос возвращает только те данные, которые должны быть напечатаны, отсутствует фильтрация данных.

Единственный минус – необходимость написания запросов на языке SQL. Впрочем, знание основ SQL является обязательным для программиста, работающего с базами данных.

Другие особенности групп

Обратим внимание на то, как группа переносится на следующую страницу:

1380	Blue Jack Aqua Center
1006	06.11.94
1079	03.05.89
1106	23.09.92
1153	16.04.94
1253	26.11.94
1384	VIP Divers Club
1007	01.05.88
1027	07.07.88

Если листать распечатку такого отчета, то непонятно, к какому клиенту относится список заказов в самом верху второй страницы. FastReport позволяет повторить вывод заголовка группы (который в нашем случае содержит информацию о клиенте), на следующей странице. Для этого у бэнда "Заголовок группы" надо включить свойство "Повторять заголовок" (или свойство ReprintOnNewPage в инспекторе объектов). При этом отчет будет выглядеть так:

1380	Blue Jack Aqua Center
1006	06.11.94
1380	Blue Jack Aqua Center
1079	03.05.89
1106	23.09.92
1153	16.04.94
1253	26.11.94
1384	VIP Divers Club
1007	01.05.88

Есть еще способ, позволяющий избежать разрыва группы. Для этого надо включить свойство заголовка группы "Не разрывать группу" (или KeepTogether в инспекторе объектов). При этом, если вся группа не помещается на странице, ее вывод переносится на новую страницу. В нашем примере это будет выглядеть так:

1356	Tom Sawyer Diving Centre
1005	20.04.88
1059	24.02.89
1072	11.04.89
1080	05.05.89
1105	21.07.92
1180	06.08.94
1266	15.12.94
1280	26.12.94
1305	20.01.95
1380	Blue Jack Aqua Center
1006	06.11.94
1079	03.05.89

При этом на некоторых страницах может образоваться много пустого места, но вся группа будет выведена целиком на странице.

Наконец, свойство "Начинать новую страницу" (StartNewPage) заголовка группы позволит выводить каждую группу на отдельной странице, что приведет к нерациональному использованию бумаги, но может понадобиться в некоторых случаях.

Нумерация записей

Давайте рассмотрим на нашем примере, как пронумеровать записи в группе (аналог графы Nп/п). Для этого добавим объект "Текст" с системной переменной [Line] внутри на оба наших бэнда (проще всего это сделать методом drag&drop из закладки "Переменные" служебного окна "Данные").

GroupHeader: GroupHeader1		
[Line]	Group."Cust"	Group."Company"
MasterData: MasterData1		
[Line]	Group."Orde"	Group."SaleDate"

Запустив отчет, мы увидим, что оба уровня данных теперь имеют свой порядковый номер:

1	1221	Kauai Dive Shoppe
1	1023	01.07.88
2	1076	16.12.94
3	1123	24.08.93
4	1169	06.07.94
5	1176	26.07.94
6	1269	16.12.94
2	1231	Unisco
1	1060	28.02.89
2	1073	15.04.89
3	1102	06.06.92
4	1160	01.06.94

В некоторых отчетах нам может понадобиться сквозная нумерация данных второго уровня. Для этого в нашем примере надо использовать переменную Line# вместо Line на дата-бэнде. Результат получится следующим:

1	1221	Kauai Dive Shoppe
1	1023	01.07.88
2	1076	16.12.94
3	1123	24.08.93
4	1169	06.07.94
5	1176	26.07.94
6	1269	16.12.94
2	1231	Unisco
7	1060	28.02.89
8	1073	15.04.89
9	1102	06.06.92

Агрегатные функции

В большинстве случаев в групповых отчетах надо выводить некую итоговую информацию: сумма по группе, количество элементов группы и т.п. В FastReport для этих целей существуют так называемые агрегатные функции. С их помощью можно подсчитать функцию от определенного значения по диапазону данных. Ниже приведен список агрегатных функций:

SUM	Возвращает сумму заданного выражения
MIN	Возвращает минимальное значение заданного выражения
MAX	Возвращает максимальное значение заданного выражения
AVG	Возвращает среднее значение заданного выражения
COUNT	Возвращает количество строк в диапазоне данных

Синтаксис всех агрегатных функций (за исключением COUNT) следующий (рассмотрим на примере ф-и SUM):

```
SUM(expression, band, flags)
SUM(expression, band)
SUM(expression)
```

Назначение параметров следующее:

expression – выражение, значение которого необходимо обработать
band – имя дата-бэнда, по которому будет идти обработка
flags – битовое поле, которое может содержать следующие значения и их комбинации:

- 1 – учитывать невидимые бэнды
- 2 – накапливать значение (не сбрасывать при очередном выводе)

Как видно, обязательным параметром является только `expression`, остальные при вызове функции могут быть опущены. Тем не менее, рекомендуется всегда использовать параметр `band`, это позволит избежать ошибок.

Функция `COUNT` имеет следующий синтаксис:

```
COUNT(band, flags)
COUNT(band)
```

Назначение параметров аналогично вышеописанным.

Существует общее для всех агрегатных функций правило: функция может быть подсчитана только для дата-бэнда и выведена только в бэнде-подвале (к последним относятся бэнды: подвал, подвал страницы, подвал группы, подвал колонки, подвал отчета).

Как работают агрегатные функции? Рассмотрим это на примере нашего отчета с группами. Добавим в отчет новые элементы:

GroupHeader: GroupHeader1		
[Group."Cust"]	[Group."Company"]	
MasterData: MasterData1		
[Group."Orde"]	[Group."SaleDate"]	[Group."ItemsTotal"]
GroupFooter: GroupFooter1		
[SUM(<Group."ItemsTotal">,MasterData1)]		

Поле `Group."ItemsTotal"` на дата-бэнде будет отображать сумму текущего заказа. В подвал группы мы поместили объект "Текст", содержащий вызов агрегатной суммы. Он будет отображать сумму всех заказов по данному клиенту. Запустив отчет на выполнение, и вооружившись калькулятором, мы убедимся, что все работает:

1221	Kauai Dive Shoppe	
1023	01.07.88	4 674,00p.
1076	16.12.94	17 781,00p.
1123	24.08.93	13 945,00p.
1169	06.07.94	9 471,95p.
1176	26.07.94	4 178,85p.
1269	16.12.94	1 400,00p.
		51450,8

Итак, каков принцип работы агрегатных функций? Перед построением отчета FastReport сканирует содержимое объектов "Текст" с целью нахождения агрегатных функций. Найденные функции привязываются к соответствующим дата-бэндам (в нашем примере функция SUM привязывается к бэнду MasterData1). При построении отчета, когда дата-бэнд выводится на экран, подсчитывается значение связанных с ним агрегатных функций. В нашем случае накапливается сумма значений поля Group."ItemsTotal". После вывода подвала группы, в котором отображается накопленное значение агрегатной функции, значение функции сбрасывается и цикл повторяется для следующих групп.

Здесь можно пояснить назначение параметра flags в агрегатных функциях. В некоторых отчетах часть дата-бэндов (или все) могут быть скрыты, однако нам все равно может понадобиться посчитать значение агрегатной функции с учетом всех дата-бэндов. Так, в нашем примере можно отключить свойство Visible у дата-бэнда, после этого он перестанет выводиться на экран. Чтобы подсчитать сумму по скрытым дата-бэндам, добавим третий параметр в вызов функции:

```
[SUM(<Group."ItemsTotal">,MasterData1,1)]
```

Это даст нам отчет следующего вида:

1221	Kauai Dive Shoppe	51450,8
1231	Unisco	85643,6
1351	Sight Diver	261575,8

Значение параметра flags = 2 позволяет не сбрасывать накопленное значение функции после ее вывода. Это позволяет печатать так называемые нарастающие итоги. Модифицируем вызов функции:

```
[SUM(<Group."ItemsTotal">,MasterData1,3)]
```

Значение 3 – это битовая комбинация 1 и 2, что означает, что нам надо учитывать невидимые бэнды и не сбрасывать сумму. В итоге получится следующее:

1221	Kauai Dive Shoppe	51450,8
1231	Unisco	137094,4
1351	Sight Diver	398670,2

Итоги по странице и по отчету

Довольно часто приходится отображать в отчете итоговое значение по странице или по всему отчету. Это также делается с помощью агрегатных функций. Рассмотрим это на нашем примере.

GroupHeader: GroupHeader1		
[Group."Cust"]	[Group."Company"]	
MasterData: MasterData1		
[Group."Orde"]	[Group."SaleDate"]	[Group."ItemsTotal"]
GroupFooter: GroupFooter1		
	[SUM(<Group."ItemsTotal">,MasterData1)]	
ReportSummary: ReportSummary1		
[Bcero: [SUM(<Group."ItemsTotal">,MasterData1)]]		
PageFooter: PageFooter1		
[На этой странице: [SUM(<Group."ItemsTotal">,MasterData1)]]		

Как видим, мы добавили бэнд "Подвал отчета" и объект "Текст" с суммой на бэнды "Подвал отчета" и "Подвал страницы". Это все, что нам нужно.

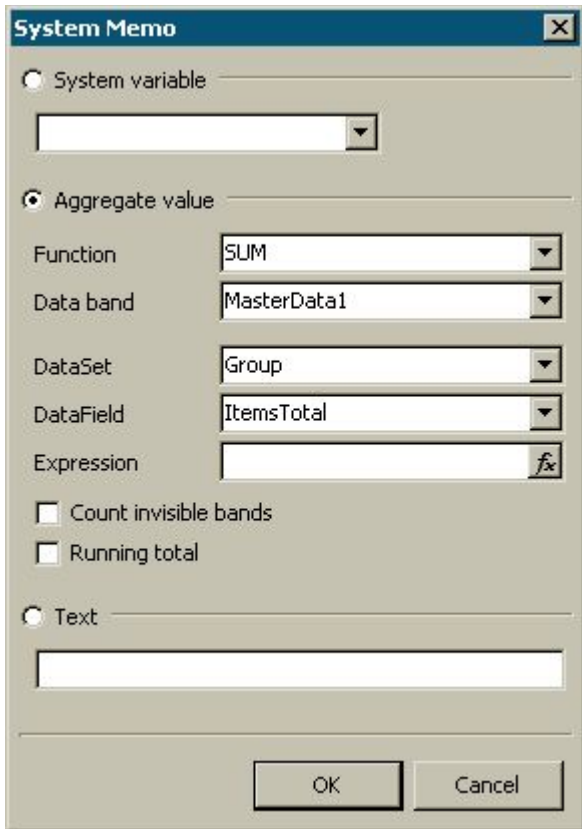
0841	Neptune's Trident Supply	
1045	16.10.88	787,80р.
1049	13.12.88	1 809,85р.
1145	17.01.94	4 229,80р.
1149	14.03.94	12 900,75р.
		19728,2
Всего: 2922666,1		

На этой странице: 101298,95


Вставка агрегатной функции

До сих пор мы вставляли агрегатные функции в объект "Текст" вручную. Рассмотрим более удобные способы вставки агрегатных функций.

Во-первых, мы можем использовать для вывода значения агрегатной функции объект "Системный текст". По сути, это тот же самый объект "Текст", но имеющий специальный редактор для более удобной вставки системных переменных или агрегатных функций.



В редакторе надо последовательно выбрать тип функции, дата-бэнд, по которому она будет считаться, и поле БД или выражение, значение которого будет вычисляться. Также можно отметить флажки "Учитывать невидимые бэнды" и "Нарастающие итоги".

Второй способ – использовать объект "Текст" и кнопку  в его редакторе. При этом открывается дополнительное окно, аналогичное рассмотренному редактору объекта "Системный текст". При нажатии кнопки ОК в текст объекта вставляется вызов агрегатной функции.

Особенности вызова агрегатной функции

Помните, мы обсуждали в предыдущей главе, в каких случаях использовать квадратные и угловые скобки при вставке выражений в объект "Текст"? Напомним, что в угловые скобки надо заключать все выражения, не являющиеся стандартными с точки зрения интерпретатора языка Pascal (а именно он используется для вычисления значения выражений). В эту группу попадают поля БД (обращение к ним идет через специальную конструкцию вида *ИмяТаблицы.ИмяПоля*), переменные из списка переменных, системные переменные.

Из-за особенностей реализации агрегатных функций при их вызове необходимо применять угловые скобки. Так, следующая форма записи некорректна:

$$[SUM(<Group."ItemsTotal">,MasterData1) * 2]$$

а следующая – верна:

$$[<SUM(<Group."ItemsTotal">,MasterData1)> * 2]$$

Напомним также, что в случае, когда в квадратных скобках содержится единственный член выражения, FastReport допускает опускать угловые скобки, т.е. формы записи

$$[<SUM(<Group."ItemsTotal">,MasterData1)>]$$

$$[SUM(<Group."ItemsTotal">,MasterData1)]$$

идентичны.

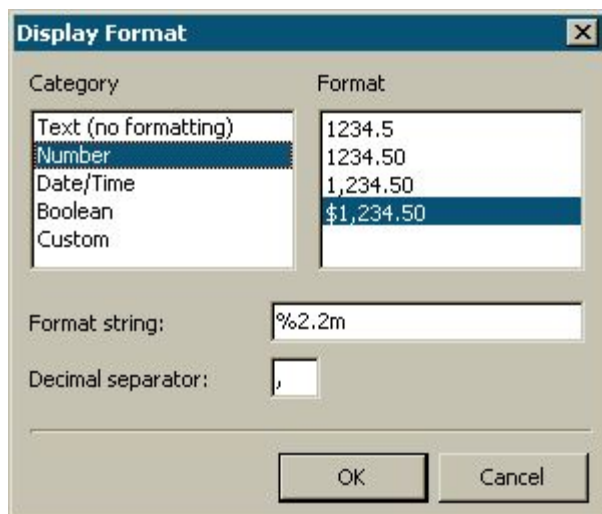
Форматирование значений

Обратим внимание на одну особенность при использовании агрегатных функций: они возвращают неформатированное числовое значение. Особенно это бросается в глаза в первом примере с функцией SUM:

1176	26.07.94	4 178,85р.
1269	16.12.94	1 400,00р.
		<hr/>
		51450,8

Это происходит потому, что поля данных, как правило, возвращают форматированное значение, которое просто отображается объектом "Текст" без изменения. Чтобы привести результат функции SUM к внешнему виду, воспользуемся встроенными в FastReport средствами для форматирования значений.

Выделим объект с суммой и вызовем его контекстное меню. Редактор формата вызывается командой меню "Форматирование..." или с помощью редактора свойства DisplayFormat в инспекторе объектов.



Как видно, слева располагается список категорий форматирования, а справа – список форматов в выбранной категории. Выберем категорию "Число", формат "1 234,50р.". При этом внизу отобразится строка форматирования, соответствующая выбранному формату, и символ-разделитель дроби. Строка форматирования – не что иное, как аргумент делфийской функции Format, с помощью которой FastReport выполняет форматирование чисел. Вы можете поменять как строку форматирования, так и разделитель (в отечественной бухгалтерии часто используют знак "—" в качестве разделителя рублей и копеек).

После нажатия клавиши ОК и построения отчета мы увидим, что теперь сумма в отчете приняла должный вид:

1176	26.07.94	4 178,85р.
1269	16.12.94	1 400,00р.
		<hr/>
		51 450,80р.

Форматирование по месту

Рассмотренный способ форматирования применяется ко всем выражениям, которые имеются в объекте. В нашем случае все работает правильно, т.к. в объекте всего одно выражение. Однако, как быть, если их два, да еще и разного типа?

Рассмотрим такой случай: вывод в одном объекте суммы и количества заказов. Для этого в объект надо поместить следующий текст:

Сумма: [SUM(<Group."ItemsTotal">,MasterData1)]
Кол-во: [COUNT(MasterData1)]

При запуске отчета убедимся, что оба значения представлены в денежном формате (который мы задали в предыдущем примере), что отчасти неверно:

1269	16.12.94	1 400,00р.
		Сумма: 51 450,80р.
		Кол-во: 6,00р.

Для корректного вывода значений надо бы отформатировать каждое из них индивидуально. Для этого есть способ – так называемые тэги формата. Они дописываются перед закрывающей квадратной скобкой выражения. В нашем примере отключим форматирование объекта (в редакторе формата выберем категорию "Текст (без форматирования)"). Теперь нужно поменять формат первой переменной, т.к. вторая будет отображена правильно (без форматирования – в виде целого числа, что нам и надо). Для этого поменяем текст объекта следующим образом:

Сумма: *[SUM(<Group."ItemsTotal">,MasterData1) #n%2,2m]*
 Кол-во: *[COUNT(MasterData1)]*

и убедимся, что теперь отчет работает правильно:

1269	16.12.94	1 400,00р.
		Сумма: 51 450,80р.
		Кол-во: 6

Теперь о том, как использовать тэги. Общий синтаксис следующий:

[expression #tag]

Обратите внимание – пробел между выражением и знаком # обязателен! Сам тэг может быть следующего вида:

#n*СтрокаФорматирования* – числовой формат
 #d*СтрокаФорматирования* – формат даты/времени
 #b*Ложь,Истина* – булевый формат

СтрокаФорматирования в каждом случае представляет собой аргумент для функции, с помощью которой выполняется форматирование. Так, для числового форматирования это делфийская функция *Format*, для даты/времени – функция *FormatDateTime*. Возможные значения строк форматирования можно узнать в справочной системе Delphi. Вот некоторые значения, используемые в FastReport:

для числового форматирования:

%g – число с минимальным количеством знаков после запятой
 %2.2f – число с фиксированным количеством знаков после запятой
 %2.2n – число с разделителем разрядов
 %2.2m – денежный формат, принятый в ОС Windows, зависит от региональных настроек в панели управления.

для формата дата/время:

dd.mm.yyyy – дата вида 23.12.2003

dd mmm yyyy – дата вида 23 ноя 2003

dd mmmm yyyy – дата вида 23 Ноябрь 2003


hh:mm – время вида 23:12

hh:mm:ss – время вида 23:12:00

dd mmmm yyyy, hh:mm – время и дата вида 23 Ноябрь 2003, 23:12


В строке для числового формата допускается указывать вместо точки запятую или тире, тогда этот символ будет использован как разделитель целой и дробной частей числа. Использование других разделителей не допускается.

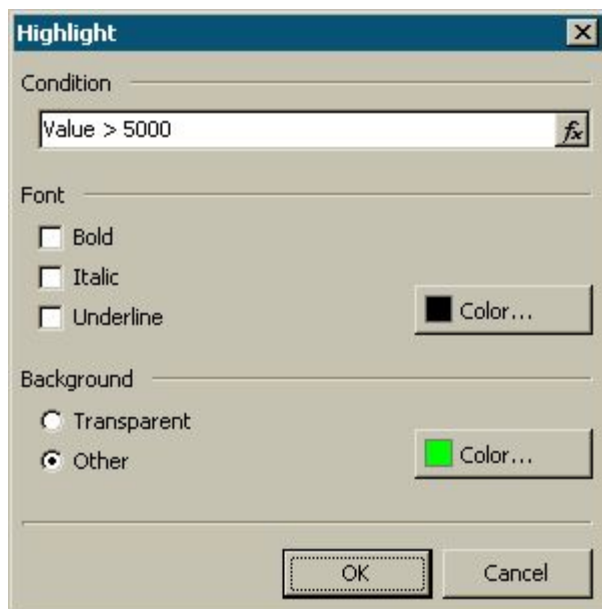
Что касается форматирования типа #b (булево), то строка форматирования представляется в виде двух значений, разделенных запятой. Первое значение соответствует False, второе – True.

Чтобы не запоминать все тэги и их значения, в редакторе объекта "Текст" есть удобное средство для вставки форматирования. При нажатии на кнопку  вызывается редактор формата, который мы уже рассматривали. После выбора формата он вставляется в текст. При этом, если курсор стоит перед или после закрывающей квадратной скобки, то формат вставляется корректно.

Условное выделение

Эта особенность объекта "Текст" позволяет выделить объект цветом в зависимости от какого-либо условия. Условием может быть любое выражение. Рассмотрим выделение на примере с группами: пусть суммы заказа более 5000 будут выделены зеленым цветом. Для этого выберем объект с полем

Group."ItemsTotal" и нажмем кнопку "Условное выделение"  на панели инструментов дизайнера. В открывшемся редакторе условного выделения впишем условие, при выполнении которого объект будет выделен, и укажем атрибуты выделения – параметры шрифта и цвет фона.



Результат будет следующим:

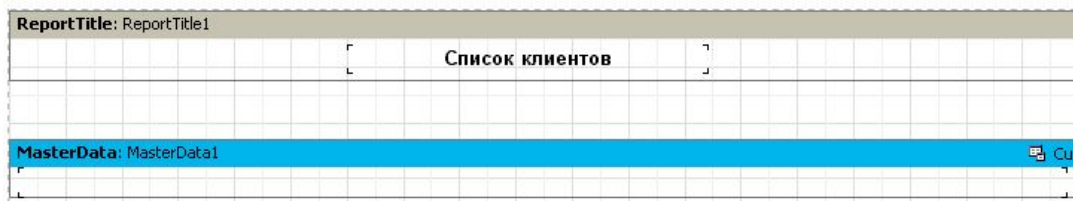
1221	Kauai Dive Shoppe	
1023	01.07.88	4 674,00p.
1076	16.12.94	17 781,00p.
1123	24.08.93	13 945,00p.
1169	06.07.94	9 471,95p.
1176	26.07.94	4 178,85p.
1269	16.12.94	1 400,00p.
		Сумма: 51 450,80p.

Обратите внимание на условие, которое мы указали – Value > 5000. Value – это значение поля БД, к которому прикреплен объект. С тем же успехом можно указать условие <Group."ItemsTotal"> > 5000. Вообще говоря, здесь можно указывать любое корректное с точки зрения FastReport выражение.

Выделение строк через одну

С помощью условного выделения можно легко придать отчету более современный вид, "раскрасив" каждую вторую строку данных. Покажем это на примере отчета типа "Список", который мы строили в предыдущей главе.

Для начала разместим на листе бэнды "Заголовок отчета" и "Данные 1 уровня". На дата-бэнд положим объект "Текст" и растянем его таким образом, чтобы он занимал почти все пространство бэнда:



Этот объект будет выполнять роль подложки, которая будет менять цвет в зависимости от номера строки данных. Выделим объект и установим в редакторе выделения следующее условие:

`<Line> mod 2 = 1`

Цвет выделения выберем серый, но не слишком насыщенный (ближе к белому).

Теперь на дата-бэнд можно класть остальные объекты:




Поскольку новые объекты лежат на подложке, ее легко не заметить. Если запустить отчет, мы увидим следующее:

Список клиентов			
1645	Action Club	813-870-0239	813-870-0282
3158	Action Diver Supply	22-44-500211	22-44-500596
1984	Adventure Undersea	011-34-09054	011-34-09064
3053	American SCUBA Supply	213-654-0092	213-654-0095
6312	Aquatic Drama	613-442-7654	613-442-7678
3984	Blue Glass Happiness	213-555-1984	213-555-1995
1380	Blue Jack Aqua Center	401-609-7623	401-609-9403
1563	Blue Sports	610-772-6704	610-772-6898
2118	Blue Sports Club	612-897-0342	612-897-0348
3054	Catamaran Dive Club	213-223-0941	213-223-2324
1354	Cayman Divers World Unlimited	011-5-697044	011-5-697064
5151	Central Underwater Supplies	27-11-4432458	27-11-4433259
2156	Davy Jones' Locker	803-509-0112	803-509-0553
3055	Diver's Grotto	213-432-0093	213-432-4821


Многостраничные отчеты

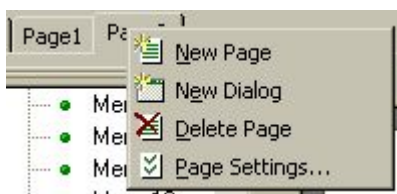
Отчет FastReport может содержать несколько страниц. Для каждой страницы вы можете задать свой размер, ориентацию, расположить на ней разные объекты и

бэнды. При построении отчета сначала будут выведены все бэнды первой страницы, потом – второй и т.д.

Когда мы создаем новый отчет в дизайнера, он уже содержит одну страницу по умолчанию. Вы можете добавить новую страницу, нажав кнопку  на панели инструментов или выбрав команду меню "Файл|Новая страница". Теперь мы видим, что в дизайнера появилась новая закладка:



Можно легко переключаться между страницами, нажав на нужную закладку мышью. Кроме того, закладки можно перетаскивать (drag&drop), тем самым легко меняя порядок страниц. Ненужную страницу можно удалить с помощью кнопки  на панели инструментов или команды меню "Правка|Удалить страницу". Также можно вызвать контекстное меню, щелкнув правой кнопкой мыши на самой закладке:



Количество страниц в отчете не ограничено. Как правило, дополнительные страницы используются для вывода титульного листа, либо в более сложных отчетах, содержащих данные из многих источников.


Рассмотрим простой пример создания титульного листа. Возьмем отчет с одним уровнем данных, который мы строили ранее. Добавим в него новую страницу – при этом она будет второй по порядку. Чтобы переместить ее в начало отчета, схватим мышью закладку страницы и переместим ее перед первой страницей. При этом порядок страниц изменится. Переключимся на новую страницу и разместим посередине листа объект "Текст" с текстом "Наш отчет" внутри. Все, отчет с титульным листом готов:

Customers				
Company	Address	Contact	Phone	Fax
Action Club	PO Box 9481-F	Michael Spurling	813-470-0280	813-470-0282
Action Diver Supply	Blue Spar Box 63	Marlene Miller	22-44-800-211	22-44-800-086
Adventure Undersea	PO Box 744	Gloria Gonzalez	011-34-00-054	011-34-00-054
American SCUBA Supply	1730 Atlantic Avenue	Lynn Gindoff	213-494-0062	213-494-0068
Aquatic Drama	921 Everglades Way	Gillian Owen	813-442-7854	813-442-7878
Blue Glass Haplopass	6348 W. Shores Lane	Christine Taylor	213-595-1984	213-595-1986
Blue Jack Aqua Center	23-7518 Piddington Lane	Ernest Bennett	401-553-7523	401-553-4403
Blue Sports	255 25th Ave. Box 748	Thomas Kurat	610-772-8104	610-772-8102
Blue Sports Club	62395 Niss Rivas Street	Harry Balthans	612-387-0342	612-387-0348
Caramenish Dive Club	Box 294 Pleasant Point	Nicola Dupont	213-223-0941	213-223-2324
Cayman Divers World Unlimited	PO Box 641	Jos Bailey	011-5-827-044	011-5-827-054
Central Underwater Supplies	PO Box 737	Merle Evantash	27-11-443-245	27-11-443-3259
Davy Jones Locker	246 South 18th Place	Tanya Wagner	803-559-0112	803-559-0553
Diver's Grotto	24891 Universal Lane	Patricia Owen	213-432-0160	213-432-4121
Divers of Blue-green	634 Complex Ave.	Nancy Swan	205-595-1154	205-595-4156
Divers of Costa, Inc.	Memorial Place 94	Charles Lopez	30-881-83394	30-881-05343
Divers of Venice	220 Elm Street	Simone Green	813-443-2395	813-443-0342
Divers-for-Hire	G.O. P Box 61	Jos Hattar	874-804-676	874-804-346
Flamethqas Aquatics	232 955 #12A-77 A.A.	Susan Wong	057-1-773-434	057-1-773-421
Fishermen's Eye	PO Box 7542	Bethan Lewis	800-555-4680	800-555-4939
Frank's Diver Supply	1485 North 48th St.	Lloyd Fellows	803-555-2776	803-555-2769
George Bean & Co.	473 King William Way	Bill Wyman	803-435-2771	803-435-5303
Gosh Coast Supply	223-0 Houston Place	Brian Fale	205-595-2460	205-595-4194
Island Finders	8133 1/2 Stone Avenue	Diamond Ortega	713-423-6776	713-423-6778
Jamaica SCUBA Centre	PO Box 68	Barbara Harvey	011-3-827-043	011-3-827-043
Jamaica Sun, Inc.	PO Box 643	Jonathan White	800-555-2746	800-555-0326
Koral Dive Shoppes	4476 Sugarbowl Hwy	Erica Norman	800-555-0289	800-555-0278
Kirk Enterprises	42 Aqua Lane	Rudolph Claus	713-595-6437	713-595-1073
Larry's Diving School	3262 NW Black Street	Isabelle Nicas	803-433-7777	803-433-0359
Naval SCUBA Club	PO Box 8334	Doris Swan	317-546-0168	317-546-0167
Neriva SCUBA Center	PO Box 6243d Zulu 7831	Stephen Bryant	88-33-88122	88-33-88146
Norwest Divers Club	872 Ocean St.	Joyce Marsh	416-493-0369	416-493-0369
Naptime's Trident Supply	PO Box 120	Louise Francis	778-897-3146	778-897-6143
Norwest's SCUBA Limited	PO Box 6334	Angela Jones	778-123-0746	778-123-0708
Ocean Adventures	PO Box 489 Khul	Paul Sell	778-893-0334	778-893-0333
Ocean Paradise	PO Box 8748	Paul Gardner	800-555-8231	800-555-8460
Ory-Targe SCUBA	7-137 63 Newkirk Road	Brian P Hilpe	416-446-0185	416-446-0223
Princess Island SCUBA	PO Box 31 Wakeport	Anne Marjoch	874-311-025	874-311-253
Professional Divers, Ltd.	4754 Melinda St.	Shirley Nichols	205-555-5333	205-555-4154
Safari Under the Sea	PO Box 7498	Anne Rack	800-403-4233	800-403-3102
San Pablo Dive Center	1701-0 N Broadway	Patricia O'Brien	823-044-2910	823-044-2960
SCUBA Heaven	PO Box O-3374	Robert Michelland	011-32-09-485	011-32-09-485
Shang-Li Sports Center	PO Box D-5468	Frank Parlague	011-32-08-674	011-32-44-038
Sight Diver	1 Neptune Lane	Phyllis Spozner	307-5-876103	307-5-870143
The Dash Champs	6540 Underwater Hwy.	Sam Wilkingson	800-555-3768	800-555-0353
The Diving Company	PO Box 8336	Brian Miller	22-44-00-058	22-44-00-078
Tom Sawyer Diving Centre	622-1 Third Frydenhof	Chris Thomas	594-765-3022	594-765-7772
Tony Tons Tons	PO Box H-4873	Kavin Rider	800-885-0143	800-885-0354
Underwater Fantasy	PO Box 942	Glen Answorth	800-555-2214	800-555-2234

Наш отчет

Необходимо отметить одну особенность многостраничного отчета. Если у второй страницы включить опцию "Печатать на предыдущей странице" (свойство PrintToPreviousPage в инспекторе объектов), то печать объектов второй страницы начнется не с нового листа, а на свободном месте предыдущей страницы. Это позволяет печатать содержимое страниц "встык".

Вложенные отчеты

Иногда нужно в определенном месте основного отчета вывести дополнительные данные, которые могут представлять собой отдельный отчет с довольно сложной структурой. Можно попробовать построить такой отчет с использованием набора бэндов FastReport, но не всегда это удастся. В таком случае можно использовать объект "Вложенный отчет" .

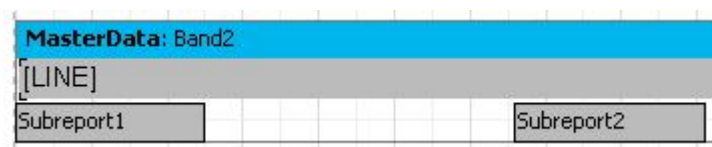
Вставив такой объект в отчет, мы увидим, что FastReport автоматически добавил новую страницу, связанную с этим объектом. Вложенный отчет по своей структуре очень похож на многостраничный. Единственное отличие – вложенный отчет выводится в заданном месте основного отчета, а не после него. При формировании отчета, когда будет встречен объект "Вложенный отчет", вместо него будет выведен отчет, расположенный на связанной странице. После этого формирование основного отчета продолжится.

На страницу вложенного отчета можно также поместить объект "Вложенный отчет", увеличив тем самым уровень вложенности. Пример такого отчета можно найти в демонстрационной программе, отчет под названием "Subreports".

Следует отметить, что способность FastReport строить многократно вложенные отчеты позволяет неограниченно наращивать уровень вложенности данных. Напомним, что без использования объекта "Вложенный отчет" число уровней вложенности в FastReport ограничено – не более шести.

Вывод вложенных отчетов рядом

Вы можете разместить два или более объектов "Вложенный отчет" рядом друг с другом на том же бэнде:



Это позволяет строить отчеты, которые не могут быть построены другим способом – когда в каждом из вложенных отчетов выводится список разной длины:

1	
1	1
2	2
3	3
4	4
5	
6	
2	
1	1

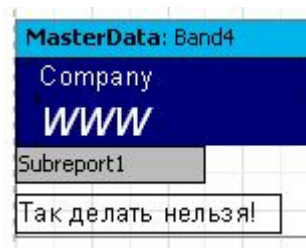
Как видно, FastReport продолжает строить основной отчет с той позиции, на которой закончился вывод наиболее длинного списка.

Ограничения на использование вложенных отчетов

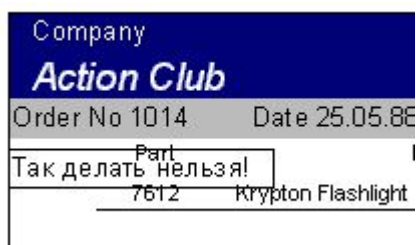
Поскольку вложенный отчет формируется на листе основного отчета, он не может содержать следующих бэндов: "Заголовок/Подвал отчета", "Заголовок/Подвал/Фон страницы", "Заголовок/Подвал колонки". Точнее, положить эти бэнды на лист вложенного отчета можно, но они не будут обработаны (на лист же основного отчета можно класть что угодно). По той же

причине нет смысла менять опции страницы вложенного отчета – при построении используются опции страницы основного отчета.

Нельзя класть объекты ниже объекта "Вложенный отчет":



При выводе вложенного отчета все, что находится ниже, затрется объектами вложенного отчета и может получиться что-то вроде этого:



Чтобы все-таки вывести объекты под вложенным отчетом, используйте уже знакомый нам child-бэнд:

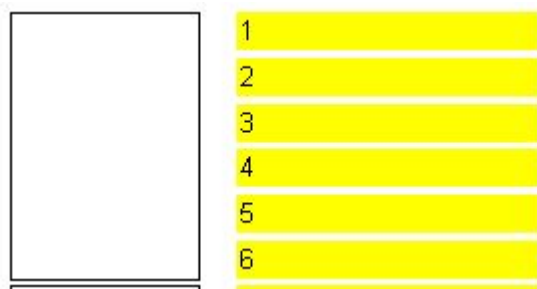


Это же касается случая, когда нужно вывести несколько вложенных отчетов друг под другом.

Опция "Печатать на родителе" (PrintOnParent)

Объект "Вложенный отчет" имеет одно свойство – PrintOnParent, которое может оказаться полезным в некоторых случаях. По умолчанию свойство равно False.

Обычный вложенный отчет выводится в виде отдельных бэндов, которые находятся на странице вложенного отчета. При этом основной бэнд на главном отчете, который содержал объект "Вложенный отчет", не имеет к бэндам вложенного отчета никакого отношения. Если включить опцию "Печатать на родителе", то объекты вложенного отчета будут выводиться на том бэнде, который содержал объект "Вложенный отчет". Это позволяет сделать такой бэнд растягиваемым и вывести рядом с вложенным отчетом растянутый на всю высоту бэнда объект:



Перекрестные (cross-tab) отчеты

Этот вид отчета имеет табличную структуру, т.е. состоит из строк и столбцов, причем заранее неизвестно, сколько строк и столбцов будет содержать таблица. Поэтому отчет растет не только вниз, как уже рассмотренные нами типы отчетов, но и вбок. Типичный пример отчета такого типа – бухгалтерская "шахматка".

Рассмотрим элементы таблицы:

	1	2	3	4
a	a1	a2	a3	a4
b	b1	b2	b3	b4

На рисунке мы видим таблицу с двумя строками и четырьмя столбцами. Здесь a, b – *заголовки строк*, 1, 2, 3, 4 – *заголовки столбцов*, a1..a4, b1..b4 – *ячейки*. Чтобы построить такой отчет, нам понадобится всего один набор данных (запрос или таблица), который имеет три поля и содержит следующие данные:

a	1	a1
a	2	a2
a	3	a3
a	4	a4
b	1	b1
b	2	b2
b	3	b3
b	4	b4

Как видно, первое поле содержит номер строки, второе – номер столбца, третье – содержимое ячейки на пересечении строки и столбца с указанным номером. При построении отчета FastReport создает в памяти таблицу и заполняет ее данными. При этом таблица динамически расширяется, если строки или столбца с заданным номером еще не существует.

Заголовки могут иметь более одного уровня. Рассмотрим следующий пример:

	10		20	
	1	2	1	2
a	a10.1	a10.2	a20.1	a20.2
b	b10.1	b10.2	b20.1	b20.2

В этом примере номер, или *индекс*, столбца – составной, т.е. состоит из двух значений. Этот отчет требует следующих данных:

a	10	1	a10.1
a	10	2	a10.2
a	20	1	a20.1
a	20	2	a20.2
b	10	1	b10.1
b	10	2	b10.2
b	20	1	b20.1
b	20	2	b20.2

Здесь первое поле, как и прежде, содержит индекс строки, второе и третье поля – индекс колонки. Последнее поле содержит значение ячейки. Чтобы вы лучше представляли, как FastReport строит таблицу со сложным заголовком, рассмотрим следующий рисунок:

	10	10	20	20
	1	2	1	2
a	a10.1	a10.2	a20.1	a20.2
b	b10.1	b10.2	b20.1	b20.2

Примерно так выглядит наша таблица перед обработкой. В процессе обработки FastReport объединяет ячейки заголовка с одинаковыми значениями, находящиеся на одном уровне.

Следующий элемент таблицы – промежуточные итоги и итоги, демонстрирует следующий рисунок:

	10			20			Итого
	1	2	Итого	1	2	Итого	
a	a10.1	a10.2	a10.1+a10.2	a20.1	a20.2	a20.1+a20.2	sum(a)
b	b10.1	b10.2	b10.1+b10.2	b20.1	b20.2	b20.1+b20.2	sum(b)
Итого	a10.1+b10.1	a10.2+b10.2	a10.1+b10.1+a10.2+b10.2	a20.1+b20.1	a20.2+b20.2	a20.1+b20.1+a20.2+b20.2	sum(a)+sum(b)

Этот отчет строится на тех же данных, что и предыдущий. Столбцы, показанные серым на рисунке, вычисляются автоматически и не входят в исходный набор данных.

Строим кросс-отчет

Перейдем от теории к практике. Построим простой кросс-отчет, показывающий зарплату сотрудников за четыре года. Для этого нам понадобится таблица `simplecross.db`, которая находится в папке `FastReport DEMOS\CROSS\DATA`. Таблица содержит данные следующего характера:

Name	Year	Salary
Ann	1999	3300
Ben	2002	2000
....		

Как обычно, создаем новый проект в Delphi, кладем на форму компоненты TTable, TfrxDBDataSet, TfrxReport и настраиваем их:

Table1:


```
DatabaseName = 'c:\Program Files\FastReport3\Demos\Cross\Data'  
TableName = 'simplecross.db'
```

естественно, значение свойства DatabaseName должно соответствовать пути к вашей папке с FastReport!

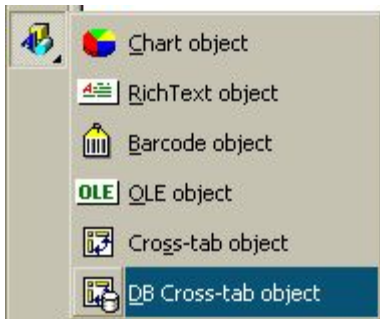
frxDBDataSet1:

```
DataSet = Table1  
UserName = 'SimpleCross'
```

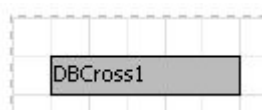
Для построения кросс-отчетов необходимо использовать компонент

TfrxCrossObject  из палитры компонент FastReport. Просто положите его на форму – ничего настраивать не требуется. При этом в список "uses" вашего проекта добавится модуль frxCross – он содержит всю необходимую функциональность.

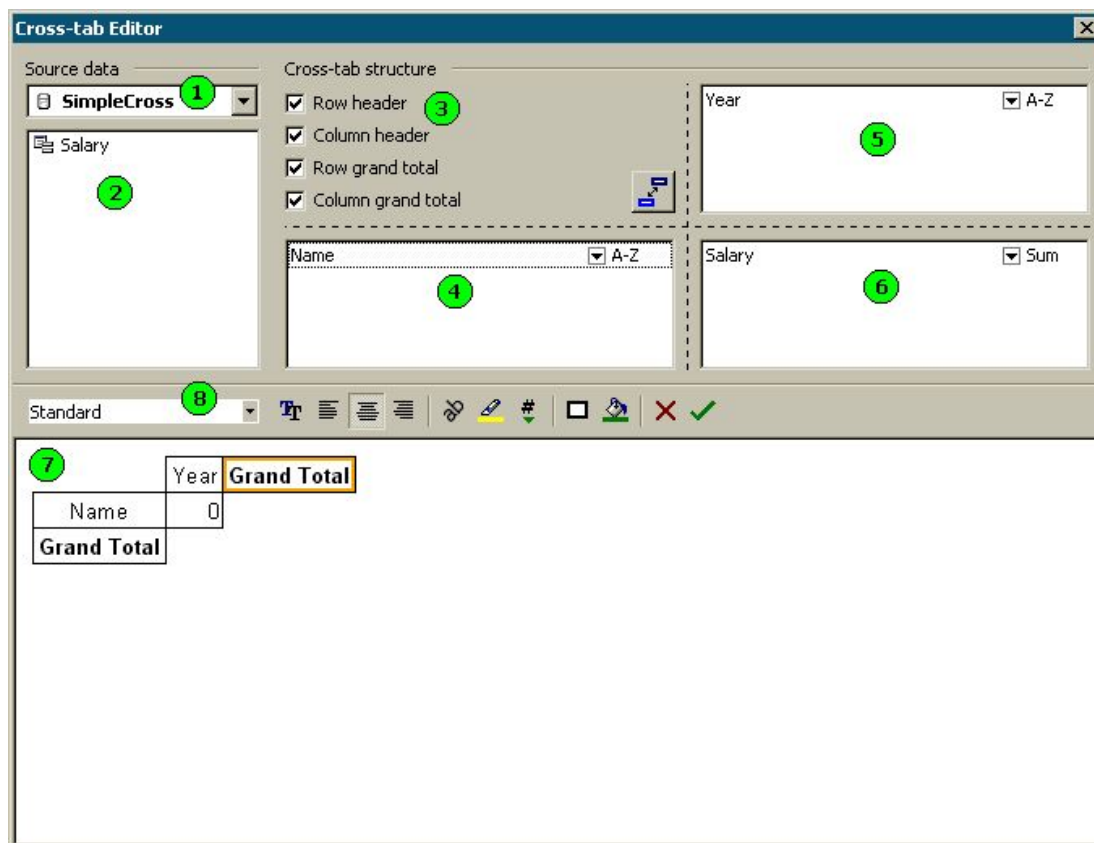
Зайдем в дизайнер отчета. Первым делом подключим наш источник данных в меню "Отчет|Данные...". На лист отчета положим объект "Кросс-таблица БД":



На листе дизайнера объект выглядит скромно:




Все настройки делаются с помощью редактора объекта. Вызовем его, сделав двойной щелчок мышью на объекте:



Цифрами помечены:

- 1 – выпадающий список доступных источников данных;
- 2 – список полей в выбранном источнике данных. Поля из этого списка можно перетаскивать в списки 4, 5, 6;
- 3 – здесь можно определить, показывать ли заголовки и итоги;
- 4 – список полей, которые образуют заголовок строки;
- 5 – список полей, которые образуют заголовок столбца;
- 6 – список полей, которые образуют ячейку таблицы;
- 7 – здесь отображается структура будущей таблицы. На элементы можно кликать мышкой;
- 8 – панель инструментов для изменения оформления таблицы:

Standard - выбор стиля таблицы;

 - параметры шрифта ячейки;



- выравнивание текста;



- поворот текста;



- условное выделение;




- формат ячейки;



- рамка и заливка ячейки.


Как видно, действовать здесь придется только мышью. В нашем случае достаточно перетащить поля из списка 2 в списки 4, 5, 6, как показано на рисунке.

Пока больше делать ничего не будем - закроем редактор кнопкой ОК . Если сейчас запустить отчет, мы увидим следующее:

	1999	2000	2001	2002	Grand Total
Ann	3300	2700	3100	1700	10800
Ben	4300	2400		2000	8700
Catherine	6100	3200			9300
Den		3999	8100		12099
Grand Total	13700	12299	11200	3700	40899


Что ж, именно то, что мы хотели. Продолжим изучение объекта, вновь вызовем его редактор. Первое, что нам захочется сделать – это сменить цвета заголовков и вместо "Grand Total" вывести "Итого". Сделать это очень просто, используя нижнее поле редактора (N7 на рисунке). Здесь отображается структура кросс-таблицы, и ее также можно настроить с помощью мыши. Активная ячейка отображается с обрамлением оранжевого цвета:

	Year	Grand Total
Name	0	
Grand Total		

Чтобы сменить цвет заголовка на серый, последовательно щелкните на объектах Year, Name, Grand Total и выберите нужный цвет кнопкой  на панели инструментов. Чтобы сменить надпись "Grand Total", дважды щелкните на ячейке – вы увидите знакомый редактор текста, в котором наберите "Итого". После этого наш отчет будет выглядеть так:

	1999	2000	2001	2002	Итого
Ann	3300	2700	3100	1700	10800
Ben	4300	2400		2000	8700
Catherine	6100	3200			9300
Den		3999	8100		12099
Итого	13700	12299	11200	3700	40899

Осталось задать формат, в котором выводятся денежные значения. Для этого в редакторе кросс-объекта последовательно щелкните на объектах "Итого" и объекте, представляющем ячейку (с текстом "0") и выберите нужный формат,

нажав кнопку  на панели инструментов. Получается вот что:

	1999	2000	2001	2002	Итого
Ann	3 300,00р.	2 700,00р.	3 100,00р.	1 700,00р.	10 800,00р.
Ben	4 300,00р.	2 400,00р.		2 000,00р.	8 700,00р.
Catherine	6 100,00р.	3 200,00р.			9 300,00р.
Den		3 999,00р.	8 100,00р.		12 099,00р.
Итого	13 700,00р.	12 299,00р.	11 200,00р.	3 700,00р.	40 899,00р.

Хотите сказать, что зарплата ваших сотрудников измеряется в долларах? ;) Это легко поправить, указав другую строку форматирования: \$%2.2n (для всех трех объектов – ведь форматирование задается отдельно для ячеек таблицы и строки/столбца итогов).

Использование функций

В нашем примере мы вывели в строке "Итого" сумму зарплат каждого сотрудника за четыре года. Помимо суммы, можно использовать следующие функции:

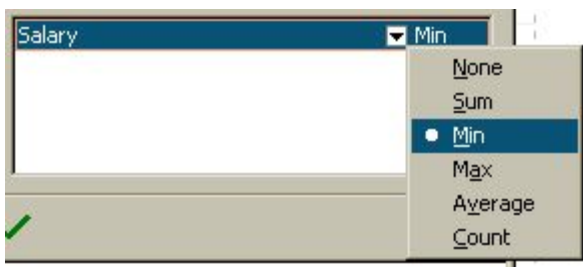
MIN – минимальное значение

MAX – максимальное значение

AVG – среднее значение

COUNT – количество значений

Давайте попробуем использовать функцию MIN в нашем примере. Для этого откройте редактор кросс-объекта, и щелкните мышкой на поле "Salary" в районе значка со стрелкой вниз.



Выберите из меню функцию MIN. Теперь можно изменить текст в ячейке итогов с "Итого" на "Минимум". Готовый отчет будет выглядеть так:

	1999	2000	2001	2002	Минимум
Ann	3300	2700	3100	1700	1700
Ben	4300	2400		2000	2000
Catherine	6100	3200			3200
Den		3999	8100		3999
Минимум	3300	2400	3100	1700	1700

Сортировка значений

По умолчанию значения строк и столбцов сортируются по возрастанию. Причем, если значения имеют численный тип, они сортируются по величине, а если строковый – в алфавитном порядке. Мы можем задать свой режим сортировки для каждого значения строки и столбца отдельно. Доступны следующие режимы: сортировка по возрастанию, по убыванию или отсутствие сортировки. В последнем случае значения в строках/колонках будут отображаться в порядке их поступления.

Поменяем сортировку колонок в нашем примере: пусть года идут в порядке убывания. Для этого зайдём в редактор кросс-объекта и выберем элемент колонки "Year". Чтобы сменить сортировку, щелкнем в районе значка со стрелкой вниз:



После этого закроем редактор и запустим отчет. Он будет выглядеть следующим образом:

	2002	2001	2000	1999	Итого
Ann	1700	3100	2700	3300	10800
Ben	2000		2400	4300	8700
Catherine			3200	6100	9300
Den		8100	3999		12099
Итого	3700	11200	12299	13700	40899

Таблица с составными заголовками

Наш предыдущий пример имел по одному значению в заголовках строки и столбца. Рассмотрим на практике построение таблицы, у которой заголовок составной, т.е. состоит из двух и более значений. Для этого нам понадобится таблица cross.db, которая находится в папке FastReport DEMOS\CROSS\DATA. Таблица содержит данные следующего характера:

Name	Year	Month	Days	Salary
Ann	1999	2	3	1000
Ben	2002	1	5	2000
....				

Как видно, данные этой таблицы похожи на таблицу simplecross.db, которой мы пользовались в предыдущем примере. Добавились два поля – Month и Days, которые содержат номер месяца и количество проработанных дней в этом месяце, соответственно. На основе этих данных уже можно построить несколько отчетов, например, зарплата сотрудников за все года с разбивкой по месяцам.

Создадим новый проект в Delphi, кладем на форму компоненты TTable, TfrxDBDataSet, TfrxReport и настраиваем их:

Table1:

```
DatabaseName = 'c:\Program Files\FastReport3\Demos\Cross\Data'
TableName = 'cross.db'
```

естественно, значение свойства DatabaseName должно соответствовать пути к вашей папке с FastReport!

frxDBDataSet1:

```
DataSet = Table1
UserName = 'Cross'
```

В дизайнера отчета делаем уже привычные вещи – подключаем источник данных в окне "Отчет|Данные..." и кладем на лист отчета объект "Кросс-таблица БД". Чтобы настроить кросс-объект, запустим его редактор двойным щелчком мыши на объекте.

Какого вида отчет мы хотим получить? Он должен быть похож на отчет из предыдущего примера, но с разбивкой годов на месяцы. Следовательно, настроить кросс-объект надо таким же образом, только добавив в заголовок столбца поле "Month":

При этом в нижней части редактора отобразится структура будущего отчета:

	Year		Grand Total
	Month	Total	
Name			
Grand Total			

При желании можно поменять цвета и заменить английские "Grand total" и "Total" русским "Итого". У нас получился следующий отчет:

	1999					2000				2001				2002		Итого
	2	10	11	12	Итого	1	2	3	Итого	1	2	3	Итого	1	Итого	
Ann	1000		1100	1200	3300	1300	1400		2700		1500	1600	3100	1700	1700	10800
Ben		2100	2200		4300		2400		2400				0	2000	2000	8700
Catherine		3000	3100		6100			3200	3200				0		0	9300
Den					0	3999			3999	4000	4100		8100		0	12099
Итого	1000	5100	6400	1200	13700	5299	3800	3200	12299	4000	5600	1600	11200	3700	3700	40899

Обратите внимание, что FastReport автоматически добавил колонку промежуточных итогов, которые выводятся после каждого года. Эта опция настраивается в редакторе кросс-объекта: достаточно выделить элемент колонки "Year" и выключить флажок "Итого":

Также можно заметить, что промежуточный итог отсутствует у самого нижнего элемента столбца (также в том случае, если этот элемент единственный) – действительно, промежуточные итоги после каждого месяца (в нашем примере) ни к чему.

Рассмотрим еще один момент, относящийся к промежуточным итогам. В нашем примере хотелось бы вместо надписи "Итого" вывести "Итого за 2000г.". Сделать это очень просто: зайдите в редактор кросс-объекта, выделите нужный объект в нижней части редактора и впишите в него следующий текст:

Итого за [Value]

В процессе построения выражение "Value" будет заменено на значение из заголовка таблицы, лежащего выше:

	1999					2000			
	2	10	11	12	Итого за 1999	1	2	3	Итого за 2000
Ann	1000		1100	1200	3300	1300	1400		2700
Ben		2100	2200		4300		2400		2400
Catherine		3000	3100		6100			3200	3200
Den					0	3999			3999
Итого	1000	5100	6400	1200	13700	5299	3800	3200	12299

Подбор ширины ячеек

На предыдущем рисунке видно, что FastReport автоматически подбирает ширину ячеек таким образом, чтобы уместились самые длинные строки. В некоторых случаях это нежелательно – при очень длинных строках таблица будет смотреться некрасиво. Что можно сделать в нашем случае? Самый простой вариант – вставить разрыв строки в текст объекта с промежуточными итогами, т.е. поместить в него строку:

*Итого
за [Value]*

Мы увидим, что теперь таблица выглядит гораздо лучше:

	1999				
	2	10	11	12	Итого за 1999
Ann	1000		1100	1200	3300
Ben		2100	2200		4300
Catherine		3000	3100		6100
Den					0
Итого	1000	5100	6400	1200	13700

Однако такой способ можно использовать далеко не всегда – что, если сами значения строк/столбцов достаточно длинные, ведь их нельзя исправить вставкой разрыва строки вручную. Для этого кросс-объект имеет свойства MinWidth и MaxWidth (минимальная и максимальная ширина ячейки соответственно). Оба этих свойства доступны только через инспектор объектов.

По умолчанию значение MinWidth = 0, MaxWidth = 200. Этого достаточно для большинства случаев. Вы можете установить свои значения, если к оформлению таблицы предъявляются особенные требования.

Так, в нашем примере можно задать MinWidth = MaxWidth = 50. Это означает, что ширина ячейки таблицы должна быть в любом случае равной 50 пикселям. Если ячейка меньше, она "дотягивается" до значения MinWidth, если больше – ее ширина фиксируется на уровне MaxWidth, а текст в ячейке переносится по словам. На нашем примере это выглядит так:


	1999				
	2	10	11	12	Итого за 1999
Ann	1000		1100	1200	3300
Ben		2100	2200		4300
Catherine		3000	3100		6100
Den					0
Итого	1000	5100	6400	1200	13700

Выделение значений цветом

Часто бывает необходимо выделить какие-либо значения другим цветом шрифта или фона. Мы уже рассматривали подобную задачу на примере отчета с группами. Тогда мы использовали условное выделение для объекта "Текст", которое нам пригодится и сейчас.

Рассмотрим выделение на нашем примере. Допустим, мы захотим выделить значения больше 3000 красным цветом шрифта. Для этого зайдём в редактор кросс-объекта и щелкнем на объекте, который представляет ячейку таблицы, в нижней части окна редактора:



Чтобы задать параметры выделения, нажмите кнопку  на панели инструментов. Откроется уже знакомое окно редактора выделения, в котором надо задать следующее условие:

Value > 3000

Это все, что необходимо. Закроем редактор кнопкой ОК и запустим наш отчет:

	1999					2000				2001				2002		Итого
	2	10	11	12	Итого за 1999	1	2	3	Итого за 2000	1	2	3	Итого за 2001	1	Итого за 2002	
Ann	1000		1100	1200	3300	1300	1400		2700		1500	1600	3100	1700	1700	10800
Ben		2100	2200		4300		2400		2400				0	2000	2000	8700
Catherine		3000	3100		6100			3200	3200				0		0	9300
Den					0	3999			3999	4000	4100		8100		0	12099
Итого	1000	5100	6400	1200	13700	5299	3800	3200	12299	4000	5600	1600	11200	3700	3700	40899

При необходимости таким же образом можно задать выделение для итоговых значений, для значений столбцов/строк.

Управление кросс-таблицей из скрипта

Если визуальных средств настройки таблицы недостаточно, можно использовать скрипт для тонкой настройки внешнего вида таблицы. Объект "Кросс-таблица" имеет следующие события:

Событие	Описание
OnAfterPrint	Событие вызывается после печати таблицы.
OnBeforePrint	Событие вызывается перед печатью таблицы.

OnCalcHeight	Событие вызывается перед подсчетом высоты строки таблицы. Обработчик события может вернуть нужное значение высоты или 0 для того, чтобы скрыть строку.
OnCalcWidth	Событие вызывается перед подсчетом ширины столбца таблицы. Обработчик события может вернуть нужное значение ширины или 0 для того, чтобы скрыть столбец.
OnPrintCell	Событие вызывается перед отображением ячейки таблицы. Обработчик события может изменить оформление или содержимое ячейки.
OnPrintColumnHeader	Событие вызывается перед отображением заголовка колонок таблицы. Обработчик события может изменить оформление или содержимое ячейки заголовка.
OnPrintRowHeader	Событие вызывается перед отображением заголовка строк таблицы. Обработчик события может изменить оформление или содержимое ячейки заголовка.

В событиях удобно использовать следующие методы объекта "Кросс-таблица":

Метод	Описание
function ColCount: Integer	Возвращает количество колонок в таблице.
function RowCount: Integer	Возвращает количество строк в таблице.
function IsGrandTotalColumn (Index: Integer): Boolean	Возвращает True, если колонка с указанным номером является итоговой.
function IsGrandTotalRow (Index: Integer): Boolean	Возвращает True, если строка с указанным номером является итоговой.
function IsTotalColumn (Index: Integer): Boolean	Возвращает True, если колонка с указанным номером является колонкой промежуточных итогов.
function IsTotalRow (Index: Integer): Boolean	Возвращает True, если строка с указанным номером является строкой промежуточных итогов.
procedure AddValue(const Rows, Columns, Cells: array of Variant)	Добавляет значение в таблицу.

Рассмотрим на примере, каким образом можно выделить третью колонку цветом фона (в нашем примере – это данные за ноябрь 1999 года). Для этого выделим кросс-таблицу и создадим обработчик события OnPrintCell:

```
procedure Cross1OnPrintCell (Memo: TfrxMemoView;
 RowIndex, ColumnIndex, CellIndex: Integer;
  RowValues, ColumnValues, Value: Variant);
begin
```

```

if ColumnIndex = 2 then
    Memo.Color := clRed;
end;

```

Мы увидим следующий результат:

1999					
	2	10	11	12	Total
Ann	1000		1100	1200	3300
Ben		2100	2200		4300
Catherine		3000	3100		6100
Den					0
Grand Total	1000	5100	6400	1200	13700

Чтобы выделить цветом заголовок колонки, создадим обработчик события OnPrintColumnHeader:

```

procedure Cross1OnPrintColumnHeader(Memo: TfrxMemoView;
    HeaderIndexes, HeaderValues, Value: Variant);
begin
    if (VarToStr(HeaderValues[0]) = '1999') and
        (VarToStr(HeaderValues[1]) = '11') then
        Memo.Color := clRed;
end;

```

Результат:

1999					
	2	10	11	12	Total
Ann	1000		1100	1200	3300
Ben		2100	2200		4300
Catherine		3000	3100		6100
Den					0
Grand Total	1000	5100	6400	1200	13700

Поясним работу скриптов. Обработчик события OnPrintCell вызывается перед печатью ячейки, которая содержится в теле таблицы (при печати ячеек из заголовка таблицы вызывается обработчик OnPrintColumnHeader или OnPrintRowHeader). При этом в обработчик OnPrintCell передается ссылка на объект "Текст", который представляет собой ячейку таблицы (параметр Memo), и "адрес" ячейки в двух вариантах: номер строки, колонки и ячейки (последнее актуально, если в вашей таблице многоуровневые ячейки) в параметрах RowIndex, ColumnIndex, CellIndex соответственно. Второй вариант "адреса" – это параметры RowValues и ColumnValues. Параметр Value – это содержимое ячейки.

Для определения "адреса" вы можете использовать как первый вариант (RowIndex, ColumnIndex), так и второй (RowValues, ColumnValues) – что удобнее в конкретном случае. В нашем случае нужно было выделить третью колонку – поэтому удобнее анализировать первый вариант. Т.к. нумерация колонок и строк начинается с 0, проверка ColumnIndex = 2 позволила нам определить 3-ю колонку. Можно было поступить иначе, анализируя нужную колонку по ее данным (нам нужен 11 месяц 1999 года):

```
procedure Cross1OnPrintCell(Memo: TfrxMemoView;
  RowIndex, ColumnIndex, CellIndex: Integer;
  RowValues, ColumnValues, Value: Variant);
begin
  if (VarToStr(ColumnValues[0]) = '1999') and
    (VarToStr(ColumnValues[1]) = '11') then
    Memo.Color := clRed;
end;
```

Значения, передаваемые в параметрах RowValues и ColumnValues – это массивы типа Variant с нулевой базой. Нулевой элемент – это значение верхнего уровня заголовка таблицы, первый – значение следующего уровня и т.д. В нашем случае ColumnValues[0] – это года, ColumnValues[1] – месяцы.

Зачем нужно преобразование VarToStr? Это гарантирует отсутствие ошибок приведения типов. Delphi при операциях с типом Variant пытается автоматически приводить строки в числовой формат, что в нашем случае вызовет ошибку при попытке приведения значения столбцов 'Total' и 'Grand Total'.

Обработчик события OnPrintColumnHeader вызывается при печати ячеек заголовка столбца. Набор параметров похож на параметры обработчика OnPrintCell, но здесь "адрес" ячейки (параметры HeaderIndexes, HeaderValues) передается иначе. Параметр HeaderValues возвращает те же значения, что и параметры ColumnValues, RowValues в обработчике OnPrintCell. Параметр HeaderIndexes также является массивом значений типа Variant и содержит адрес ячейки заголовка в другой форме: нулевой элемент – это порядковый номер верхнего уровня заголовка таблицы, первый – номер следующего уровня и т.д. Принцип нумерации ячеек заголовка станет понятен, если взглянуть на рисунок:

	0					1				2			
	0	1	2	3	4	0	1	2	3	0	1	2	3
0	1000		1100	1200	3300	1300	1400		2700		1500	1600	3100
1		2100	2200		4300		2400		2400				0
2		3000	3100		6100			3200	3200				0
3					0	3999			3999	4000	4100		8100
4	1000	5100	6400	1200	13700	5299	3800	3200	12299	4000	5600	1600	11200

В нашем случае удобно анализировать значение HeaderValues, но можно написать и такой обработчик:

```

procedure Cross1OnPrintColumnHeader(Memo: TfrxMemoView;
  HeaderIndexes, HeaderValues, Value: Variant);
begin
  if (HeaderIndexes[0] = 0) and (HeaderIndexes[1] = 2) then
    Memo.Color := clRed;
end;

```

Управление размером строк/колонок

С помощью обработчиков событий OnCalcWidth, OnCalcHeight можно управлять шириной и высотой строк и столбцов таблицы. Покажем на примере, как увеличить ширину колонки, соответствующей 11 месяцу 1999 года. Для этого создадим обработчик события OnCalcWidth:

```

procedure Cross1OnCalcWidth(ColumnIndex: Integer;
  ColumnValues: Variant; var Width: Extended);
begin
  if (VarToStr(ColumnValues[0]) = '1999') and
    (VarToStr(ColumnValues[1]) = '11') then
    Width := 100;
end;

```

Результат:

	1999				
	2	10	11	12	Total
Ann	1000		1100	1200	3300
Ben		2100	2200		4300
Catherine		3000	3100		6100
Den					0
Grand Total	1000	5100	6400	1200	13700

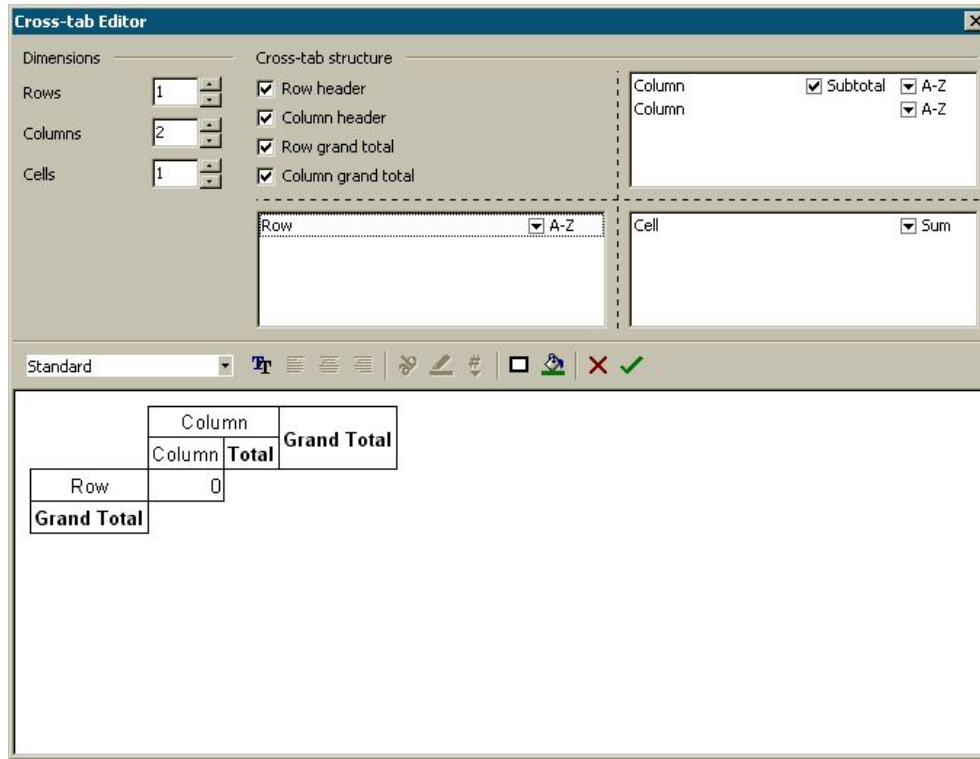
Чтобы скрыть колонку, в нашем примере достаточно вернуть Width := 0. Заметим, что при этом суммы пересчитываться не будут – матрица к этому моменту уже заполнена значениями.

Заполнение таблицы вручную

Как мы уже знаем, есть две разновидности кросс-таблицы: объекты "Кросс-таблица БД" и "Кросс-таблица". Все это время мы работали с первым объектом, который привязывается к данным из таблицы БД и автоматически заполняет себя при запуске отчета. Рассмотрим второй объект – "Кросс-таблица".

Этот объект не привязан к данным из БД. Вы должны сами позаботиться о заполнении таблицы данными. У этого объекта похожий редактор, только здесь

вместо полей БД надо выбрать количество измерений в заголовках таблицы и в ее ячейках:



Рассмотрим работу с объектом "Кросс-таблица" на примере. Положим на лист отчета объект и настроим его свойства так, как показано на предыдущем рисунке: количество уровней в заголовке строк – 1, в заголовке колонок – 2, в ячейке – 1. Чтобы заполнить таблицу данными, воспользуемся обработчиком события OnBeforePrint объекта:

```
procedure Cross1OnBeforePrint(Sender: TfrxComponent);
begin
    with Cross1 do
    begin
        AddValue(['Ann'], [2001, 2], [1500]);
        AddValue(['Ann'], [2001, 3], [1600]);
        AddValue(['Ann'], [2002, 1], [1700]);

        AddValue(['Ben'], [2002, 1], [2000]);

        AddValue(['Den'], [2001, 1], [4000]);
        AddValue(['Den'], [2001, 2], [4100]);
    end;
end;
```

В обработчике необходимо добавить нужные данные в таблицу с помощью метода TfrxCrossView.AddValue. Этот метод имеет три параметра, каждый из которых является массивом значений типа Variant. Первый параметр – это значения строки, второй – значения столбца, третий – значения ячеек. Заметьте – количество


значений в каждом массиве должно соответствовать настройке объекта! В нашем случае объект имеет один уровень в заголовке строк, два уровня в заголовке колонок и один уровень ячеек – соответственно, мы передаем в AddValue одно значение для строк, два значения для столбцов и одно значение для ячеек.

Запустив отчет на выполнение, мы увидим следующее:

	2001				2002		Grand Total
	1	2	3	Total	1	Total	
Ann		1500	1600	3100	1700	1700	4800
Ben				0	2000	2000	2000
Den	4000	4100		8100		0	8100
Grand Total	4000	5600	1600	11200	3700	3700	14900

Метод AddValue можно точно так же использовать для объекта "Кросс-таблица БД". Это позволяет добавлять в кросс-таблицу данные, которых нет в источнике данных, привязанном к объекту. Либо, если такие данные есть, они суммируются с данными из таблицы.

Построение диаграмм

FastReport позволяет вставлять в отчет диаграммы. Для этого используется компонент TfrxChartObject  из палитры компонент FastReport. Компонент основан на библиотеке TeeChart, которая поставляется в комплекте с Delphi. Так же можно использовать библиотеку TeeChartPro, которая приобретается отдельно.

Рассмотрим построение простой диаграммы на примере. Для этого нам понадобится таблица country.db из комплекта демонстрационных баз данных DBDEMOS. Таблица содержит данные о странах, их площади и населении:

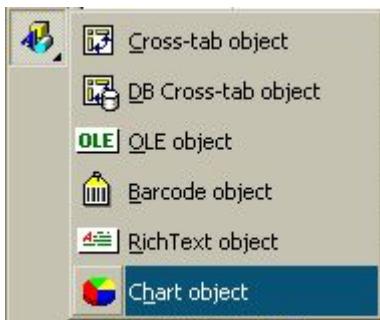
Name	Area	Population
Argentina	2 777 815	32 300 003
Bolivia	1 098 575	7 300 000
....		

Создадим новый проект в Delphi. Положим на форму компоненты TTable, TfrxDBDataSet, TfrxReport и настроим их:

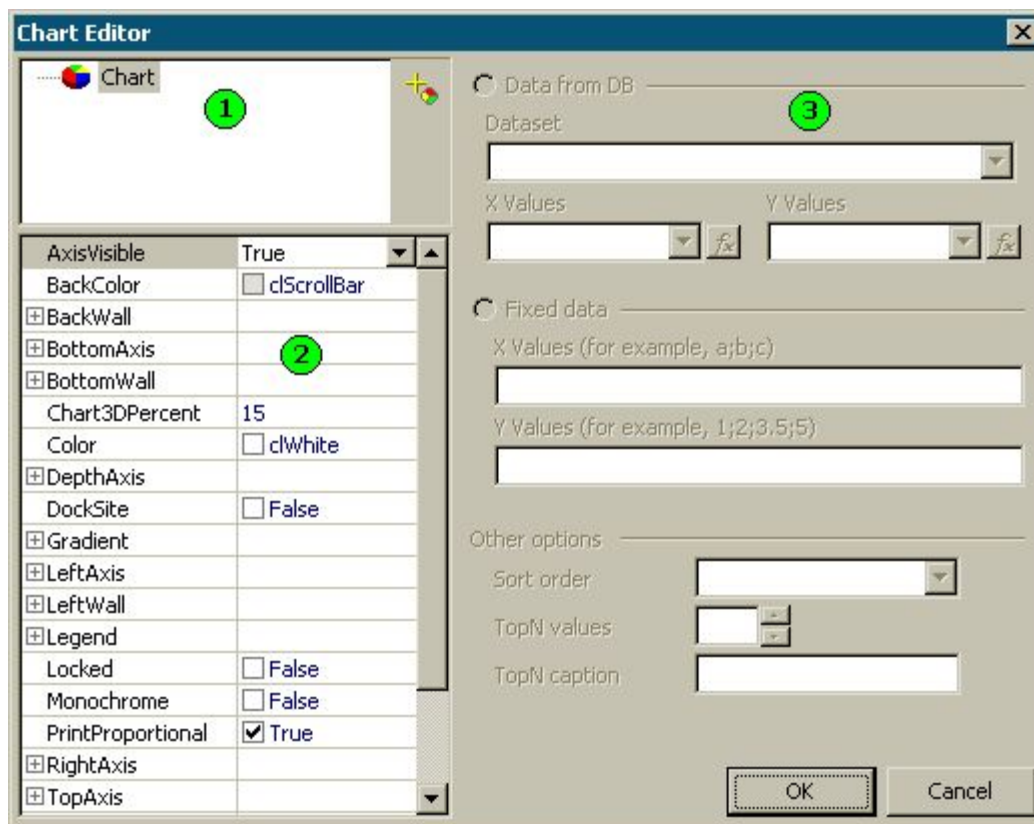
```
Table1:  
DatabaseName = 'DBDEMOS'  
TableName = 'country.db'
```

```
frxDBDataSet1:  
DataSet = Table1  
UserName = 'Country'
```

Зайдем в дизайнер отчета и подключим источник данных в окне "Отчет|Данные...". Положим на лист отчета объект "Диаграмма":




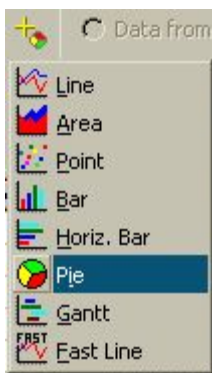
Установим размеры объекта – 18х8см. Чтобы настроить объект, вызовем его редактор двойным щелчком мыши.



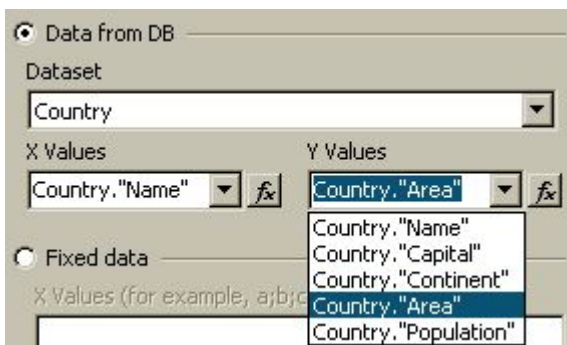
Цифрами на рисунке обозначены:

- 1 – структура диаграммы. Диаграмма может содержать одну или несколько серий (series).
- 2 – инспектор объектов, который отображает свойства выбранного в окне 1 элемента. Таким образом можно произвести тонкую настройку свойств диаграммы.
- 3 – панель привязки серии к данным, становится активной при выборе серии в окне 1.

При первом запуске окно редактора будет иметь вид, показанный на рисунке. Первое, что необходимо сделать – добавить одну или несколько серий (в нашем примере – одну). Для этого нажмите кнопку  и выберите из выпадающего списка круговую диаграмму:

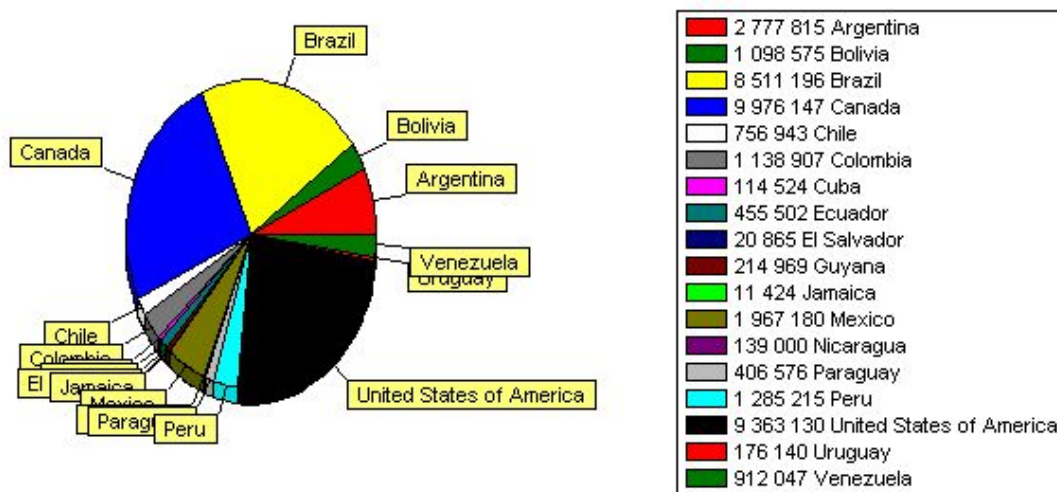


Как видим, доступно восемь разных типов серий. После добавления серии панель 3 стала активной. Здесь надо указать, какие данные будут использоваться при построении диаграммы. Сначала выберем набор данных из выпадающего списка "Набор данных". Поля "X значения" и "Y значения" заполним следующим образом – их также можно выбрать из выпадающих списков:

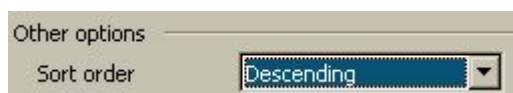


Здесь необходимо пояснить, что значения по оси X могут быть любого типа (например, строкового) – они несут информационный характер. Значения же по оси Y должны быть строго числового типа. В нашем случае (с круговой диаграммой) значения по оси X используются для отображения поясняющих надписей, а для построения диаграммы используются только значения по оси Y.

Пока закончим настройку, закрыв редактор кнопкой ОК. Запустим отчет на построение:



Что можно улучшить в этом отчете? Во-первых, неплохо бы отсортировать значения по убыванию. Снова заходим в редактор диаграммы и выбираем серию в верхней части окна. Теперь выбираем нужный режим сортировки:

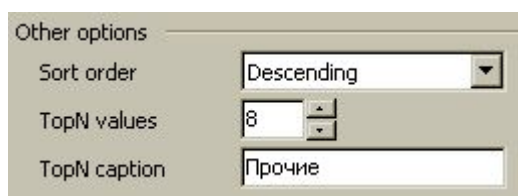


Если теперь запустить отчет, мы увидим, что данные в поясняющей таблице отсортированы.

Ограничение количества значений в диаграмме

Наша диаграмма выглядит довольно перегруженной – слишком много мелких значений, которые все равно не видны на диаграмме. FastReport позволяет ограничить количество значений в диаграмме определенным значением. При этом все значения, которые не уложились в заданный предел, выведутся в виде одного значения, которое представляет собой сумму не уместившихся значений.

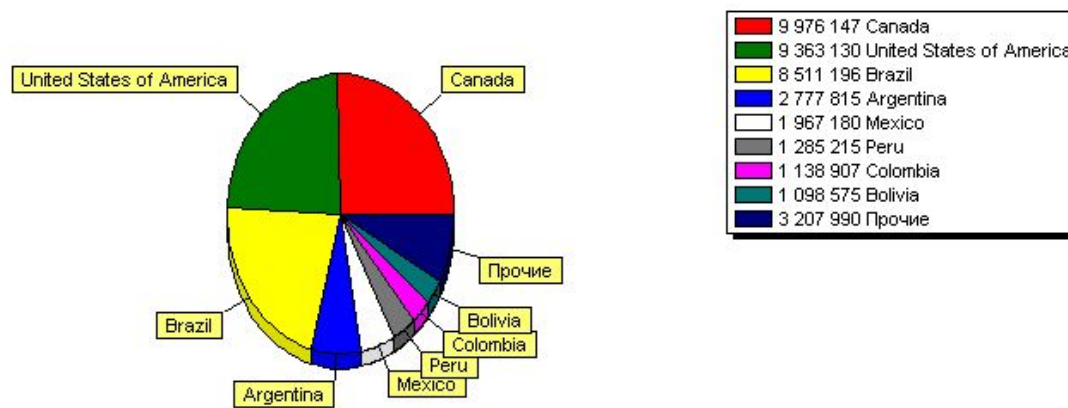
В нашем примере диаграмма имеет 18 значений, можно вывести только 8 из них. Зайдем в редактор и настроим ограничение:



Ограничение будет работать, если поле "TopN" не равно нулю. В поле "TopN заголовок" надо указать название, которое выведется напротив суммарного

значения. Режим сортировки значения не имеет – значения будут отсортированы по убыванию.

В результате отчет будет выглядеть таким образом:



Некоторые полезные настройки

Рассмотрим некоторые настройки, которые могут пригодиться для задания внешнего вида диаграммы. Эти настройки можно сделать только в инспекторе объектов.

Следующие основные свойства доступны при выборе диаграммы в списке сверху:



Gradient – настройки градиентной заливки фона. Для отображения градиента включите свойство `Gradient.Visible`.

Legend – настройки внешнего вида поясняющей таблицы. Таблицу можно отключить с помощью свойства `Legend.Visible`. Положение таблицы настраивается с помощью свойства `Legend.Alignment`.

При выборе серии доступны свойства:

ColorEachPoint – раскрашивать каждое значение разным цветом.

ExplodeBiggest – выделять наибольшее значение (только для серии типа "круговая диаграмма").

Marks – настройки внешнего вида поясняющих подсказок.

ValueFormat – строка форматирования значений.

Диаграмма с фиксированными данными

В предыдущем примере мы строили диаграмму на основе данных из таблицы БД. Есть еще один способ построить диаграмму – ввести необходимые данные вручную. Этот способ удобно использовать для построения небольших диаграмм.

Покажем, как это делается, на небольшом примере. Положим на лист отчета диаграмму и зайдем в ее редактор. Добавим серию типа "Столбчатая диаграмма"



и настроим ее свойства:

☒ Fixed data

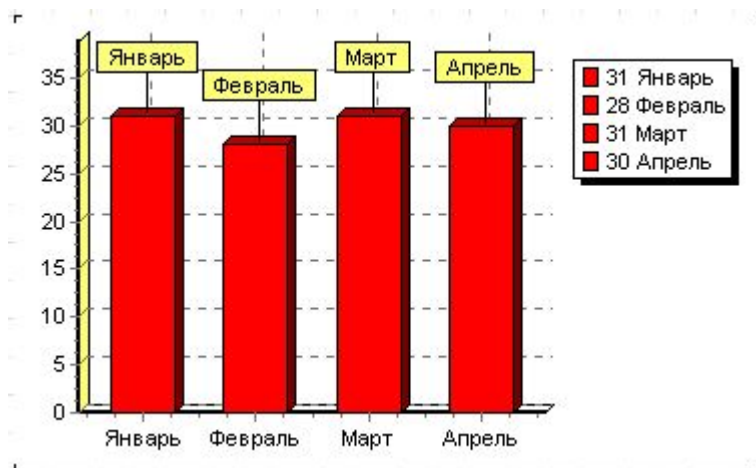
X Values (for example, a;b;c)

Январь;Февраль;Март;Апрель

Y Values (for example, 1;2;3.5;5)

31;28;31;30

Результат можно видеть даже в дизайнера, не запуская отчета:



Скрипт

Скрипт – это программа на языке высокого уровня, которая является частью отчета. При запуске отчета на выполнение также запускается и скрипт. Скрипт позволяет выполнить обработку данных, которую невозможно сделать штатными средствами ядра FastReport, например, скрыть ненужные данные в зависимости от какого-либо условия. Скрипт также используется для управления диалоговыми формами, входящими в состав отчета.

Скрипт может быть написан на одном из языков, входящих в состав скриптового движка, FastScript. На сегодняшний день поддерживаются следующие языки:

- PascalScript
- C++Script
- BasicScript
- JScript

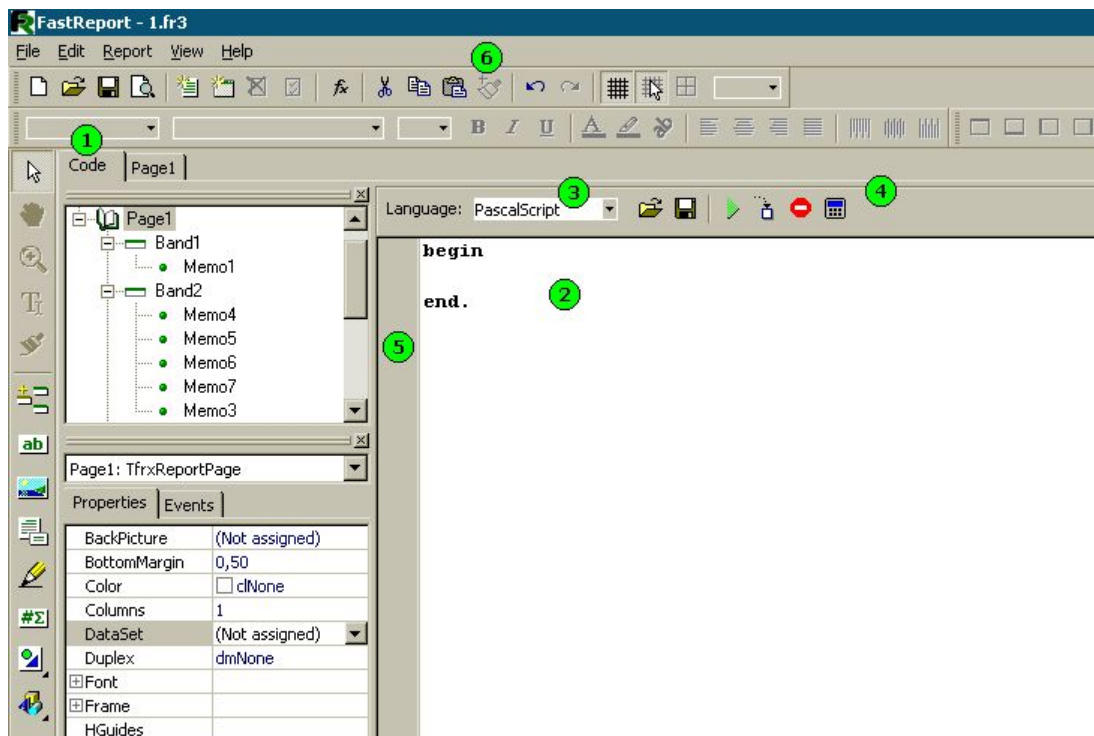
Возможности скриптового движка FastScript следующие:

- стандартный языковой набор: переменные, константы, процедуры, функции (с возможностью вложенности) с переменными/постоянными/умалчиваемыми параметрами, все стандартные операторы (включая case, try/finally/except, with), типы (целый, дробный, логический, символьный, строковый, многомерные массивы, множество, variant), классы (с методами, событиями, свойствами, индексами и свойствами по умолчанию);
- отсутствуют объявления типов (records, classes) в скрипте; нет записей (records), указателей (pointers), множеств (sets) (однако возможно использование оператора 'IN' - "a in ['a'..'c','d']"), нет типа shortstring, нет безусловного перехода (GOTO);
- проверка совместимости типов;
- доступ к любому объекту отчета.

Вы можете создавать скрипты в дизайнера FastReport, который содержит редактор скриптов с подсветкой синтаксиса. Также есть встроенный отладчик, имеющий следующие функции: Step, Breakpoint, Run to cursor, Evaluate.

Первое знакомство

Средства для работы со скриптом находятся на закладке "Код" дизайнера FastReport. Так выглядит экран дизайнера при переключении на эту закладку:



Цифрами на рисунке отмечены:

- 1 – закладка "Код";
- 2 – окно редактора скрипта;
- 3 – выпадающий список для выбора языка, на котором будет писаться скрипт;
- 4 – панель управления отладчика:



- запуск отчета на выполнение в режиме отладки;



- выполнение очередной строки кода (Step into);



- прерывание работы скрипта;



- просмотр значений выражений (Evaluate).

5 – на этом поле отображаются закладки (bookmark), точки останова (breakpoint), подсвечиваются строки, имеющие исполняемый код;

6 – вы также можете использовать кнопки на основной панели управления:



- вырезать текст;



- копировать текст;



- вставить текст;



- отмена последнего действия.

Ниже приведен список клавиш, которые можно использовать в редакторе скрипта.

Клавиша	Значение
Стрелки курсора	Перемещение курсора
PageUp, PageDown	Переход на предыдущую/последующую страницу
Ctrl+PageUp	Переход в начало текста
Ctrl+PageDown	Переход в конец текста
Home	Переход в начало строки
End	Переход в конец строки
Enter	Переход на следующую строку
Delete	Удаление символа в позиции курсора, удаление выделенного текста
Backspace	Удаление символа слева от курсора
Ctrl+Y	Удаление текущей строки
Ctrl+Z	Отмена последнего действия (до 32 событий)
Shift+Стрелки курсора	Выделение блока текста
Ctrl+A	Выделить весь текст
Ctrl+U	Сдвиг выделенного блока на 2 символа влево
Ctrl+I	Сдвиг выделенного блока на 2 символа вправо
Ctrl+C, Ctrl+Insert	Копирование выделенного блока в буфер обмена
Ctrl+V, Shift+Insert	Вставка текста из буфера обмена
Ctrl+X, Shift+Delete	Перенос выделенного блока в буфер обмена
Ctrl+Shift+<цифра>	Установка закладки с номером 0..9 на текущей строке
Ctrl+<цифра>	Переход на установленную закладку
Ctrl+F	Поиск строки
Ctrl+R	Замена строки
F3	Повторный поиск/замена с позиции курсора
F4 или F5	Установка точки прерывания и запуск скрипта (Run to cursor)
Ctrl+F2	Остановка скрипта (Program reset)
Ctrl+F7	Просмотр значений переменных (Evaluate)
F9	Запуск скрипта на выполнение (Run)
F7 или F8	Выполнение строки кода (Step into)

Структура скрипта

Структура скрипта зависит от используемого языка, но в ней можно выделить общие элементы. Это заголовок скрипта, тело и главная процедура, которая будет выполнена при запуске отчета на выполнение. Ниже приведены примеры скриптов для всех четырех поддерживаемых языков:

Структура PascalScript:

```
#language PascalScript // опционально
program MyProgram;      // опционально

// раздел uses - должен быть перед любым другим разделом
uses 'unit1.pas', 'unit2.pas';

var                      // раздел переменных - может быть в любом месте
  i, j: Integer;

const                   // раздел констант
  pi = 3.14159;

procedure p1;           // процедуры и функции
var
  i: Integer;

  procedure p2;         // вложенная процедура
  begin
  end;

begin
end;

begin                   // главная процедура.
end.
```

Структура C++Script:

```
#language C++Script     // опционально

// раздел include - должен быть перед любым другим разделом
#include "unit1.cpp", "unit2.cpp"

int i, j = 0;           // раздел переменных - может быть в любом месте

#define pi = 3.14159    // раздел констант

void p1()               // функции
{                       // вложенных процедур нет
}

{                       // главная процедура.
}
```

Структура JScript:

```
#language JScript          // опционально

// раздел import - должен быть перед любым другим разделом
import "unit1.js", "unit2.js"

var i, j = 0;              // раздел переменных - может быть в любом месте

function p1()              // функции
{                           //
}                           //

                           // главная процедура.
p1();
for (i = 0; i < 10; i++) j++;
```

Структура BasicScript:

```
#language BasicScript     // опционально

// раздел imports - должен быть перед любым другим разделом
imports "unit1.vb", "unit2.vb"

dim i, j = 0              // раздел переменных - может быть в любом месте

function p1()              // функции
{                           //
}                           //

                           // главная процедура.
for i = 0 to 10
    p1()
next
```

Более детальное описание возможностей скриптового движка FastScript можно найти в его документации. Автор не стал дублировать следующие моменты в настоящем руководстве:

- синтаксические диаграммы всех поддерживаемых языков;
- поддерживаемые типы данных;
- работа с классами, свойствами, методами, событиями;
- встроенные функции;
- перечисления, множества.

В дальнейшем мы будем рассматривать примеры скриптов на языке PascalScript. При создании нового отчета этот язык выбирается по умолчанию.

Скрипт "Hello, World!"

Мы уже рассматривали пример отчета "Hello, World!" – теперь посмотрим, как сделать простейший скрипт, выводящий на экран окно с приветственной надписью.

Создадим пустой проект в Delphi. Положим на форму компонент TfrxReport. Зайдем в дизайнер и нажмем кнопку "Новый отчет", чтобы FastReport автоматически создал пустой шаблон. Переключимся на закладку "Код" и напишем следующий скрипт:

```
begin
  ShowMessage('Hello, World!');
end.
```

После этого запустим отчет на выполнение. Как и ожидалось, FastReport вывел на экран маленькое окошко с приветствием:



Поясним некоторые моменты. Мы создали скрипт, состоящий из одного блока begin..end. Таким образом, наш скрипт имеет очень простую структуру – состоит только из главной процедуры (см. предыдущий раздел "Структура скрипта"). Главная процедура выполняется в момент старта отчета. В нашем случае она выводит окно с приветствием на экран, а после его закрытия завершается. После завершения главной процедуры начинается построение отчета.

Использование объектов в скрипте

Из скрипта можно обращаться к любому объекту отчета. Так, если в отчете есть страница Page1 и объект Memo1 – можно использовать их в скрипте, обращаясь к ним по именам, например:

```
Memo1.Color := clRed
```

Список объектов отчета, доступных из скрипта, отображается в служебном окне "Дерево отчета". Какие свойства объектов доступны в скрипте? Ответ простой – те, что видны в инспекторе объектов. А в нижней части инспектора есть подсказка по выбранному свойству. Оба окна (дерево отчета и инспектор) доступны во время работы со скриптом. Для получения подробной справки о свойствах и методах объектов используйте файл справки FastReport, который поставляется в комплекте.

Продemonстрируем сказанное небольшим примером. Поместим на страницу отчета объект "Текст" с именем MyTextObject и текстом "Тест". В скрипте напишем:

```
begin
```

```
MyTextObject.Color := clRed  
end.
```

Запустим отчет на выполнение и увидим, что цвет нашего объекта стал красным.

Обращение к переменным из списка переменных отчета

Из скрипта можно обращаться к любой переменной, которая определена в списке переменных отчета (пункт меню "Отчет|Переменные..."). Имя переменной при этом надо заключать в угловые скобки:

```
if <my variable> = 10 then ...
```

Альтернативный вариант – использование функции Get:

```
if Get('my variable') = 10 then ...
```

Изменение значения такой переменной возможно только с помощью процедуры Set:

```
Set('my variable', 10);
```

Аналогичным образом следует обращаться и к системным переменным, таким как Page#:

```
if <Page#> = 1 then ...
```

Обращение к полям БД

Так же, как и в случае с переменными, при обращении к полям БД следует использовать угловые скобки:

```
if <Table1."Field1"> = Null then...
```

И точно так же можно использовать функцию Get (вообще говоря, эта функция всегда используется в неявном виде для вычисления выражений, помещенных в угловые скобки).

Использование агрегатных функций в скрипте

В отличие от других функций, к агрегатным функциям (SUM, MIN, MAX, AVG, COUNT) необходимо обращаться с помощью угловых скобок или функции Get (см. "Особенности вызова агрегатной функции"):

```
if <Sum(<Table1."Field1">, MasterData1)> > 10 then...
```

Другая особенность агрегатной функции – она должна быть использована внутри объекта "Текст", после чего к ней можно обращаться в скрипте. Если использовать агрегатную функцию только в скрипте (без использования в объекте "Текст"), то будет выдано сообщение об ошибке. Так происходит потому, что для корректной работы агрегатной функции она должна быть привязана к определенному бэнду.

Вывод значения переменной в отчете

Чтобы показать содержимое какой-либо скриптовой переменной в отчете, надо описать эту переменную и присвоить ей значение. Вот простой пример скрипта:

```
var
  MyVariable: String;
begin
  MyVariable := 'Hello!';
end.
```

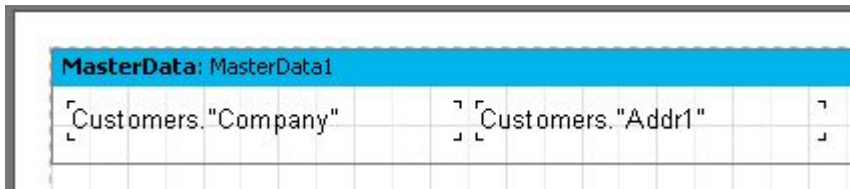
Вывести значение переменной можно, например, в объекте "Текст", поместив в него строку *[MyVariable]*.

Имя переменной должно быть уникальным, т.е. не должно совпадать с именами объектов отчета, стандартных функций, констант. При любой ошибке в скрипте на экран будет выведено сообщение и отчет строиться не будет.

События

До сих пор мы рассматривали скрипты с единственной главной процедурой, которая выполняется при старте отчета. В главной процедуре можно выполнить какие-либо начальные установки, инициализировать переменные. Но для полного контроля над процессом формирования отчета этого недостаточно. Чтобы максимально гибко управлять отчетом, каждый объект отчета имеет несколько событий, которым можно назначить обработчик – процедуру из скрипта. Например, можно в обработчике, привязанном к дата-бэнду, выполнять фильтрацию записей, т.е. скрывать или показывать бэнд в зависимости от каких-либо условий.

Рассмотрим процесс формирования отчета и события, которые при этом генерируются, на примере простого отчета, содержащего одну страницу, один бэнд "Данные 1 уровня" и два объекта "Текст" на бэнде:



В самом начале отчета, как уже говорилось, вызывается главная процедура скрипта. После этого начинается собственно процесс построения отчета. В начале отчета вызывается событие `OnStartReport` объекта "Отчет". Перед формированием страницы вызывается событие страницы `OnBeforePrint`. Это событие вызывается один раз для каждой страницы шаблона отчета (не путать со страницами готового отчета!). В нашем случае, сколько бы ни было страниц в готовом отчете – событие вызовется один раз, т.к. шаблон отчета состоит из одной страницы.

Далее начинается печать дата-бэндов. Происходит это следующим образом:

1. вызывается событие бэнда `OnBeforePrint`;
2. вызываются события `OnBeforePrint` всех объектов, лежащих на бэнде;
3. все объекты заполняются данными (в нашем случае – значениями полей БД `Company` и `Addr1`), после этого вызываются события `OnAfterData` всех объектов;
4. происходит позиционирование объектов на бэнде (если среди них есть растягиваемые объекты) и подсчет высоты бэнда и его растягивание (если бэнд растягиваемый);
5. вызывается событие бэнда `OnAfterCalcHeight`;
6. если бэнд не помещается на свободном месте страницы, формируется новая страница;
7. бэнд и все его объекты выводятся на страницу готового отчета;
8. вызывается событие `OnAfterPrint` всех объектов бэнда;
9. вызывается событие `OnAfterPrint` самого бэнда.

Печать бэндов происходит до тех пор, пока есть данные в источнике, подключенном к бэнду. После этого формирование отчета в нашем случае завершается и вызываются события `OnAfterPrint` страницы отчета и наконец – событие `OnStopReport` объекта "Отчет".

Таким образом, используя события разных объектов, можно контролировать практически каждый момент формирования отчета. Ключ к правильному использованию событий – полное понимание процесса печати бэндов, изложенного выше в девяти пунктах. Так, большинство действий можно выполнить, используя только событие бэнда `OnBeforePrint` – любые изменения, внесенные в объект, будут тут же отображены. Но в этом событии невозможно анализировать, на какой странице будет напечатан бэнд, если он растягиваемый – ведь подсчет высоты бэнда будет выполнен в пункте 4. Это можно сделать с помощью событий `OnAfterCalcHeight` в пункте 5 или `OnAfterPrint` в пункте 8, но в последнем случае бэнд уже будет напечатан и действия над объектами ничего не дадут. Одним словом, вы должны четко представлять, в какой момент времени вызывается каждое из событий и использовать те, которые соответствуют поставленной задаче.

Пример использования события OnBeforePrint

Продemonстрируем вышесказанное на практике. Создадим отчет – список клиентов, в котором будут представлены только компании, название которых начинается с буквы "А".

Создадим новый проект в Delphi, положим на форму компоненты TTable, TfrxDBDataSet, TfrxReport и настроим их:

Table1:

DatabaseName = 'DBDEMOS'

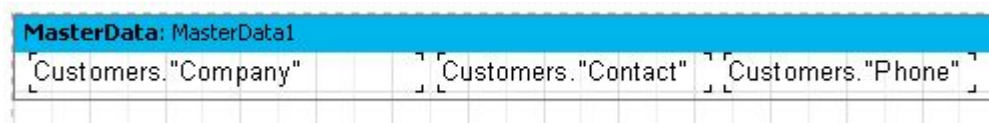
TableName = 'customer.db'

frxDBDataSet1:

DataSet = Table1

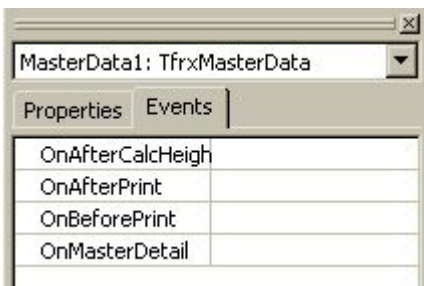
UserName = 'Customers'

Зайдем в редактор отчета и создадим отчет следующего вида:

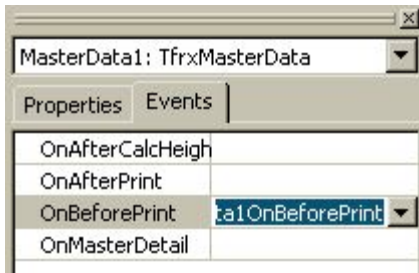


MasterData: MasterData1		
[Customers."Company"]	[Customers."Contact"]	[Customers."Phone"]

Выделим дата-бэнд и переключимся на закладку "События" в инспекторе объектов:



Чтобы создать обработчик события OnBeforePrint (именно оно нам подходит больше всего), надо сделать двойной щелчок мышью на пустом поле напротив имени события:



При этом в текст скрипта добавляется пустой обработчик и дизайнер переключается на закладку "Код":

```
procedure MasterData1OnBeforePrint(Sender: TfrxComponent);  
begin  
  |  
end;  
  
begin  
  
end.
```

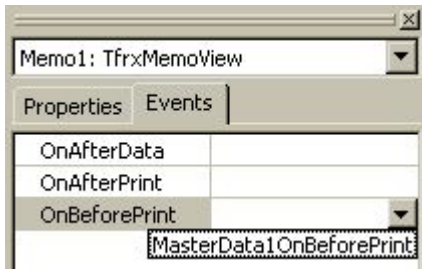
Как видим, все очень похоже на то, как работает среда Delphi. Нам остается только вписать следующий код в тело обработчика:

```
if Copy(<Customers."Company">, 1, 1) = 'A' then  
  MasterData1.Visible := True else  
  MasterData1.Visible := False;
```

Запустим отчет на выполнение и убедимся, что скрипт работает правильно:

Action Club	Michael Spurling	813-870-0239
Action Diver Supply	Marianne Miles	22-44-500211
Adventure Undersea	Gloria Gonzales	011-34-09054
American SCUBA Supply	Lynn Cinciripini	213-654-0092
Aquatic Drama	Gillian Owen	613-442-7654

Поясним некоторые моменты. Вы можете назначить один обработчик сразу для нескольких событий разных объектов – в этом случае параметр Sender определяет тот объект, который инициировал событие (аналогично параметру Sender в событиях Delphi). Чтобы присвоить событию имя уже существующего обработчика, можно ввести его вручную в инспекторе объектов, а можно выбрать из выпадающего списка – опять же, аналогично тому, как это происходит в среде Delphi:



Удаляется ссылка на обработчик просто – выделите нужное свойство и нажмите клавишу Delete.

Печать итоговой суммы по группе в заголовке группы

Этот довольно часто используемый прием требует использования скрипта. Ведь в обычном отчете значение суммы становится доступным только после того, как будут обработаны все записи группы. Чтобы вывести сумму в заголовке группы (т.е. до того, как будет обработана группа), используется следующий алгоритм:

- отчет делается двухпроходным;
- на первом проходе считается сумма по каждой группе и сохраняется в каком-нибудь массиве;
- на втором проходе значения извлекаются из массива и печатаются в заголовке группы.

Продemonстрируем, как решить эту задачу двумя способами. Для начала создадим новый проект в Delphi, на форму положим компоненты TQuery, TfrxReport, TfrxDBDataSet. Настроим их следующим образом:

```
Query1:
DatabaseName = 'DBDEMOS'
SQL =
select * from customer, orders
where orders.CustNo = customer.CustNo
order by customer.CustNo, orders.OrderNo
```

```
frxDBDataSet1:
DataSet = Query1
UserName = 'Group'
```

Зайдем в дизайнер и подключим наш источник данных к отчету. В настройках отчета (пункт меню "Отчет|Настройки") включим двойной проход. Добавим в отчет два бэнда: "Заголовок группы" и "Данные 1 уровня". В редакторе бэнда "Заголовок группы" укажем условие – поле данных Group.CustNo. Дата-бэнд привяжем к источнику данных Group и разместим объектами следующим образом:

GroupHeader: GroupHeader1		
[Group."CustNo"]	[Group."Company"]	
MasterData: MasterData1		
[Group."OrderNo"]	[Group."SaleDate"]	[Group."ItemsTotal"]
GroupFooter: GroupFooter1		
[SUM(<Group."ItemsTotal">,MasterData1)]		

Выделенный на рисунке объект (его имя – Memo8) мы используем для вывода суммы.

Способ 1.

Мы используем в качестве массива для хранения сумм класс TStringList. Значения будем хранить в виде строк. При этом первая строка в списке будет соответствовать значению первой группы, и т.д. Для подсчета номера группы будет использована целочисленная переменная, которую мы будем увеличивать после печати очередной группы.

Итак, наш скрипт будет выглядеть следующим образом:

```
var
  List: TStringList;
  i: Integer;

procedure frReport1OnStartReport(Sender: TfrxComponent);
begin
  List := TStringList.Create;
end;

procedure frReport1OnStopReport(Sender: TfrxComponent);
begin
  List.Free;
end;

procedure Page1OnBeforePrint(Sender: TfrxComponent);
begin
  i := 0;
end;

procedure GroupHeader1OnBeforePrint(Sender: TfrxComponent);
begin
  if Engine.FinalPass then
    Memo8.Text := 'Sum: ' + List[i];
end;

procedure GroupFooter1OnBeforePrint(Sender: TfrxComponent);
begin
  List.Add(FloatToStr(<SUM(<Group."ItemsTotal">,MasterData1)>));
  Inc(i);
end;
```

begin

end.

По именам процедур можно видеть, какие события мы использовали: Report.OnStartReport, Report.OnStopReport, Page1.OnBeforePrint, GroupHeader1.OnBeforePrint, GroupFooter1.OnBeforePrint. Что касается первых двух событий, то они, как уже говорилось, вызываются в начале и в конце отчета, соответственно. Чтобы создать обработчики для этих событий, надо выделить объект "Отчет" в окне "Дерево отчета" – его свойства появятся в инспекторе объектов. Далее действуем стандартным образом – переключаемся на закладку "События" инспектора и создаем обработчики.

Почему мы не воспользовались для создания списка List главной процедурой, а сделали это в событии OnStartReport? Потому, что созданный объект надо после завершения отчета освободить. Поэтому логично создавать объекты в событии OnStartReport, а освобождать их в OnStopReport. В других случаях (когда не нужно освобождать память) можно пользоваться главной процедурой для инициализации переменных.

С созданием и освобождением объекта List все понятно. Теперь рассмотрим, как работает скрипт. В начале страницы счетчик текущей группы (переменная i) сбрасывается в 0 и увеличивается на единицу после печати каждой группы (в событии GroupFooter1.OnBeforePrint). В этом же событии в список добавляется вычисленное значение суммы. Событие GroupHeader1.OnBeforePrint на первом проходе не срабатывает (проверка Engine.FinalPass). На втором проходе (когда список List заполнен значениями), в этом событии извлекается значение, соответствующее текущей группе, и записывается в текст объекта Memo8, который и показывает сумму в заголовке группы. В готовом отчете это выглядит так:

1221	Kauai Dive Shoppe	Sum: 51450,8
1023	01.07.88	4 674,00p.
1076	16.12.94	17 781,00p.
1123	24.08.93	13 945,00p.
1169	06.07.94	9 471,95p.
1176	26.07.94	4 178,85p.
1269	16.12.94	1 400,00p.
		51 450,80p.

Как видим, алгоритм достаточно простой. Но и его можно упростить.

Способ 2.

Мы используем в качестве массива для хранения сумм список переменных отчета. Как мы помним, обращение к таким переменным осуществляется с помощью функций Get и Set. Это избавит нас от необходимости создавать лишние объекты и освобождать память. Наш скрипт будет следующим:

```
procedure GroupHeader1OnBeforePrint(Sender: TfrxComponent);  
begin  
    if Engine.FinalPass then  
        Memo8.Text := 'Sum: ' + Get(<Group."CustNo">);  
end;  
  
procedure GroupFooter1OnBeforePrint(Sender: TfrxComponent);  
begin  
    Set(<Group."CustNo">,  
        FloatToStr(<SUM(<Group."ItemsTotal">,MasterData1)>));  
end;  
  
begin  
  
end.
```

Как видно, скрипт значительно упростился. Код в обработчике GroupFooter1.OnBeforePrint устанавливает значение переменной с именем, равным номеру клиента (можно использовать любой идентификатор, однозначно идентифицирующий клиента, например его имя <Group."Company">). Если такой переменной нет – она создается, если есть – меняется ее значение. В обработчике GroupHeader1.OnBeforePrint извлекается значение переменной с номером текущей группы.

Событие OnAfterData

Это событие генерируется после того, как объект отчета был наполнен данными, к которым он привязан. Событие удобно использовать для анализа значения поля БД или выражения, которое содержится в объекте. Дело в том, что это значение помещается в служебную переменную Value, значение которой доступно только в этом событии. Так, имея два объекта "Текст" с содержимым [Table1."Field1"] и [<Table2."Field1"> + 10], удобно анализировать значение этих выражений, ссылаясь на переменную Value:

```
if Value > 3000 then  
    Memo1.Color := clRed
```

вместо того, чтобы писать что-то вроде:

```
if <Table1."Field1"> > 3000 then  
    Memo1.Color := clRed
```

Более того, использование Value вместо выражения дает возможность написания одного универсального обработчика события OnAfterData и подключения его к нескольким объектам.

Одно замечание – если в объекте содержится несколько выражений, например *[expr1] [expr2]* – в переменную Value попадет значение последнего выражения.

Служебные объекты

Помимо объектов, имеющих в отчете (страницы, бэнды, объекты "Текст" и пр.), в скрипте доступны некоторые служебные объекты, которые могут пригодиться при управлении построением отчета. К таким объектам относится использованный нами в предыдущей главе объект Engine. Список служебных объектов приведен ниже:

- Report – объект "Отчет";
- Engine – ссылка на движок отчета;
- Outline – ссылка на элемент управления "Дерево отчета" в окне предварительного просмотра.

Рассмотрим каждый из объектов.

Объект Report

Представляет собой ссылку на текущий отчет. Свойства этого объекта можно видеть, выбрав элемент "Отчет" в окне "Дерево отчета".

Методы:

Метод	Описание
function Calc(const Expr: String): Variant	Возвращает значение выражения Expr, например, Report.Calc('1+2') вернет 3. В качестве выражения можно передавать любое выражение, являющееся корректным с точки зрения FastReport
function GetDataSet(const Alias: String): TfrxDataSet	Возвращает набор данных с указанным именем. Набор данных должен быть включен в список данных отчета (диалог "Отчет Данные...").

Объект Engine

Это самый полезный и интересный объект, который представляет собой ссылку на движок (ядро FastReport, управляющее построением отчета). Используя свойства и методы движка, можно строить воистину экзотические типы отчетов. Рассмотрим свойства и методы этого объекта.

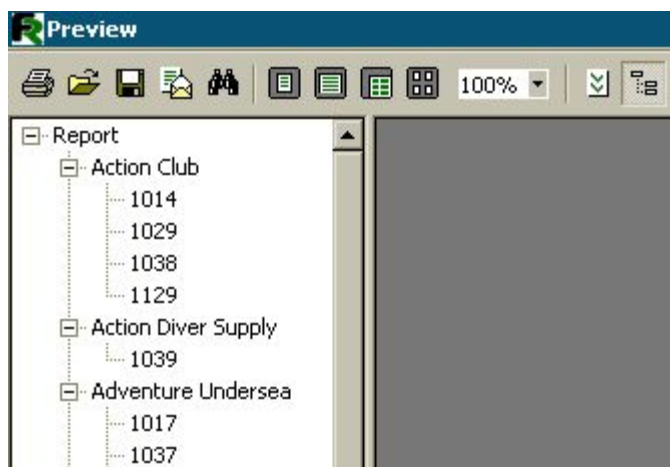
Свойство	Тип	Описание
CurColumn	Integer	Номер текущей колонки в многоколоночном отчете. Этому свойству можно присваивать значение.
CurX	Extended	Текущее смещение координат по оси X. Этому свойству можно присваивать значение.
CurY	Extended	Текущее смещение координат по оси Y. Этому свойству можно присваивать значение.
DoublePass	Boolean	Равно True, если отчет является двухпроходным. Аналогично Report.EngineOptions.DoublePass.
FinalPass	Boolean	Равно True, если выполняется последний проход двухпроходного отчета.
PageHeight	Extended	Высота области печати, в пикселах.
PageWidth	Extended	Ширина области печати, в пикселах.
StartDate	TDateTime	Время старта отчета. Аналог системной переменной <Date>.
StartTime	TDateTime	Время старта отчета. Аналог системной переменной <Time>.
TotalPages	Integer	Количество страниц в отчете. Аналог системной переменной <TotalPages>. Для использования этой переменной отчет должен быть двухпроходным.


Методы:

Метод	Описание
procedure AddAnchor(const Text: String)	Добавляет "якорь" в список якорей. Подробнее см. далее.
procedure NewColumn	Формирует новую колонку в многоколоночном отчете. После последней колонки автоматически формируется разрыв страницы.
procedure NewPage	Формирует новую страницу (разрыв страницы).
procedure ShowBand(Band: TfrxBand)	Показывает бэнд с указанным именем. После вывода бэнда автоматически смещается позиция CurY.
function FreeSpace: Extended	Возвращает высоту оставшегося свободного места на странице, в пикселах.
function GetAnchorPage(const Text: String): Integer	Возвращает номер страницы, на которой находится заданный якорь.

Объект Outline

Этот объект представляет собой элемент управления "Дерево отчета" в окне предварительного просмотра.



Этот элемент отображает древовидную структуру готового отчета. При щелчке на каком-либо узле дерева происходит переход на страницу отчета, связанную с этим узлом. Для отображения дерева нужно либо включить его кнопкой  на панели инструментов окна предварительного просмотра, либо указать это в свойстве `Report.PreviewOptions.OutlineVisible = True`. Там же можно указать ширину элемента управления в пикселах: `Report.PreviewOptions.OutlineWidth`.

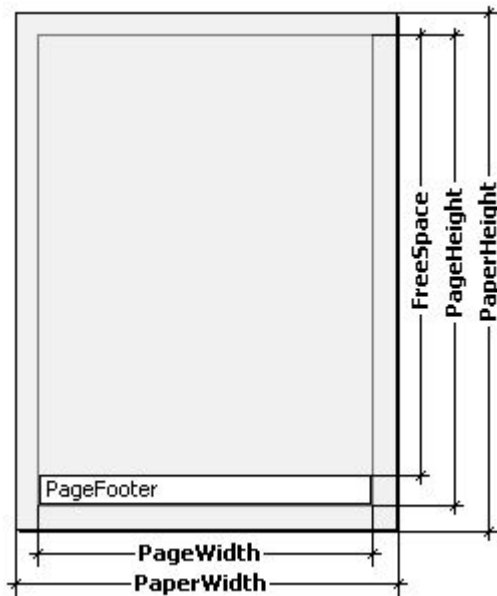
Рассмотрим методы этого объекта.

Метод	Описание
procedure AddItem(const Text: String)	Добавляет элемент с названием Text в текущую позицию дерева. С элементом ассоциируется текущая страница отчета и текущая позиция на странице.
procedure LevelRoot	Смещает текущую позицию в дереве на корневой уровень.
procedure LevelUp	Смещает текущую позицию в дереве на 1 уровень выше.

Применение объекта Engine

Мы уже упоминали, что объект Engine представляет собой движок отчета, управляющий построением отчета. Используя свойства и методы движка, можно управлять процессом размещения бэндов на странице. Для начала – немного теории.

На рисунке ниже представлено изображение страницы отчета и название свойств, которые возвращают то или иное измерение страницы.



Страница имеет физические размеры `PaperWidth`, `PaperHeight`. Эти размеры соответствуют одноименным свойствам страницы, что видно в инспекторе объектов при выборе страницы. Так, страница формата A4 имеет размеры 210x297мм.

Параметры `PageWidth`, `PageHeight` определяют размер области печати, которая почти всегда меньше физических размеров страницы. Размер области печати определяют поля страницы, которые задаются свойствами страницы отчета `LeftMargin`, `TopMargin`, `RightMargin`, `BottomMargin`. Размер области печати в пикселах возвращают свойства `Engine.PageWidth`, `Engine.PageHeight`.

Наконец, параметр `FreeSpace` определяет высоту свободного места на странице. Если на странице есть бэнд "Подвал страницы" (Page Footer), его высота учитывается при вычислении `FreeSpace`. Этот параметр в пикселах возвращает функция `Engine.FreeSpace`. Следует учесть, что после вывода очередного бэнда свободное место на странице уменьшается, что учитывается при вычислении `FreeSpace`.

Как происходит формирование страниц готового отчета? Ядро FastReport выводит бэнды на страницу до тех пор, пока на ней остается свободное место, достаточное для вывода бэнда. Когда свободного места не остается, печатается бэнд "Подвал страницы" (если он есть) и формируется новая пустая страница. Как уже говорилось, после вывода очередного бэнда высота свободного места уменьшается. Кроме того, вывод очередного бэнда начинается с текущей позиции, которая определяется координатами по оси X и Y. Эта позиция возвращается в свойствах `Engine.CurX`, `Engine.CurY`. После печати очередного бэнда позиция `CurY` автоматически увеличивается на высоту напечатанного бэнда. После формирования

новой страницы позиция CurY = 0. Позиция CurX изменяется при печати многоколоночных отчетов.

Свойства Engine.CurX, Engine.CurY доступны не только для чтения, но и для записи. Это значит, что можно смещать бэнды вручную, используя одно из подходящих событий. Например, имея отчет следующего вида:

MasterData: MasterData1		
[Customers."Company"]	[Customers."Contact"]	[Customers."Phone"]

можно напечатать его таким образом:

Action Club	Michael Spurling	813-870-0239
Action Diver Supply	Marianne Miles	22-44-500211
Adventure Undersea	Gloria Gonzales	011-34-09054
American SCUBA Supply	Lynn Cinciripini	213-654-0092
Aquatic Drama	Gillian Owen	613-442-7654
Blue Glass Happiness	Christine Taylor	213-555-1984

Это результат работы скрипта, назначенного событию OnBeforePrint бэнда:

```
procedure MasterData1OnBeforePrint(Sender: TfrxComponent);
begin
    Engine.CurX := Engine.CurX + 5;
end;
```

Манипуляция свойством CurY позволяет, например, напечатать бэнды внахлест:

Action Club	Michael Spurling	813-870-0239
Action Diver Supply	Marianne Miles	22-44-500211
Adventure Undersea	Gloria Gonzales	011-34-09054
American SCUBA Supply	Lynn Cinciripini	213-654-0092
Aquatic Drama	Gillian Owen	613-442-7654
Blue Glass Happiness	Christine Taylor	213-555-1984
Blue Jack Aqua Center	Ernest Barratt	401-609-7673
Blue Sports Club	Theresa Kunes	619-222-6704

Соответствующий скрипт:

```
procedure MasterData1OnBeforePrint(Sender: TfrxComponent);
begin
    Engine.CurY := Engine.CurY - 15;
end;
```

Метод Engine.NewPage позволяет вставлять разрыв страницы в нужном месте отчета. При этом печать продолжается с новой страницы. Так, в нашем примере, можно вставить разрыв после печати второй записи:

```
procedure MasterData1OnAfterPrint(Sender: TfrxComponent);  
begin  
    if <Line> = 2 then  
        Engine.NewPage;  
end;
```

Обратите внимание – теперь мы делаем это в событии OnAfterPrint, т.е. после того, как бэнд уже напечатан. Служебная переменная Line, напомним, возвращает порядковый номер записи.

Метод Engine.NewColumn вставляет разрыв колонки в многоколоночном отчете. После последней колонки этот метод формирует новую страницу.

Якоря

Якорь (anchor) – один из элементов системы гиперссылок, которая позволяет при щелчке на объекте готового отчета (в окне предварительного просмотра) перейти на элемент, связанный с этим объектом.

Якорь – это специальная метка, которая устанавливается методом Engine.AddAnchor. Якорь имеет имя, и ему соответствует номер страницы и позиция на странице. Перейти на якорь с указанным именем можно, поместив в свойство URL любого объекта отчета строку вида:

```
#ИмяЯкоря  
или  
#[ИмяЯкоря]
```

В последнем случае, при построении отчета FastReport раскроет выражение, находящееся в квадратных скобках.

При щелчке на этом объекте произойдет переход на то место отчета, где был добавлен якорь.

Якоря удобно использовать при построении раздела "Содержание" со ссылками на соответствующие разделы. Покажем, как это делается, на небольшом примере. Для этого нам понадобится уже знакомая таблица Customer.db.

Наш отчет будет двухстраничным (имеется в виду – две страницы в режиме дизайнера). На первой странице мы разместим раздел "Содержание", на второй – собственно список клиентов. При щелчке на строке содержания будет осуществлен переход на соответствующий элемент отчета.

Первая страница:

ReportTitle: ReportTitle1			
[Table of contents]			
MasterData: MasterData1			
Customers			
Customers."Company"			

В свойство URL объекта "Текст", который лежит на дата-бэнде, поместим строку:

```
#[Customers."Company"]
```

и установим свойства шрифта – синий цвет и подчеркивание, чтобы имитировать внешний вид гиперссылки.

Вторая страница:

ReportTitle: ReportTitle2			
Customers			
PageHeader: PageHeader1			
Company	Address	Contact	Phone
MasterData: MasterData2			
Customers			
Customers."Company"	Customers."Addr1"	Customers."Contact"	Customers."Ph

Чтобы добавить якорь, в скрипте бэнда MasterData2.OnBeforePrint напомним:

```
procedure MasterData2OnBeforePrint(Sender: TfrxComponent);
begin
  Engine.AddAnchor(<Customers."Company">);
end;
```

Вот и все, что нужно. Запустив отчет, убедимся, что наши "гиперссылки" работают.

Последнее, что можно упомянуть, это функция Engine.GetAnchorPage. Эта функция возвращает номер страницы, на которой был добавлен соответствующий якорь. Эта функция так же полезна для создания раздела "Содержание". Для ее использования отчет должен быть двухпроходным.

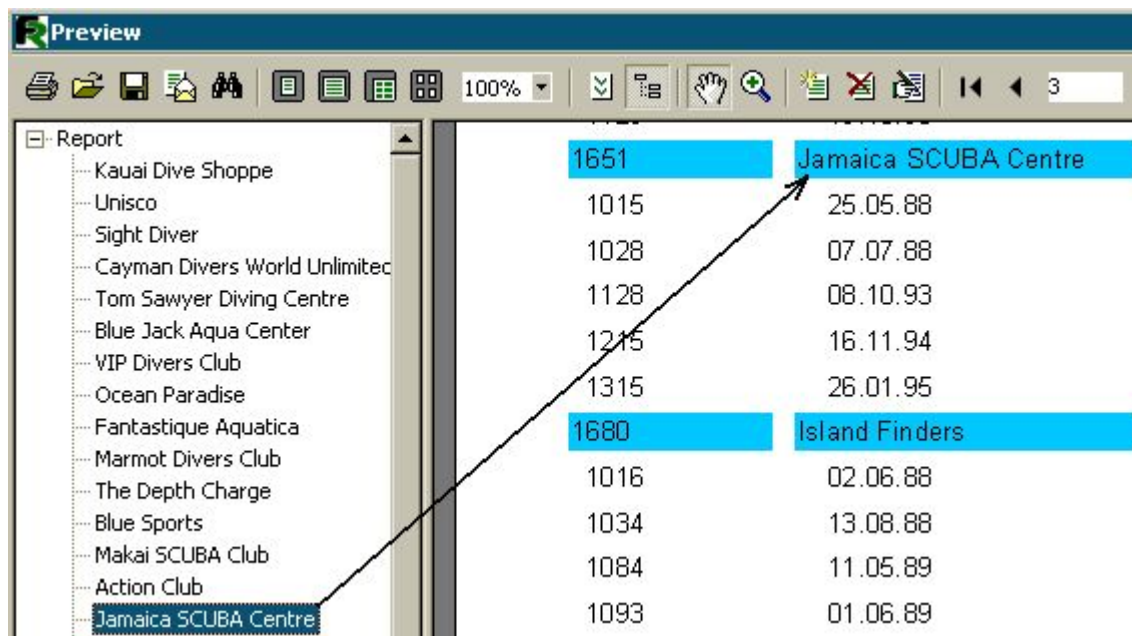
Применение объекта Outline

Объект Outline, как уже упоминалось, представляет собой дерево отчета, которое может быть показано в окне предварительного просмотра. При щелчке на элементе дерева произойдет переход на страницу отчета, которая связана с элементом дерева. Для работы с Outline необязательно использовать скрипт, т.к. некоторые бэнды имеют механизм, позволяющий формировать дерево автоматически. Рассмотрим два примера использования Outline, с помощью бэндов и из скрипта.

Для автоматического формирования дерева почти все бэнды имеют свойство OutlineText, в которое можно поместить строку-выражение. Выражение будет вычислено при формировании отчета и его значение при печати бэнда будет добавлено в дерево. При этом иерархия элементов в дереве повторяет иерархию бэндов в отчете. Это значит, что в дереве будут главные и подчиненные элементы, соответствующие главным и подчиненным бэндам в отчете (пример – отчет с двумя уровнями данных или с группами). Рассмотрим работу с деревом на примере отчета с группами, который мы изучали в предыдущей главе.

GroupHeader: GroupHeader1		Group."CustNo"
Group."CustNo"	Group."Company"	
MasterData: MasterData1		Group
Group."OrderNo"	Group."SaleDate"	

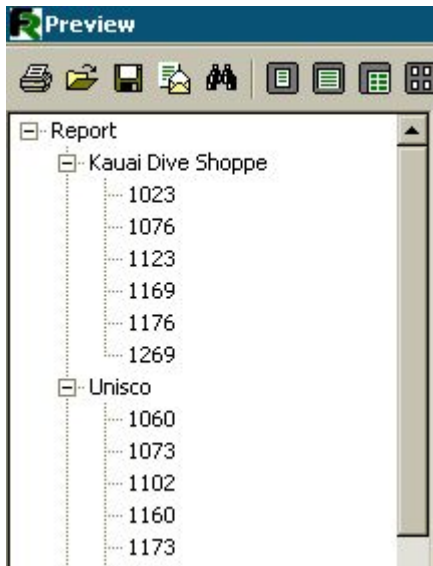
Укажем значение свойства бэнда GroupHeader1.OutlineText равным `<Group."Company">`. Чтобы автоматически включить панель с деревом отчета при показе окна предварительного просмотра, установим свойство `Report.PreviewOptions.OutlineVisible = True`. При запуске отчета мы увидим следующее:



	1651	Jamaica SCUBA Centre
	1015	25.05.88
	1028	07.07.88
	1128	08.10.93
	1215	16.11.94
	1315	26.01.95
	1680	Island Finders
	1016	02.06.88
	1034	13.08.88
	1084	11.05.89
	1093	01.06.89

При щелчке на любом элементе дерева произойдет переход на соответствующую страницу отчета таким образом, что выбранный элемент окажется в верхней части окна.

Давайте добавим второй уровень в дерево отчета. Для этого надо всего лишь установить свойство бэнда MasterData.OutlineText равным `<Group."OrderNo">`. При этом дерево будет выглядеть так:



Как видим, теперь возможна навигация и по номерам заказов, причем иерархия элементов дерева повторяет иерархию отчета.

Теперь покажем, как сформировать аналогичное дерево с помощью скрипта, без использования свойства OutlineText. В нашем отчете очистим свойства OutlineText обоих бэндов и создадим два обработчика событий GroupHeader1.OnBeforePrint и MasterData1.OnBeforePrint:

```
procedure GroupHeader1OnBeforePrint(Sender: TfrxComponent);  
begin  
    Outline.LevelRoot;  
    Outline.AddItem(<Group."Company">);  
end;  
  
procedure MasterData1OnBeforePrint(Sender: TfrxComponent);  
begin  
    Outline.AddItem(<Group."OrderNo">);  
    Outline.LevelUp;  
end;  
  
begin  
  
end.
```

Запустив отчет, убедимся, что он работает аналогично предыдущему отчету, где дерево формировалось автоматически. Рассмотрим, как происходит формирование дерева.

Метод `Outline.AddItem` добавляет к текущему узлу дерева дочерний узел и делает его текущим. Таким образом, если несколько раз подряд вызвать `AddItem`, то получится "лесенка" типа

```
Item1
  Item2
    Item3
    ...
```

Для управления текущим элементом служат методы `Outline.LevelUp` и `LevelRoot`. Первый метод перемещает указатель на элемент, расположенный уровнем выше. Так, скрипт

```
Outline.AddItem('Item1');
Outline.AddItem('Item2');
Outline.AddItem('Item3');
Outline.LevelUp;
Outline.AddItem('Item4');
```

построит дерево вида

```
Item1
  Item2
    Item3
    Item4
```

т.е. элемент `Item4` будет являться дочерним по отношению к элементу `Item2`. Метод `LevelRoot` передвигает текущий элемент в корень дерева. Например, скрипт

```
Outline.AddItem('Item1');
Outline.AddItem('Item2');
Outline.AddItem('Item3');
Outline.LevelRoot;
Outline.AddItem('Item4');
```

построит дерево вида

```
Item1
  Item2
    Item3
  Item4
```

После этих разъяснений понятно, как работает наш отчет. Каждый раз при печати заголовка группы текущим элементом делается корень дерева, куда добавляется имя компании. После этого печатается список заказов, и каждый заказ

добавляется в виде дочернего элемента компании. Чтобы номера заказов располагались на одном уровне, а не выводились в виде "лесенки", в скрипте делается переход на уровень вверх с помощью метода `Outline.LevelUp`.

Событие страницы `OnManualBuild`

Построением отчета обычно занимается ядро FastReport. Оно выводит бэнды отчета в определенной последовательности столько раз, сколько имеется данных, формируя таким образом готовый отчет. Иногда необходимо вывести отчет нестандартной формы, который ядро FastReport сформировать не в состоянии. В этом случае можно воспользоваться возможностью построения отчета вручную, с помощью события `OnManualBuild`, имеющегося у страницы отчета. Если определить обработчик этого события, ядро FastReport при формировании страницы передаст управление ему. При этом ядро отчета автоматически выводит имеющиеся на странице бэнды "Заголовок отчета", "Заголовок страницы", "Заголовок колонки", "Подвал отчета", "Подвал страницы", "Подвал колонки", "Фон". Ядро также обрабатывает формирование новых страниц/колонок. Задача обработчика события `OnManualBuild` – вывести в определенном порядке дата-бэнды и их заголовки/подвалы.

Т.е., суть обработчика `OnManualBuild` состоит в том, чтобы давать ядру FastReport команды на вывод определенных бэндов. Все остальное ядро сделает самостоятельно: сформирует новую страницу, когда место на текущей закончится, выполнит скрипты, прикрепленные к событиям и т.д.

Приведем пример простого обработчика. В отчете имеется два бэнда `master data`, не подключенных к данным:

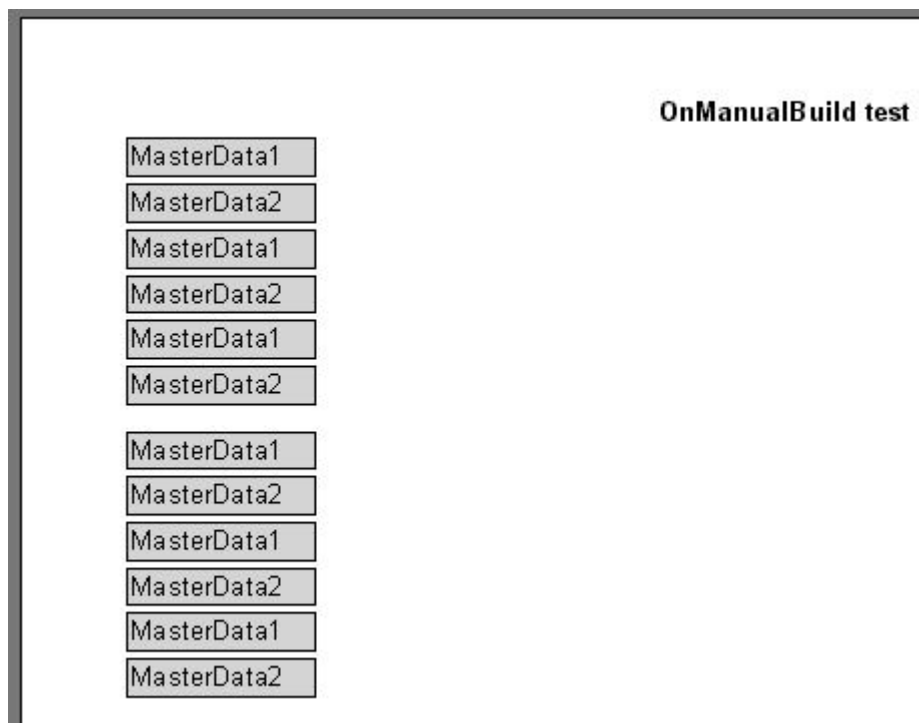
ReportTitle: ReportTitle1	
[OnManualBuild test]	
MasterData: MasterData1	
MasterData1	
MasterData: MasterData2	
MasterData2	
PageFooter: PageFooter1	

Обработчик выведет эти бэнды в чередующемся порядке, каждый по 6 раз. После шести бэндов будет сделан небольшой промежуток.

```

procedure Page1OnManualBuild(Sender: TfrxComponent);
var
    i: Integer;
begin
    for i := 1 to 6 do
    begin
        { выводим бэнды друг за другом }
        Engine.ShowBand(MasterData1);
        Engine.ShowBand(MasterData2);
        { делаем небольшой промежуток }
        if i = 3 then
            Engine.CurY := Engine.CurY + 10;
    end;
end;

```



Следующий пример выведет две группы бэндов рядом друг с другом.

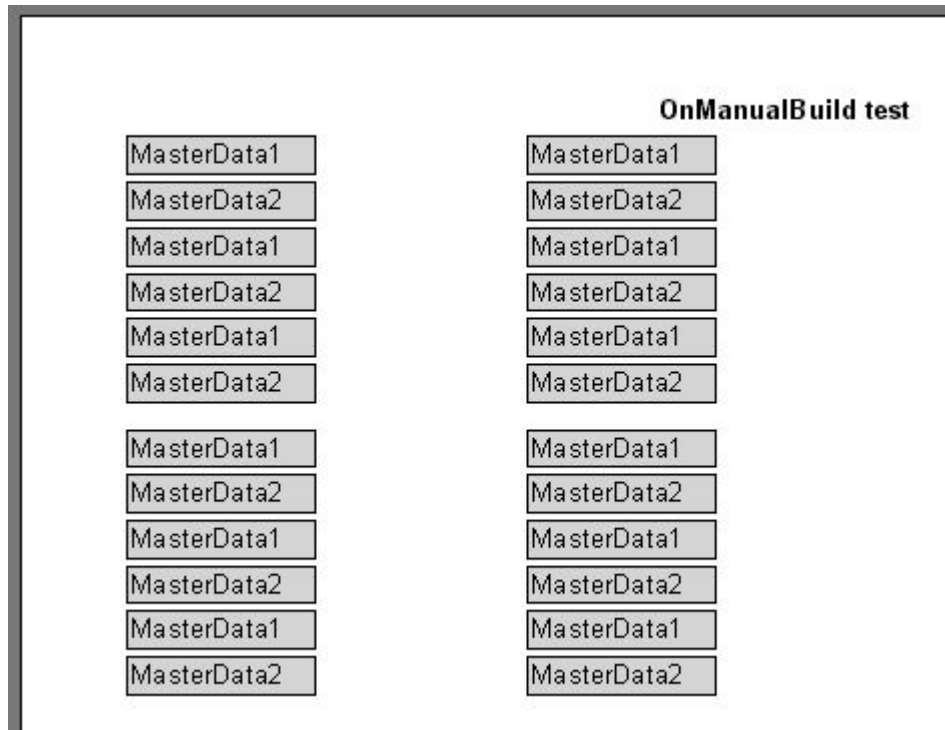
```

procedure Page1OnManualBuild(Sender: TfrxComponent);
var
    i, j: Integer;
    SaveY: Extended;
begin
    SaveY := Engine.CurY;
    for j := 1 to 2 do
    begin
        for i := 1 to 6 do
        begin
            Engine.ShowBand(MasterData1);
            Engine.ShowBand(MasterData2);
            if i = 3 then

```



```
        Engine.CurY := Engine.CurY + 10;  
    end;  
    Engine.CurY := SaveY;  
    Engine.CurX := Engine.CurX + 200;  
end;  
end;
```



Как видно на этих примерах, мы управляли только печатью дата-бэндов. Все остальные бэнды (например, "Заголовок отчета" в нашем случае) были напечатаны автоматически.

Наконец, покажем, как построить отчет типа "Список клиентов" (мы его строили неоднократно по ходу данной книги) с помощью события OnManualBuild. В нашем примере подключим дата-бэнд к источнику данных.

ReportTitle: Band1		
Customers		
PageHeader: Band2		
Company	Contact	Phone
MasterData: MasterData1		
Customers."Company"	Customers."Contact"	Customers."Ph
PageFooter: Band3		

Скрипт события следующий:

```

procedure Page1OnManualBuild(Sender: TfrxComponent);
var
    DataSet: TfrxDataSet;
begin
    DataSet := MasterData1.DataSet;
    DataSet.First;
    while not DataSet.Eof do
        begin
            Engine.ShowBand(MasterData1);
            DataSet.Next;
        end;
    end;

```

Запустив отчет, убедимся, что результат работы скрипта ничем не отличается от стандартного отчета. Обратим внимание на то, как получается ссылка на DataSet: в нашем примере мы подключили бэнд к источнику данных, поэтому строка

```
DataSet := MasterData1.DataSet;
```

вернет ссылку на источник данных. Если бэнд не подключен к источнику, то получить ссылку на нужный источник можно так:

```
DataSet := Report.GetDataSet('Customers');
```

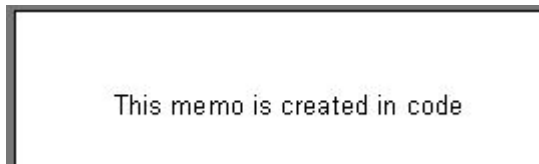
Естественно, интересующий нас источник должен быть добавлен в отчет в диалоге "Отчет|Данные...".

Создание объектов в скрипте

Используя скрипт, можно добавлять новые объекты в отчет. Покажем на маленьком примере, как это делается. Для этого создадим пустой отчет и напишем в главной процедуре скрипта:


```
var
    Band: TfrxReportTitle;
    Memo: TfrxMemoView;
begin
    Band := TfrxReportTitle.Create(Pagel);
    Band.Height := 20;
    Memo := TfrxMemoView.Create(Band);
    Memo.SetBounds(10, 0, 100, 20);
    Memo.Text := 'This memo is created in code';
end.
```

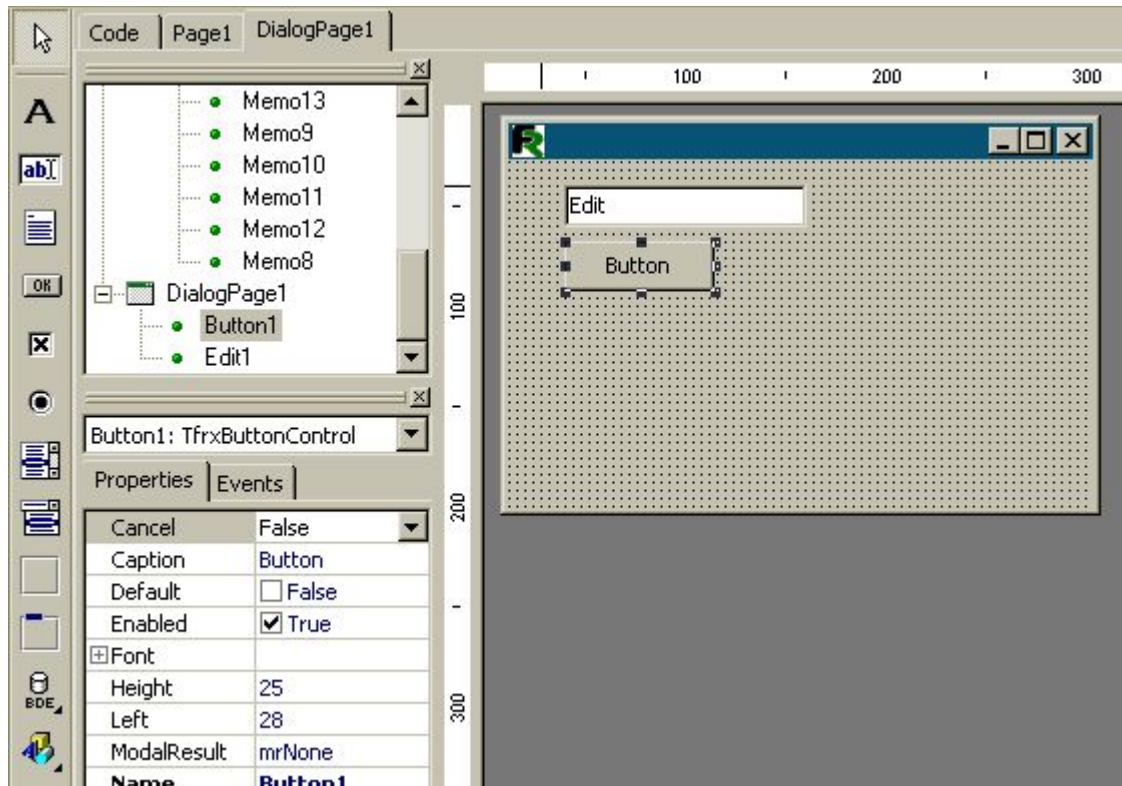
Запустим отчет:



Заметьте – мы нигде не разрушаем созданные объекты. Этого не требуется – объекты автоматически разрушатся после завершения формирования отчета.


Диалоговые формы











В отчете, помимо обычных страниц отчета, может быть несколько диалоговых форм. Для разработки диалоговых форм используется тот же дизайнер, что и для страниц отчета. Для создания новой формы служит кнопка  на панели инструментов дизайнера - она добавляет в отчет новую страницу. При переключении на страницу с формой диалога рабочее поле дизайнера изменяется - теперь это форма, на которой можно размещать объекты - элементы управления:



Не правда ли, напоминает среду Delphi?

Элементы управления

Элементы управления диалоговых форм подключаются при использовании в проекте компонента `TfrxDialogControls`  из палитры компонентов FastReport. Для этого достаточно положить компонент на любую форму в вашем проекте или добавить в список **uses** `frxDCtrl`. Это приводит к подключению следующих элементов управления:

Элемент	Название	Описание
	TfrxLabelControl	Назначение этого элемента управления - вывод поясняющей надписи на диалоговой форме
	TfrxEditControl	Элемент управления предназначен для ввода строки текста с клавиатуры.
	TfrxMemoControl	Элемент управления предназначен для ввода нескольких строк текста с клавиатуры.
	TfrxButtonControl	Элемент управления представляет собой кнопку.
	TfrxCheckBoxControl	Элемент управления представляет собой флажок, который может быть в двух состояниях: включенном и выключенном. Около флажка выводится поясняющая надпись.
	TfrxRadioButtonControl	Элемент управления представляет собой аналог переключателя с зависимой фиксацией. По этой причине в одиночку не применяется.
	TfrxListBoxControl	Элемент управления представляет собой список строк с возможностью выбора одной из них.
	TfrxComboBoxControl	Элемент управления представляет собой выпадающий список строк с возможностью выбора одной из них.
	TfrxDateEditControl	Элемент управления представляет собой поле ввода даты с выпадающим календарем.
	TfrxGroupBoxControl	Элемент управления представляет собой панель с поясняющей надписью, которая служит для объединения нескольких элементов управления.

	TfrxPanelControl	Элемент управления представляет собой панель, которая служит для объединения нескольких элементов управления.
	TfrxBitBtnControl	Элемент управления представляет собой кнопку с картинкой.
	TfrxSpeedButtonControl	Элемент управления представляет собой кнопку с картинкой.
	TfrxMaskEditControl	Элемент управления представляет собой поле для ввода информации по заданному шаблону.
	TfrxCheckListBoxControl	Элемент управления представляет собой список строк с флажками.
	TfrxBevelControl	Элемент управления предназначен для оформления диалоговой формы.
	TfrxImageControl	Элемент управления представляет собой картинку в формате BMP, ICO, WMF, EMF.

Как видно, все элементы управления аналогичны тем, что используются в Delphi. Справку по реализованным свойствам, событиям и методам каждого элемента можно получить в справочной системе FastReport.

Отчет "Hello, World!"

На этот раз мы создадим отчет, выводящий перед построением окно с приветственной надписью, используя диалоговую форму. Создадим новый проект в Delphi, положим на форму следующие компоненты: TfrxReport, TfrxDialogControls. Вызовем дизайнер FastReport двойным щелчком на компоненте TfrxReport и добавим в отчет диалоговую форму. На форму поместим объекты TfrxLabelControl, TfrxButtonControl:



Настроим свойства объектов:

```
TfrxLabelControl:  
Caption = 'Hello, World!'
```

```
TfrxButtonControl:  
Caption = 'OK'  
Default = True  
ModalResult = mrOk
```

У самой формы установим свойство `BorderStyle = bsDialog`. Как видим, все элементы управления и форма имеют тот же набор свойств, что и соответствующие элементы управления Delphi.

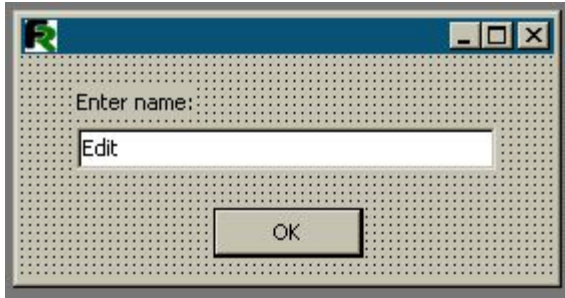
Закончив настройку диалоговой формы, вернемся на страницу отчета и поместим на нее объект "Текст" с каким-нибудь текстом внутри. Запустим отчет на выполнение и увидим нашу форму:



Если нажать кнопку ОК, отчет будет построен и показан. Если же закрыть окно кнопкой X, отчет строиться не будет. Таков алгоритм работы FastReport: при наличии в отчете диалоговых форм отчет будет построен только в том случае, если каждая форма была закрыта кнопкой ОК, т.е. вернула `ModalResult = mrOk`. Именно поэтому мы установили свойство `ModalResult` нашей кнопки равным `mrOk`.

Ввод параметров и передача их в отчет

Усложним наш пример, чтобы показать, каким образом можно передать введенные в диалоговой форме значения в отчет. Для этого изменим нашу форму следующим образом:



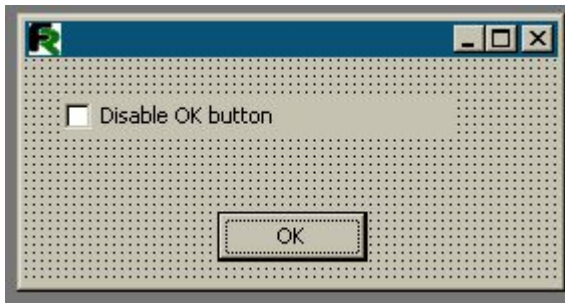
На странице отчета расположим объект "Текст" со следующим текстом внутри:

You entered:
[Edit1.Text]

Запустим отчет и убедимся, что введенный нами параметр успешно отображается в отчете. Аналогичным образом можно обращаться к другим объектам диалоговой формы. Так как каждый объект имеет имя, уникальное в пределах всего отчета, его можно использовать в любом месте отчета.

Взаимодействие элементов управления

Используя скрипт, можно легко реализовать логику работы диалоговой формы, например, взаимодействие ее элементов управления. Покажем это на простом примере. Модифицируем нашу форму следующим образом:



Дважды кликнем на объекте "CheckBox" – при этом создается обработчик события `OnClick`, и напомним следующий скрипт:

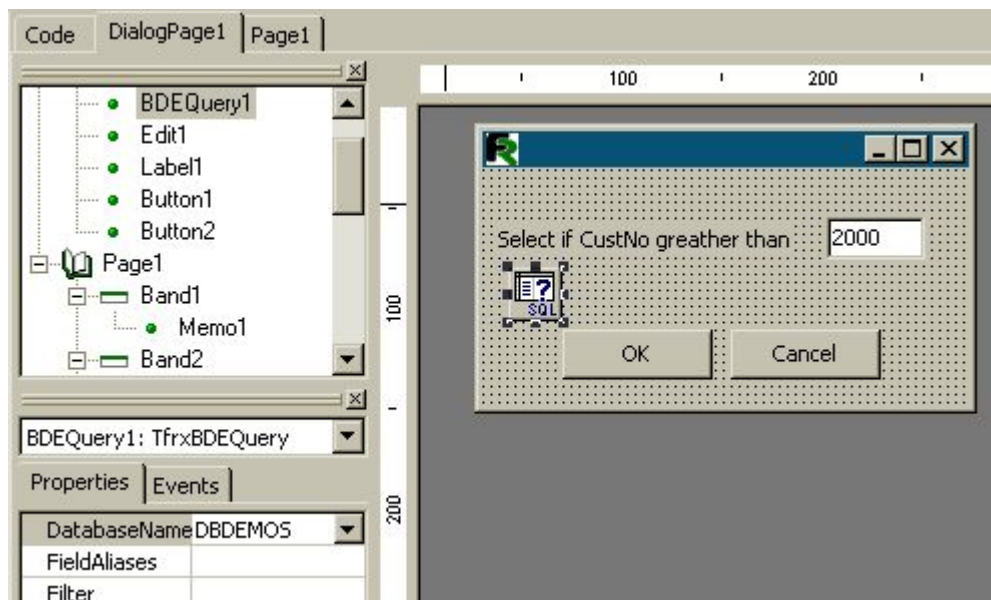
```
procedure CheckBox1OnClick(Sender: TfrxComponent);  
begin  
    Button1.Enabled := not CheckBox1.Checked;  
end;
```

Как видим, код ничем не отличается от того, что мы привыкли видеть в Delphi. Запустив отчет, увидим, что кнопка реагирует на изменение состояния флажка.

Компоненты доступа к данным


Большинство отчетов, как правило, основано на данных из БД. Для доступа к таким данным Delphi предоставляет эффективные механизмы, которые и используются в FastReport. Речь идет о компонентах TTable и TQuery, которые могут выступать в качестве источников данных для отчета. Вообще, можно использовать с этой целью любые компоненты - наследники TDataSet.





Кроме доступа к данным, определенным в проекте, FastReport позволяет создавать новые компоненты в run-time. В FastReport принципы создания компонентов доступа к данным максимально приближены к тем, что используются в среде Delphi. Так же, как и в Delphi, на форму кладется компонент и в инспекторе объектов настраиваются его свойства. Компонентная идеология очень гибкая: можно легко создавать новые компоненты для поддержки разных движков доступа к данным.



Описание компонентов

Рассмотрим использование компонентов для доступа к данным с помощью BDE. Они подключаются при использовании в проекте компонента

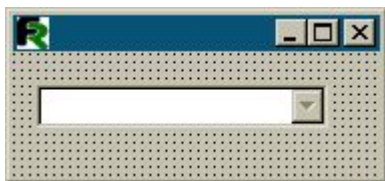
TfrxBDEComponents  из палитры FastReport. При этом в панели объектов дизайнера появляются следующие объекты: TfrxDBLookupComboBox, TfrxBDETable, TfrxBDEQuery, TfrxBDEDataBase. Эти компоненты по своему назначению аналогичны соответствующим компонентам Delphi (TDBLookupComboBox, TTable, TQuery, TDataBase).

Иконка	Название	Описание
	TfrxDBLookupComboBox	Элемент управления, предназначенный для выбора значения из справочника.
	TfrxBDETable	Предназначен для доступа к таблице БД.
	TfrxBDEQuery	Предназначен для выполнения SQL-запроса.
	TfrxBDEDataBase	Предназначен для соединения с БД.

Рассмотрим каждый компонент.

TfrxDBLookupComboBox

Этот элемент управления предназначен для выбора значения из таблицы-справочника. При этом вместо выбираемого значения подставляется его идентификатор в справочнике.



Элемент имеет следующие свойства:

Свойство	Описание
DataSet	Источник данных, к которому подключен элемент управления.
ListField	Имя поля БД, которое будет отображаться в элементе управления.
KeyField	Имя ключевого поля БД, которое будет идентифицировать выбранную запись.
KeyValue	Значение ключевого поля, которое было выбрано в списке.
Text	Значение поля БД, отображаемого в списке.

Для подключения элемента управления к справочнику необходимо заполнить значения трех свойств: DataSet, ListField и KeyField. Выбранное значение доступно через свойства Text или KeyValue, с помощью KeyValue можно установить начальную позицию указателя в списке.

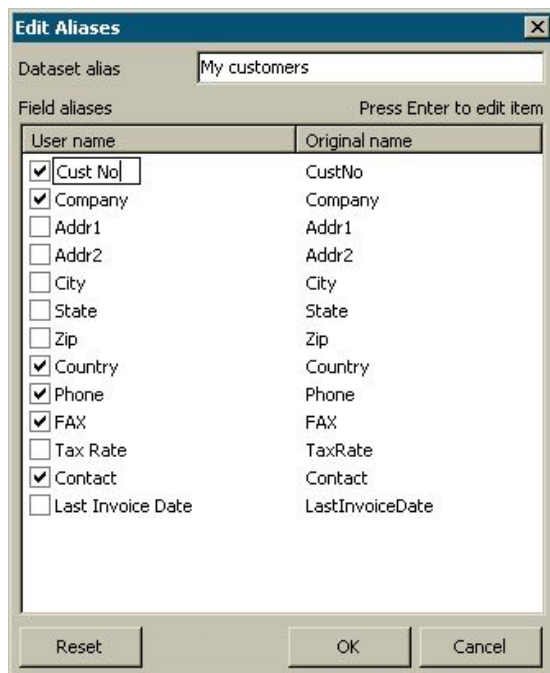
TfrxBDETable

Компонент предназначен для организации доступа к таблице БД. Компонент имеет следующие свойства:

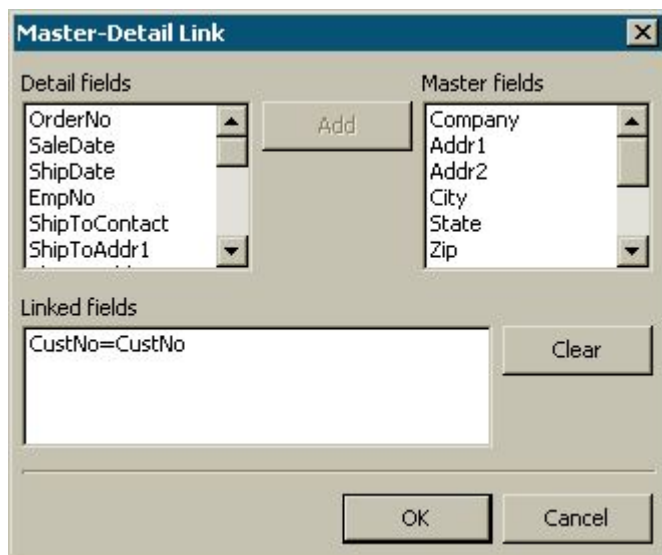
Свойство	Описание
Active	Определяет, активна ли таблица.
DatabaseName	Имя псевдонима (alias) БД.
FieldAliases	Позволяет задать пользовательские имена полей.
Filter	Выражение для фильтрации записей.
Filtered	Определяет, надо ли применять фильтр.
IndexName	Имя вторичного индекса.
MasterFields	Поля, связанные с master-набором данных.
Master	Мастер-набор данных.
SessionName	Имя сессии BDE.
TableName	Имя таблицы БД.

Назначения свойств компонента аналогичны свойствам Delphi TTable. Для подключения компонента к таблице БД достаточно заполнить свойства DatabaseName и TableName. Открытие таблицы осуществляется с помощью установки Active := True или с помощью метода Open.

Редактор свойства FieldAliases позволяет выбрать поля, которые будут доступны при обращении к таблице, и задать пользовательское имя для каждого поля и для всей таблицы.



Редактор свойства MasterFields используется для создания master-detail связей между двумя таблицами. Для связывания двух таблиц отношением master-detail у подчиненной таблицы надо указать в свойстве Master основную таблицу и вызвать редактор свойства MasterFields для подчиненной таблицы. Если у таблицы есть вторичные индексы, которые необходимо использовать, настройте предварительно свойство IndexName.



Здесь можно визуальнo связать поля master и detail наборов данных. Когда наборы связаны друг с другом отношением Master-Detail, то при перемещении по master набору содержимое detail набора фильтруется таким образом, чтобы в нем содержались только записи, имеющие отношение к текущей записи master набора.

Для связи полей наборов выделите поле из списка слева (detail набор), затем поле из списка справа (master набор), и нажмите кнопку "Добавить". При этом связка полей переместится в нижний список. Чтобы очистить нижний список, воспользуйтесь кнопкой "Очистить". Связываемые поля должны иметь одинаковый тип и быть ключевыми.

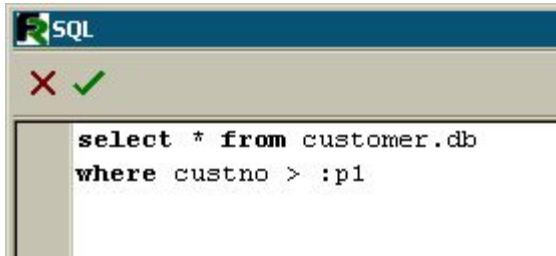
TfrxBDEQuery

Компонент предназначен для выполнения SQL-запросов к БД. Компонент имеет следующие свойства:

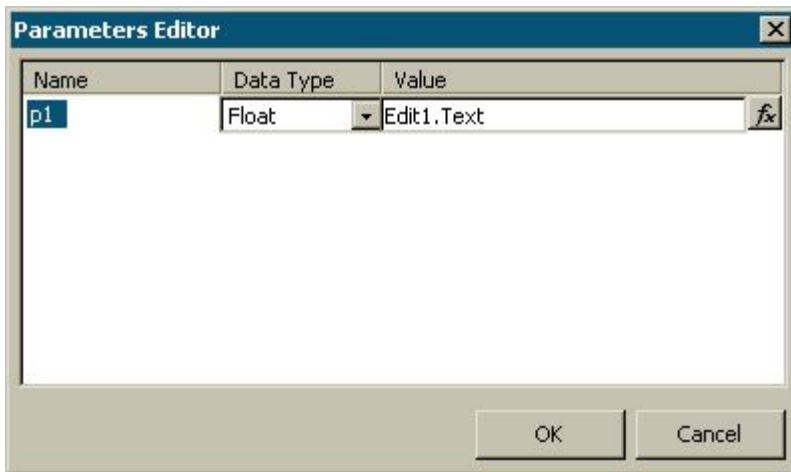
Свойство	Описание
Active	Определяет, активен ли запрос.
DatabaseName	Имя псевдонима (alias) БД.
FieldAliases	Позволяет задать пользовательские имена полей.
Filter	Выражение для фильтрации записей.
Filtered	Определяет, надо ли применять фильтр.
Master	Мастер-набор данных.

Params	Список параметров запроса.
SQL	Текст запроса.

Свойства Active, DatabaseName, FieldAliases, Filter, Filtered, Master аналогичны вышеописанным свойствам компонента TfrxBDETable. Свойство SQL имеет свой редактор для заполнения SQL-запроса.



Свойство Params также имеет свой редактор. Оно доступно, если текст запроса содержит параметры.



Параметр может быть двух типов: назначаемый из master-источника либо имеющий конкретное значение (причем в качестве значения может выступать как константа, так и ссылка на переменную или свойство объекта, как показано на рисунке).

В случае, когда параметр берется из master-набора данных, необходимо настроить свойство TfrxBDEQuery.Master. Набор данных должен содержать поле с именем, совпадающим с именем параметра. При этом указывать тип параметра и его значение необязательно.

TfrxBDEDataBase

Этот компонент служит для подключения к базе данных. Его назначение аналогично компоненту Delphi TDataBase. Компонент имеет следующие свойства:

Свойство	Описание
AliasName	Псевдоним, на основе настроек которого будет выполнено подключение к БД.
Connected	Если True, активизирует подключение.
DatabaseName	Имя, которое будет добавлено в список псевдонимов.
DriverName	Имя драйвера, обеспечивающего подключение к БД.
LoginPrompt	Определяет, надо ли запрашивать у пользователя пароль при подключении к БД.
Params	Параметры подключения.

Компонент осуществляет подключение к базе данных (обычно его используют для подключения к серверной СУБД). Настройки для подключения берутся либо из соответствующего алиаса (свойство AliasName), либо прописываются вручную (для этого необходимо указать имя драйвера DriverName). Компонент должен иметь заполненное свойство DatabaseName - это значение попадет в список псевдонимов.

Для установки параметров подключения нужно вызвать редактор свойства Params.

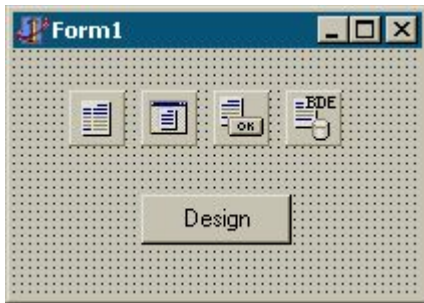
Свойство LoginPrompt определяет, надо ли спрашивать у пользователя пароль при подключении к БД. Если LoginPrompt = False, то имя пользователя и пароль должны быть указаны в параметрах подключения, например:

```
SERVER NAME=Путь_к_файлу_*.gdb  
USER NAME=SYSDBA  
PASSWORD=masterkey
```

Построение отчетов

Рассмотрим построение простого отчета, содержащего компоненты доступа к данным. В качестве данных будем использовать демонстрационные таблицы из поставки Delphi - DBDEMOS.

Для начала создадим проект, с помощью которого будем проводить эксперименты. Для этого создайте новый проект в Delphi и разместите на форме компоненты TfrxReport, TfrxDesigner, TfrxDialogControls, TfrxBDEComponents.



Для кнопки "Design" определите следующий обработчик:


```
procedure TForm1.Button1Click(Sender: TObject);  
begin  
    frxReport1.DesignReport;  
end;
```

После этого скомпилируйте и запустите проект. Это все, что требуется для создания end-user дизайнера отчетов.

При нажатии на кнопку Design открывается дизайнер, содержащий пустой отчет. Рассмотрим построение простых отчетов в этой среде.

Простой отчет типа "Список"

Этот отчет будет содержать данные из одной таблицы БД. Для построения отчета сделайте следующие шаги.

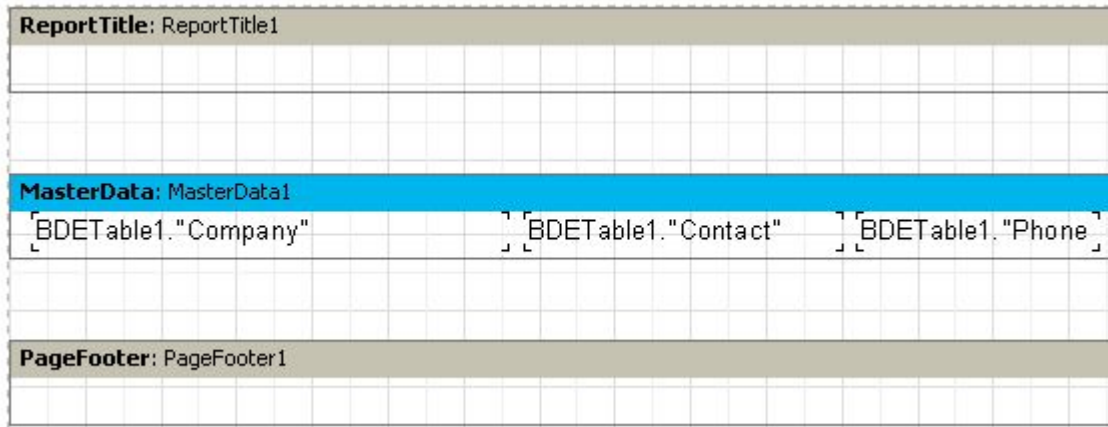
Нажмите кнопку "Новый отчет"  на панели инструментов дизайнера. При этом создается страница отчета с бэндами "Заголовок отчета", "Данные 1 уровня" и "Подвал страницы".


Добавьте в отчет диалоговую форму. Эта форма будет использована для размещения компонента доступа к таблице БД.

На форму положите компонент TfrxBDETable. Настройте его свойства:
DatabaseName = 'DBDEMOS'
TableName = 'Customer.db'

Переключитесь на страницу с формой отчета. Для подключения бэнда "Данные 1 уровня" к таблице, сделайте двойной щелчок на нем и в открывшемся окне выберите нашу таблицу.

Перетащите нужные поля из окна "Дерево данных" на лист отчета. После этого наш отчет будет выглядеть примерно так:

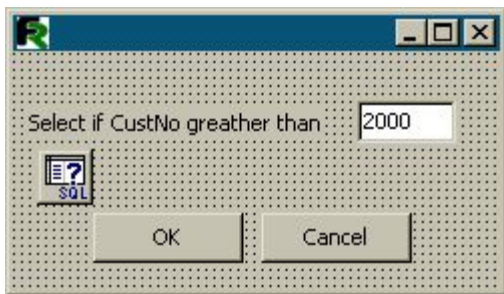


Для просмотра полученного отчета нажмите кнопку "Предварительный просмотр"  на панели инструментов.

Отчет с запросом параметров

Рассмотрим построение более сложного отчета, где перед построением отчета у пользователя запрашиваются параметры в диалоговом окне. Для этого сделайте следующие действия.

Добавьте в отчет диалоговую форму. Положите на форму отчета компоненты Query, Label, Edit, Button:



Настройте свойства компонентов:

Query1:

DatabaseName = 'DBDEMOS'

*SQL = 'Select * from Customer.db where CustNo > :p1'*

Label1:

Caption = 'Select if CustNo greather than'

Edit1:

Text = '2000'

Button1:

Caption = 'OK'

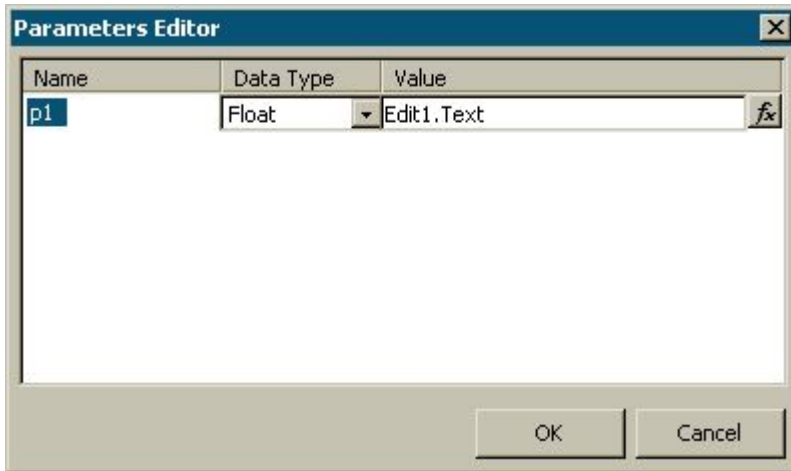
ModalResult = mrOk

Button2:

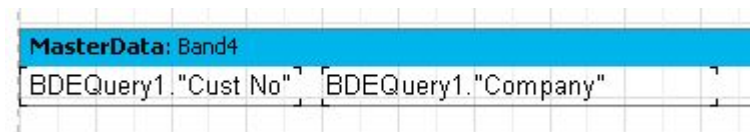
Caption = 'Cancel'

ModalResult = mrCancel

Откройте редактор свойства Params компонента Query и настройте параметр:



После этого перейдите на страницу с формой отчета и создайте отчет, аналогично тому, как мы сделали в предыдущем примере:



При построении отчета на экран будет выведен диалог, в котором предлагается ввести номер покупателя. После ввода нужного значения и нажатия кнопки ОК построение отчета будет выполнено. На печать выведутся покупатели с номерами, большими чем введенный.

