

Быстрые отчеты

Руководство пользователя FastReport FMX

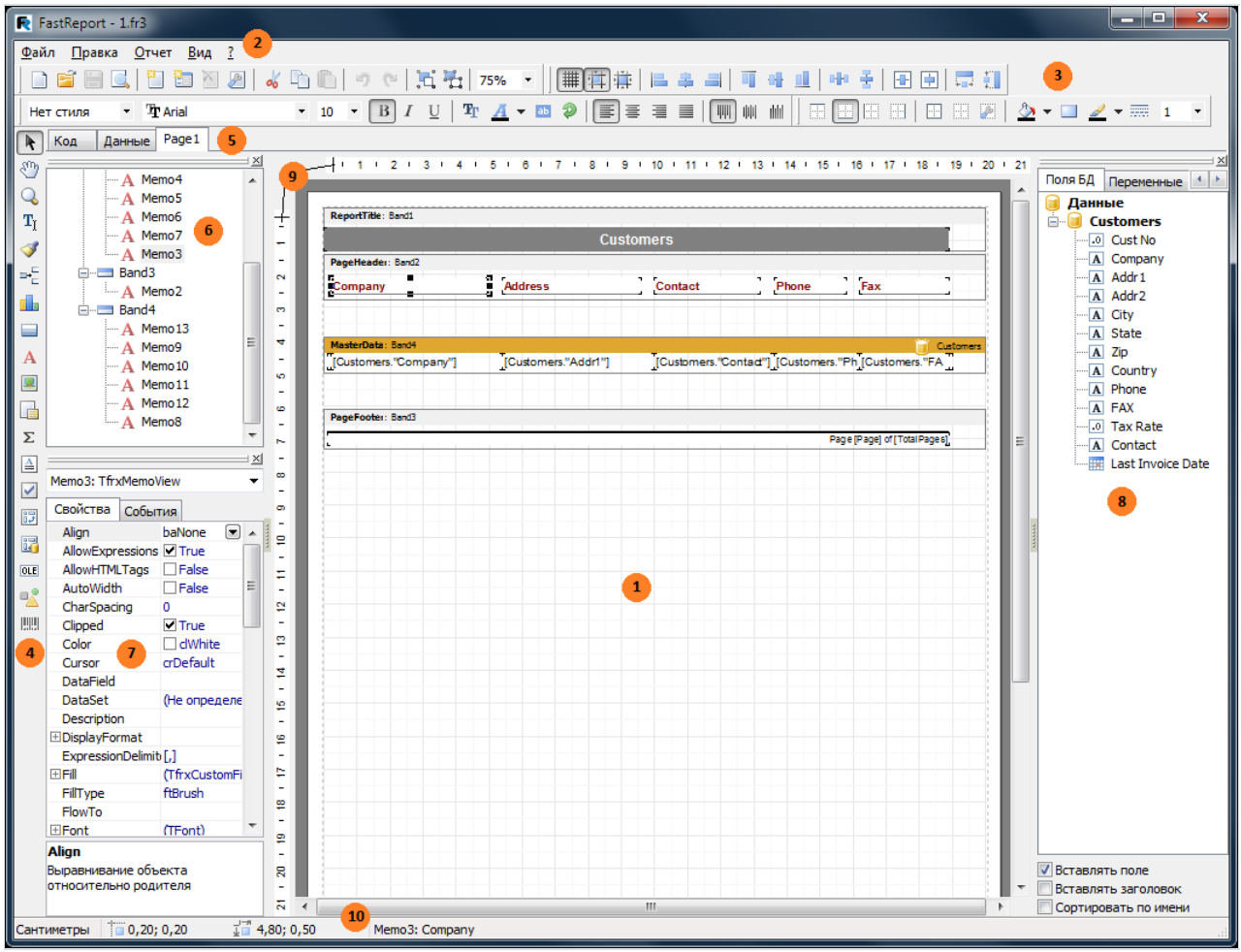
Версия 2026.2

© 2008-2026 ООО Быстрые отчеты

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, 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 frxDesgn. 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 – закладки страниц отчета и редактора кода;

6 – окно "Дерево отчета";

7 – окно "Инспектор объектов";

8 – окно "Дерево данных". Из этого окна можно перетаскивать элементы на лист отчета;

9 – линейки. При перетаскивании линейки на лист отчета образуется выносная линия, к которой могут прилипать объекты;

10 – строка состояния.

Клавиши управления

Клавиши	Описание
Ctrl+O	Команда меню "Файл/Открыть..."
Ctrl+S	Команда меню "Файл/Сохранить"
Ctrl+P	Команда меню "Файл/Просмотр"
Ctrl+Z	Команда меню "Правка/Отменить"
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, Tab	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.
Ctrl + Колесо мыши	Масштаб
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 создастся 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.

Иконка	Название	Описание
	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.
	Отменить	Отменяет последнюю операцию. Клавиатурный аналог - Ctrl+Z.
	Повторить	Повторяет последнюю отмененную операцию. Клавиатурный аналог – Ctrl+Y.
	Сгруппировать	Группирует выделенные объекты.
	Разгруппировать	Разгруппирует выделенные объекты.
75%	Масштаб	Задает масштаб страницы отчета.

Панель инструментов "Текст"



Иконка	Название	Описание
Нет стиля	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.
10	Размер шрифта	Позволяет выбрать размер шрифта из выпадающего списка. Размер можно также ввести вручную.
	Утолщение	Устанавливает/снимает утолщение шрифта.
	Наклон	Устанавливает/снимает наклон шрифта.
	Подчеркивание	Устанавливает/снимает подчеркивание шрифта.
	Свойства шрифта	Позволяет задать свойства шрифта, используя стандартный диалог выбора шрифта.
	Цвет шрифта	Выбирает цвет шрифта из выпадающего списка.
	Условное выделение	Показывает диалог с атрибутами выделения для выбранного объекта "Текст".
	Поворот текста	Позволяет выбрать поворот текста.
	Выравнивание влево	Устанавливает выравнивание текста влево.

Иконка	Название	Описание
	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.
	Редактор рамки	Вызывает редактор рамки.
	Цвет фона	Выбирает цвет фона из выпадающего списка.
	Цвет линии	Выбирает цвет линии из выпадающего списка.
	Редактор заливки	Вызывает редактор заливки.
	Стиль линии	Выбирает стиль линии из выпадающего списка.
1	Толщина линии	Выбирает толщину линии из выпадающего списка.

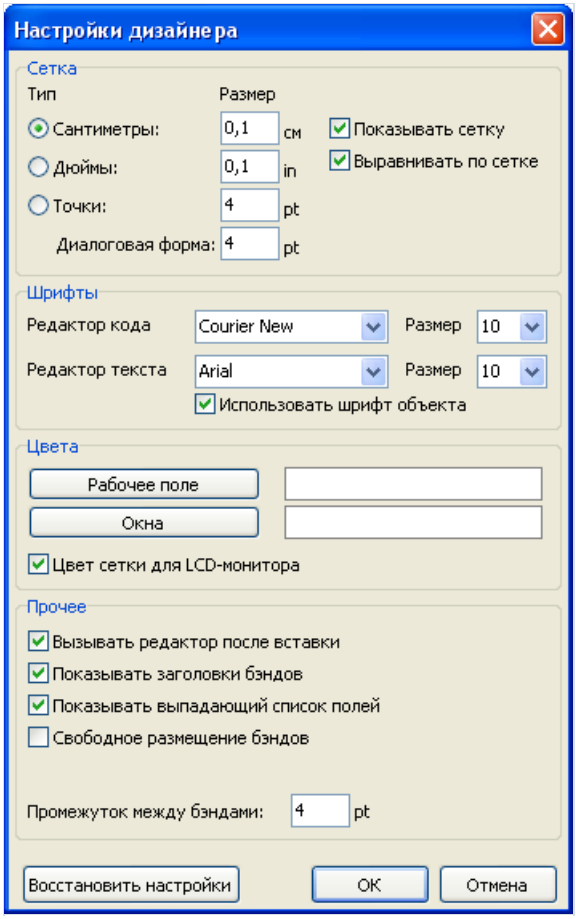
Панель инструментов "Выравнивание"



Иконка	Описание
	Показать сетку.
	Выравнивать объекты по сетке.
	Расположить в узлах сетки.
	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|Настройки...".



Здесь можно задать используемые единицы измерения (сантиметры, дюймы, пикселы), 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 шрифт 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 "Рабочее поле" и "Окна". Опция "Цвет сетки для LCD-монитора" слегка увеличивает контрастность линий сетки, что позволяет их лучше видеть на жидкокристаллических дисплеях.

Опция "Вызывать редактор 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.

Отключив опцию "Показывать выпадающий список полей", можно запретить показ выпадающего списка при наведении курсора мыши на объект "Текст", подключенный к данным. Это может быть необходимо, если

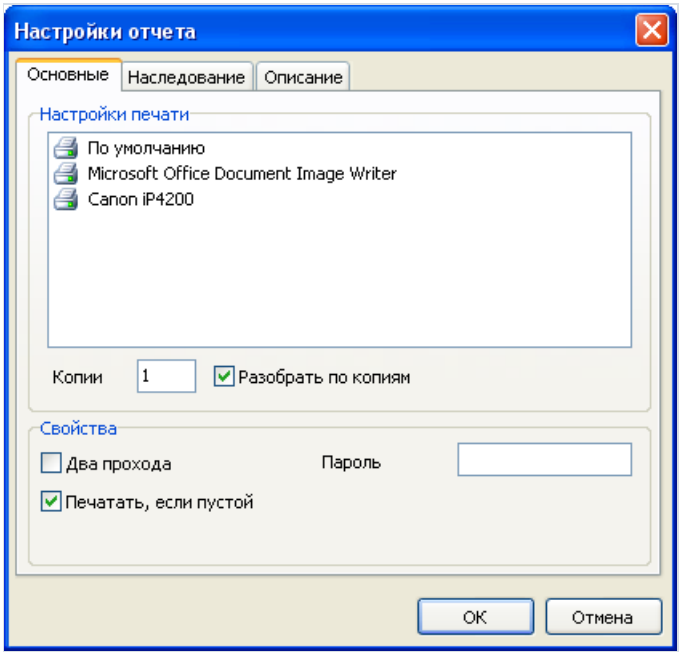
в отчете много мелких объектов.

Опция "Свободное размещение 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|Настройки...". Диалог имеет три страницы.

Первая страница - основные настройки:



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 текущий активный принтер.

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

Если флажок "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`.

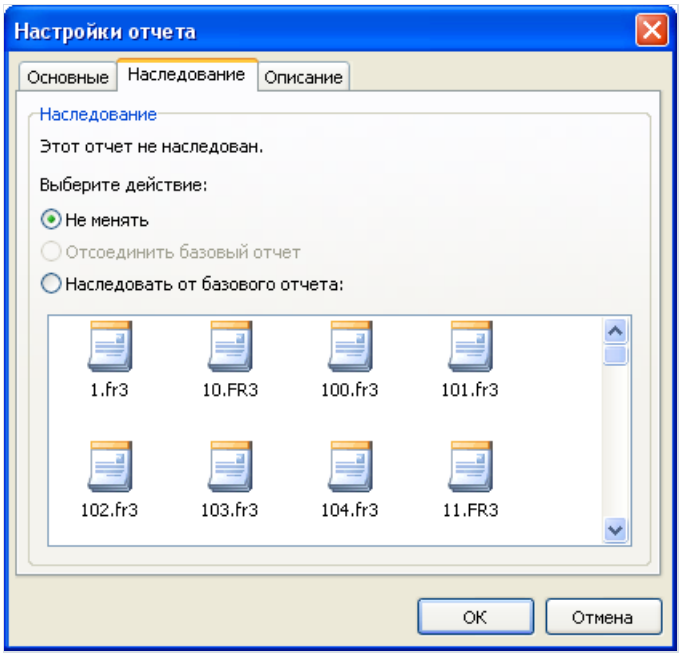
> Наиболее частая ошибка - попытка применить переменную `TOTALPAGES` в однопроходном отчете, в этом случае она возвращает 0.

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

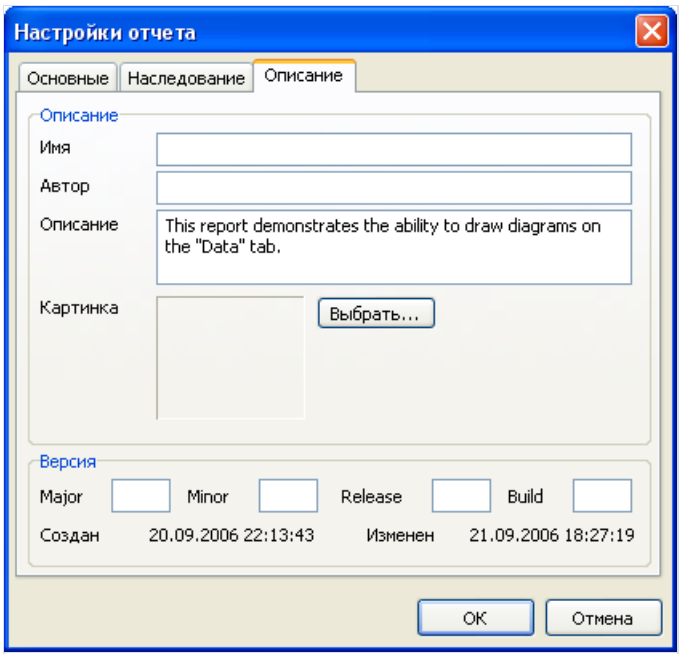
Поле "Пароль" позволяет задать пароль, который будет запрошен у пользователя при открытии отчета.

Элементы управления на второй странице позволяют задать опции наследования отчета.



Подробнее о наследовании см. в главе "Наследование отчетов". Здесь можно посмотреть, от какого отчета наследован текущий отчет, отсоединить базовый отчет (при этом отчет перестает быть наследованным и становится самостоятельным), а также наследовать отчет от одного из выбранных.

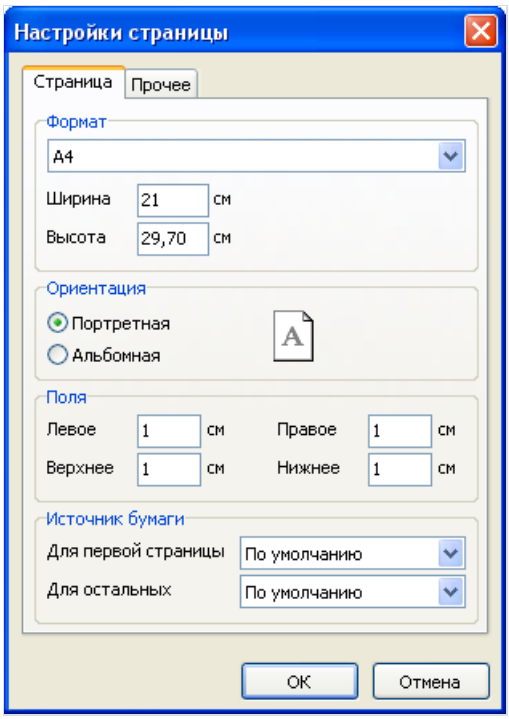
Элементы управления на третьей странице диалога позволяют задать описание отчета.



Вы можете задать такие параметры отчета, как имя (оно отображается в окне предварительного просмотра и присваивается заданию печати), автор, описание, картинка, версия. Все эти параметры являются необязательными и служат для информационных целей.

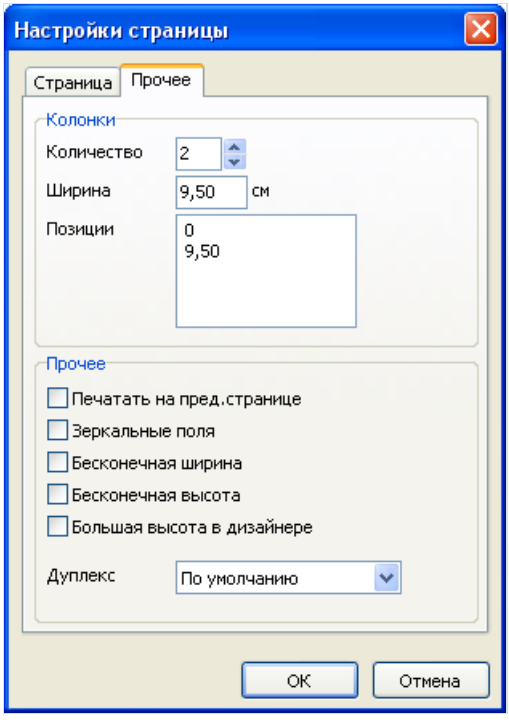
Параметры страницы

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 дизайнере.



Флажок "Печатать на пред. странице" 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.

Опции "Бесконечная ширина", "Бесконечная высота" позволяют включить режим, при котором размер страницы растет в зависимости от выводимых данных. При этом в окне предварительного просмотра вы видите одну безразмерную страницу, содержащую все данные (в отличие от нормального режима, когда при достижении конца страницы формируется новая страница).

Опция "Большая высота в дизайнере" 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

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 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 – линии, стрелки, прямоугольник, прямоугольник с круглыми углами, эллипс, треугольник, ромб.
Диаграмма		Отображает данные в виде диаграмм различных типов (круговая диаграмма, гистограмма и т.п.).
CheckBox		Отображает значок - галочку или крестик.
Кросс-таблица		Позволяет строить сводную таблицу.
Штрихкод		Отображает данные в виде штрихкода (доступно около десятка разных типов штрихкодов).

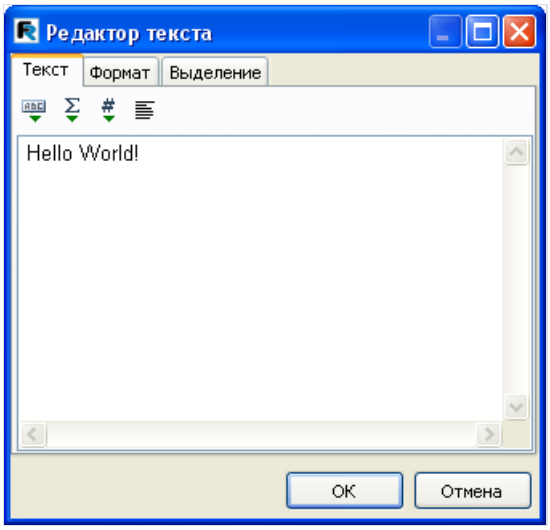
Основные объекты, с которыми вам придется работать больше всего - это объекты "Бэнд" и "Текст". Далее по ходу этой главы вы подробно ознакомитесь с их возможностями.

Отчет "Hello, World!"

Раасмотрим создание простейшего отчета, который будет содержать всего одну надпись "Hello, World!".

- Открываем дизайнер FastReport.
- 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 OK.

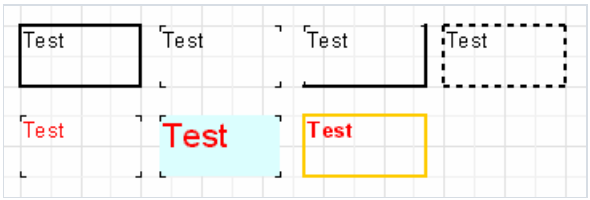


Отчет готов. Для его просмотра выберите пункт "Файл/Просмотр", или нажмите кнопку на панели инструментов. Вы увидите окно просмотра с единственной страницей отчета, которая содержит надпись "Hello World!". Полученный отчет можно распечатать, сохранить в файл или экспортировать в один из поддерживаемых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

Изучаем объект "Текст"

Объект "Текст" обладает очень широкими возможностями. 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.

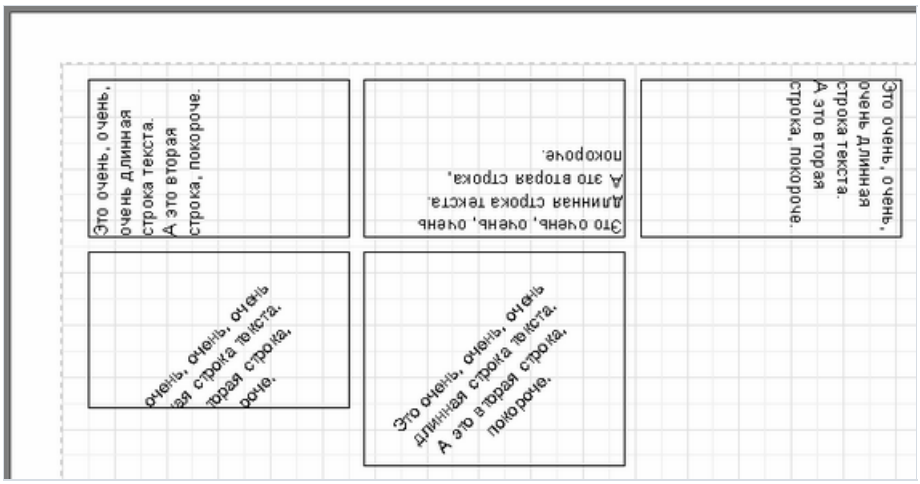
Теперь проверим, как работает выравнивание текста внутри объекта. Кнопки выравнивания расположены на панели инструментов "Текст" и позволяют независимо задать выравнивание текста по горизонтали и по вертикали. Обратите внимание на кнопку "Выравнивание по ширине" - она позволяет выровнять параграф по 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, установите свойство `Rotation` в инспекторе объектов.

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.

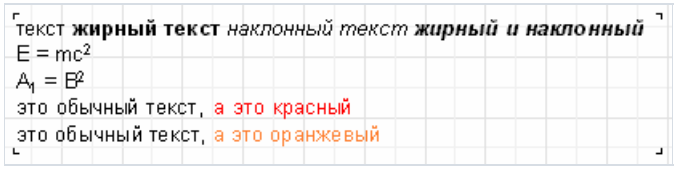
HTML-тэги в объекте "Текст"

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

Тэг	Значение
<b>	жирный текст
<i>	наклонный текст
<u>	подчеркнутый текст
<strike>	зачеркнутый текст
<sub>	подстрочный текст
<sup>	надстрочный текст
<font color>	цвет шрифта
<nowrap>	текст не разрывается при использовании `WordWrap`, а переносится целиком

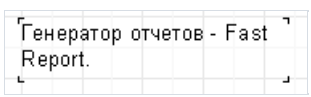
Продемонстрируем применение тэгов на примерах.

```
текст <b>жирный текст</b> <i>наклонный текст</i> <b><i>жирный и наклонный</b></i>
E = mc<sup>2</sup>
A<sub>1</sub> = B<sup>2</sup>
это обычный текст, <font color=red>а это красный</font>
это обычный текст, <font color="#FF8030">а это оранжевый</font>
```

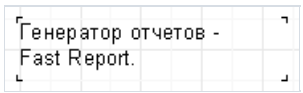


Пример использования тэга `<nowrap>`:

```
Генератор отчетов - Fast Report.
```




```
Генератор отчетов - <nowrap>Fast Report.</nowrap>
```



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 - поместим в объект "Текст" следующую 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 тексте" в контекстном меню), и все выражения, имеющиеся в тексте, будут проигнорированы.

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%>
```

Используем бэнды

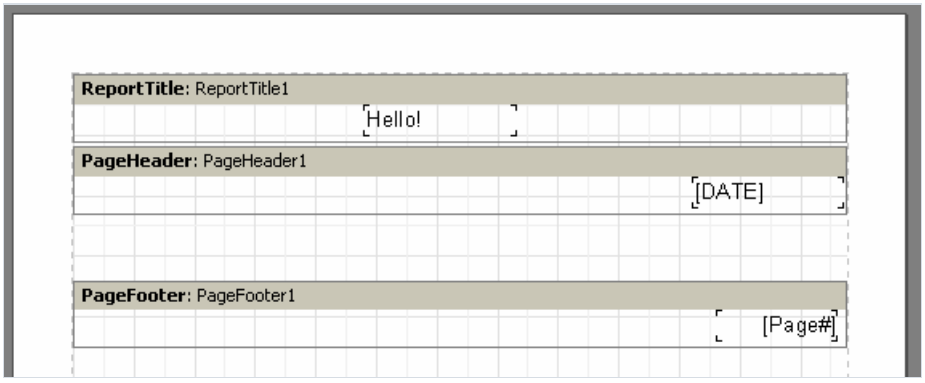
Слово "бэнд" (band) по-английски означает "полоска". Бэнды применяются для логической группировки объектов. Так, разместив объект на бэнде типа "Заголовок страницы", мы тем самым говорим FastReport, что данный объект надо вывести на каждой странице готового отчета вверху. Аналогичным образом бэнд "Подвал 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.

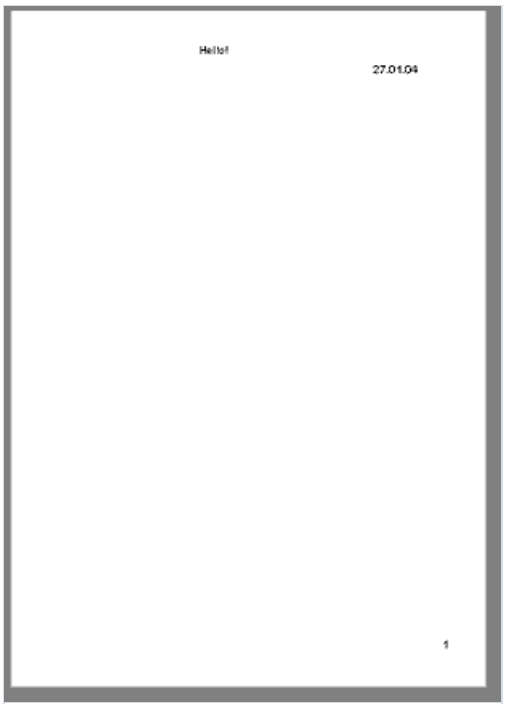
Зайдите в дизайнер FastReport и нажмите кнопку "Новый отчет" на панели инструментов. Вы увидите шаблон отчета, который уже содержит три бэнда: "Заголовок отчета", "Данные 1 уровня" и "Подвал страницы". Пока удалим бэнд "Данные 1 уровня" - щелкните мышкой на любом свободном месте внутри бэнда или на его 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 вверху 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 вверху, но после бэнда "Заголовок страницы" (это регулируется свойством страницы `TitleBeforeHeader`, которое задается в инспекторе объектов);
- 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 заранее неизвестное количество строк (записей), каждая из которых содержит определенное количество колонок (полей). Для печати такого рода информации FastReport использует особый тип бэндов - бэнды-данные, или дата-бэнды. Это бэнды с названиями "Данные xxx уровня". Чтобы напечатать всю таблицу или некоторые ее поля, необходимо:

- добавить дата-бэнд в отчет;
- подключить его к таблице;
- разместить на нем объекты "Текст" 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 дизайнере FastReport выберите пукнт 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 OK, закрываем диалог и видим, что поле вставлено в текст.

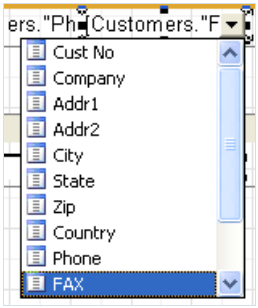
Второй способ вставки поля БД в отчет похож на тот, что широко применяется в среде Delphi - мы сделаем это с помощью настройки свойств в инспекторе объектов. Положим на бэнд второй объект, в редакторе ничего писать не будем. В инспекторе настроим свойства объекта:

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

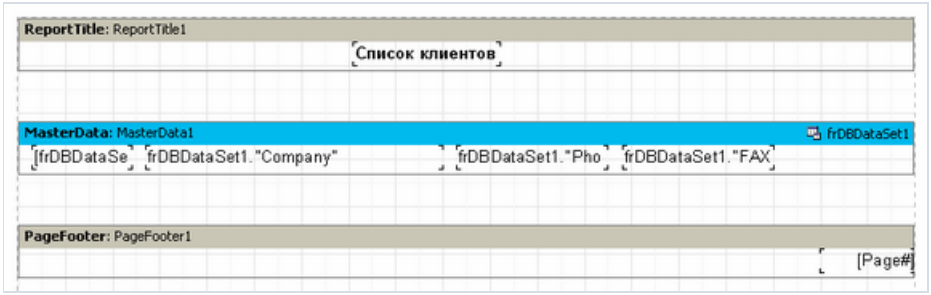
Т.к. оба свойства представляют собой список, нам достаточно выбрать нужные значения мышкой.

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. Эта возможность настраивается в опциях дизайнера (меню "Вид/Настройки...", флажок "Показывать выпадающий список полей").



Итак, наш отчет готов:



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, имеющихся 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:

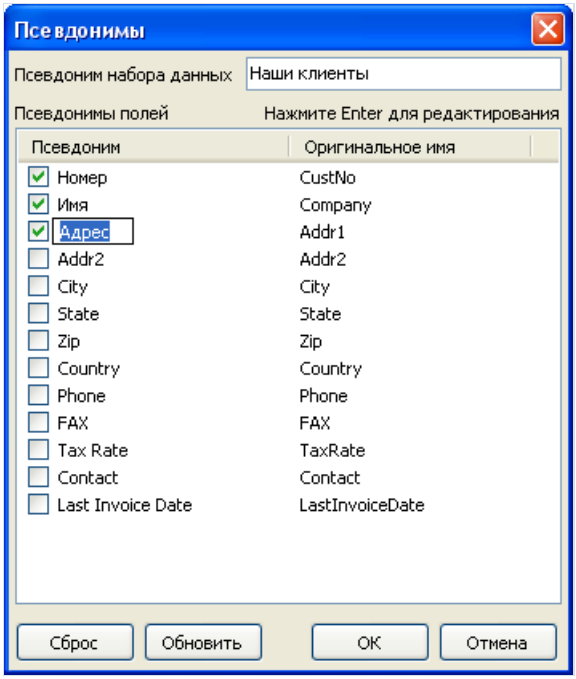
```
Длина в см: [<frxDBDataSet1."Length_in"> * 2.54]
```

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. Хочется переименовать источник данных в "Наши клиенты", а поле - в "Номер".

Это легко сделать, используя псевдонимы (алиасы). 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 `frxDBDataSet1`, 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:



Заметим, что алиас самого источника можно поменять и без использования редактора алиасов - для этого измените свойство `UserName` компонента `frxDBDataSet1`.

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.

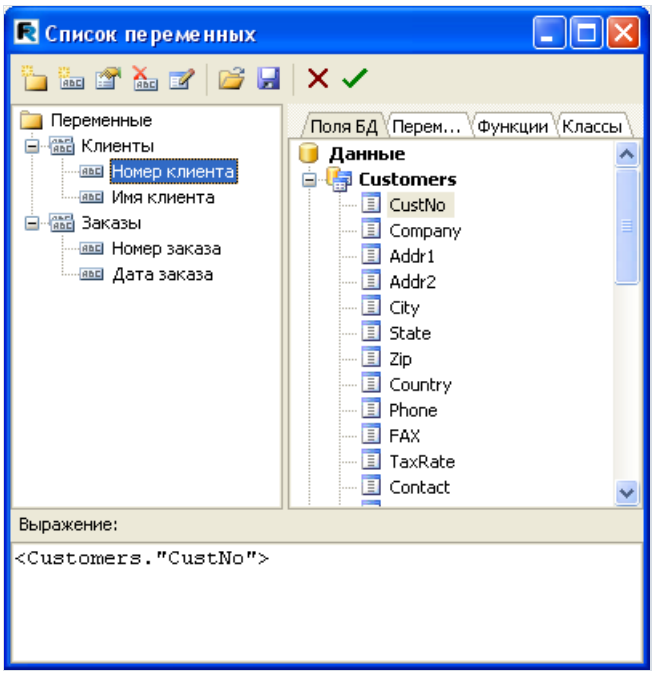
Кроме того, категории нужны только для логической группировки переменных и в отчет не вставляются. Поэтому, давая имя переменной, не забывайте, что оно должно быть уникально - две одинаковые переменные в разных категориях создать нельзя.

Продемонстрируем использование переменных на примере. Допустим, у нас есть два источника данных: первый - Customers с полями "CustNo" и "Name" и второй - Orders 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 OK.



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

- Авторазмер (`AutoSize`)
- Растягивание (`Stretch`) - включено по умолчанию
- Центрировать (`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 "Центрировать" 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.

Наконец, еще одно полезное свойство - `FileLink`, доступное из инспектора объектов. Здесь можно указать имя файла с картинкой, например: c:\picture.bmp. Картинка будет загружена при запуске отчета на выполнение. Также в это свойство можно поместить переменную, например: [picture_file]. При запуске отчета FastReport вычислит значение переменной (это должно быть все то же имя файла) и загрузит картинку.

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.

Продемонстрируем 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'
```

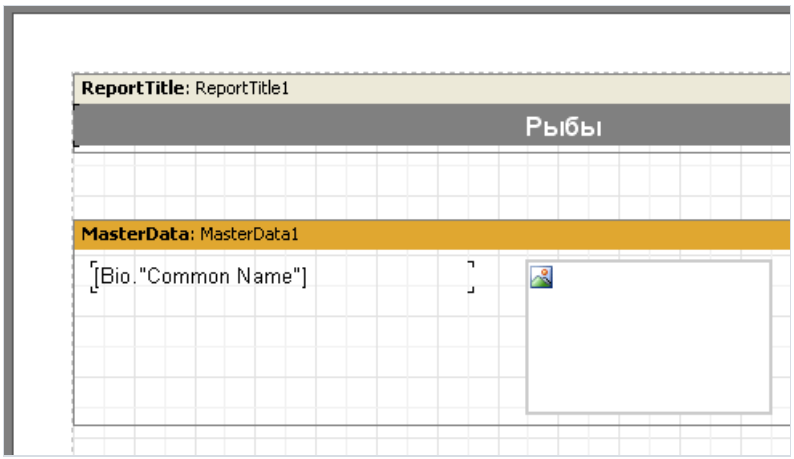
Наконец, положим на форму компонент `TfrxReport`. Зайдем в дизайнер и нажмем кнопку "Новый отчет", чтобы FastReport автоматически создал пустой шаблон. Подключим таблицу к отчету в окне "Отчет| Данные...".

Приступим к созданию формы отчета.

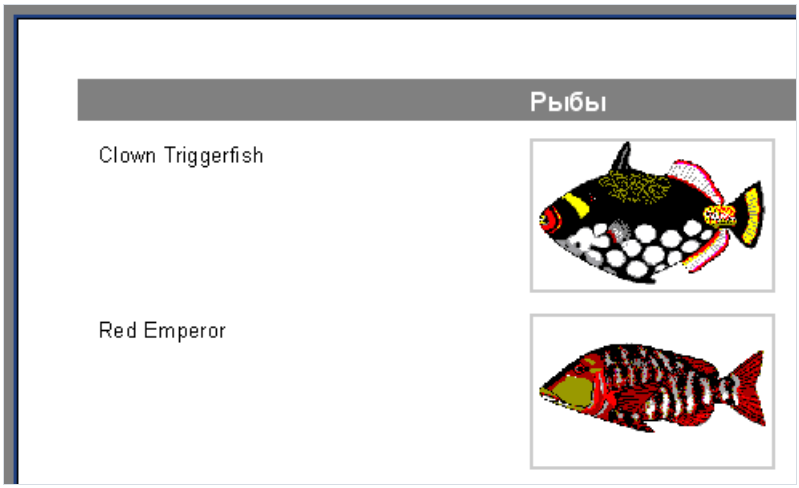
- На бэнд "Заголовок отчета" положим объект "Текст" с текстом "Рыбы".
- Бэнд "Данные 1 уровня" подключим к источнику данных (сделаем двойной щелчок на бэнде и выберем "Bio" из списка).
- Высоту бэнда увеличим до 3см, чтобы уместить картинку.
- На бэнд положим объект "Текст" и подключим его к полю "CommonName" 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.



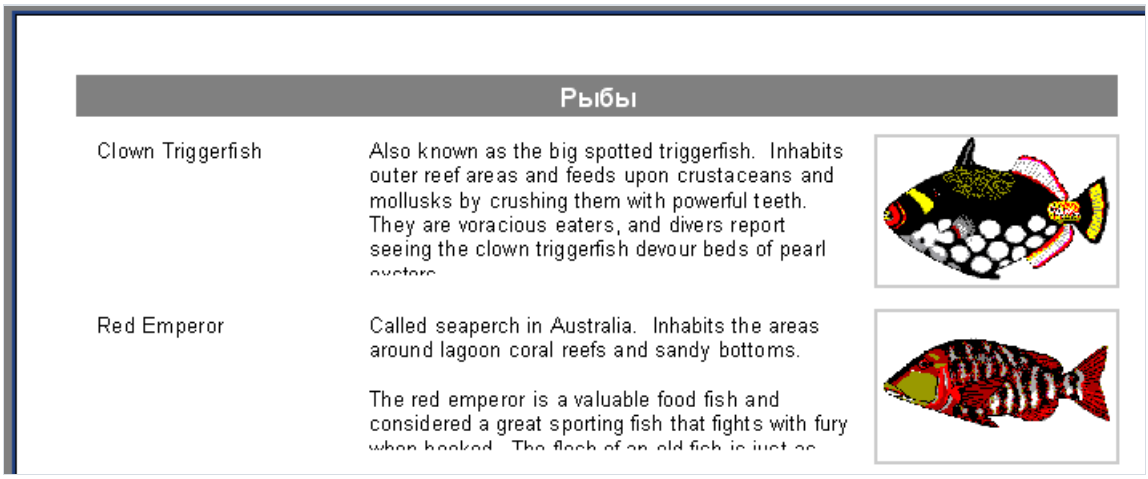
Все, отчет готов:



Отображение многострочного текста

Вернемся к предыдущему примеру с рыбами. В таблице Biolife 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:



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:

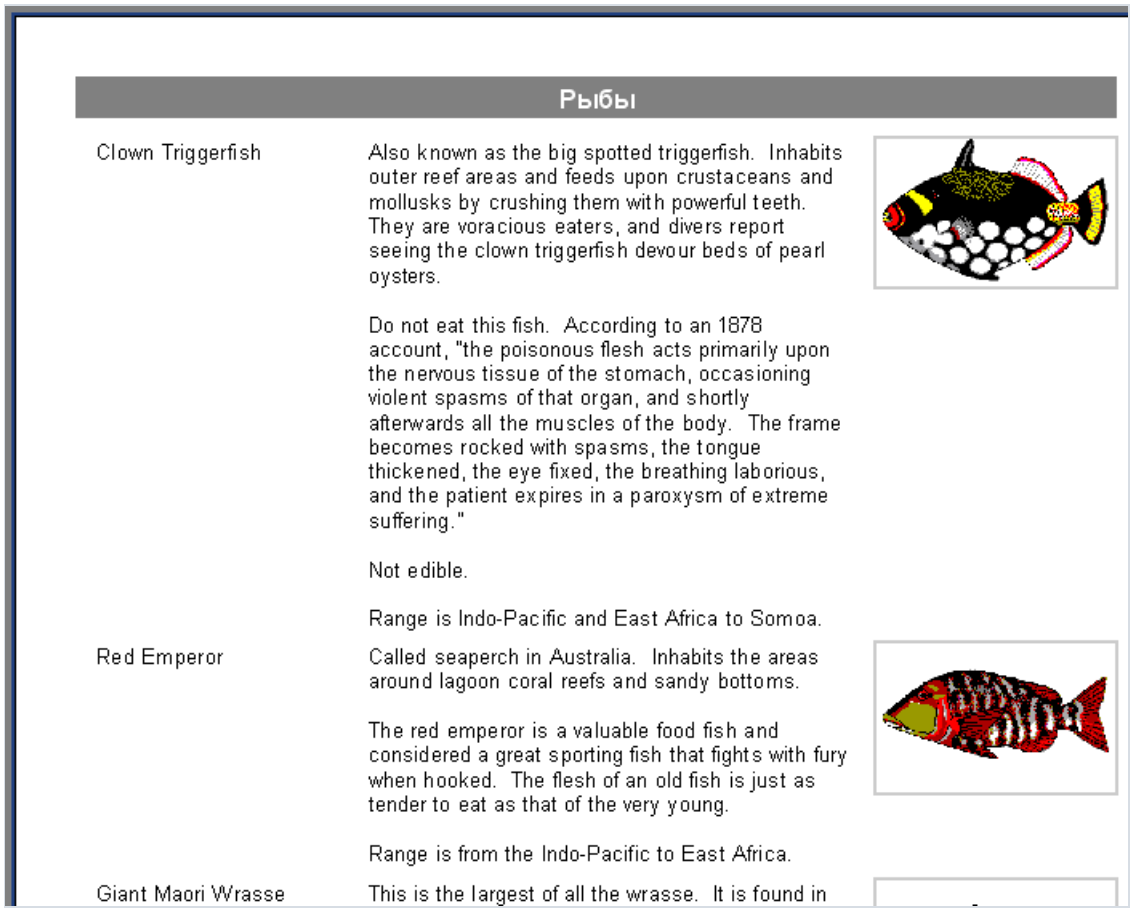
`smDontStretch` - не растягивать объект, значение по умолчанию;

`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.



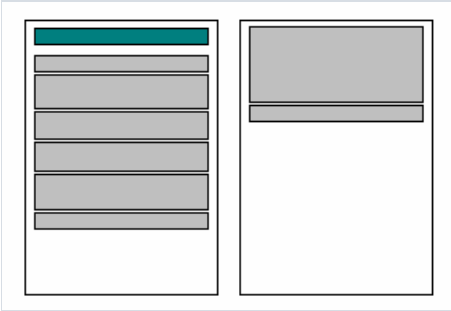
Рыбы		
Clown Triggerfish	Also known as the big spotted triggerfish. Inhabits outer reef areas and feeds upon crustaceans and mollusks by crushing them with powerful teeth. They are voracious eaters, and divers report seeing the clown triggerfish devour beds of pearl oysters. Do not eat this fish. According 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 Somoa.	
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.	
Giant Maori Wrasse	This is the largest of all the wrasse. It is found in	

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.

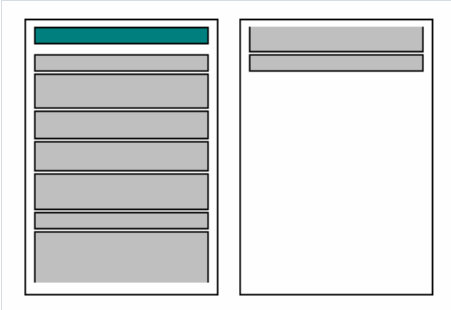
Разрыв данных

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:



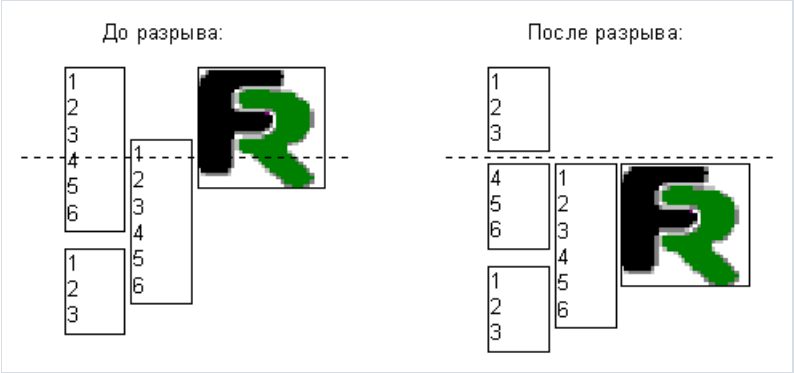
Чтобы рациональнее использовать бумагу, воспользуемся возможностью FastReport разбивать содержимое бэндов на части. Все, что нужно - это включить опцию "Разрыв" (`AllowSplit`) у бэнда "Данные 1 уровня". Мы видим, что пустого места внизу страниц отчета значительно поубавилось:



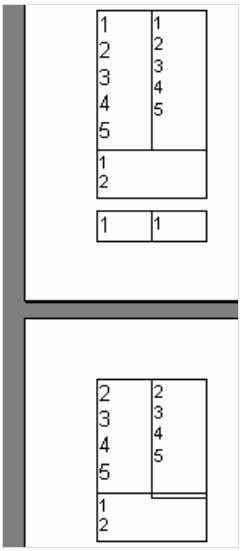
Как работает разрыв бэнда? В FastReport есть несколько объектов, которые поддерживают эту возможность. Это объекты "Текст", "Линия" и "RichEdit". Они могут быть "разорваны", остальные объекты - нет. Когда FastReport сталкивается с необходимостью выполнить разрыв, он делает следующее:

- 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:



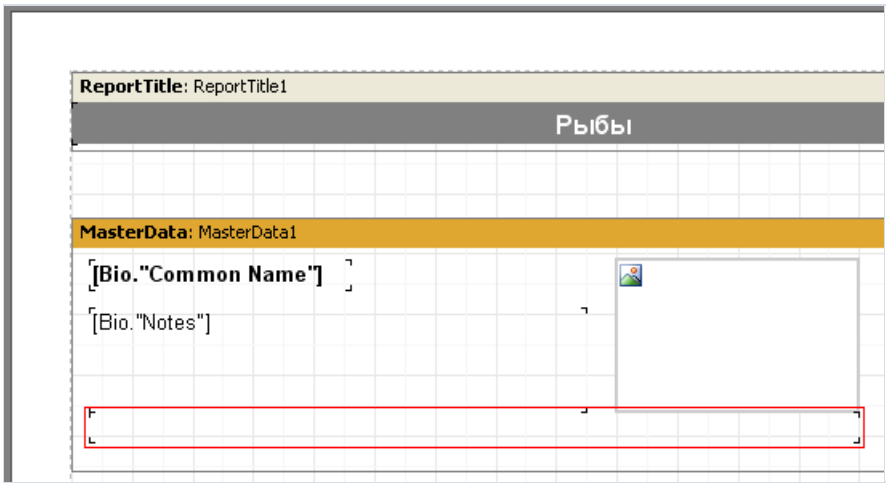
Следует отметить, что алгоритм разрыва не обеспечивает 100% качества получаемого отчета. Поэтому используйте эту опцию аккуратно, если объекты на разрываемом бэнде сгруппированы сложным образом и к тому же имеют разный размер шрифта. Вот пример того, что может получиться:



Обтекание объектов текстом

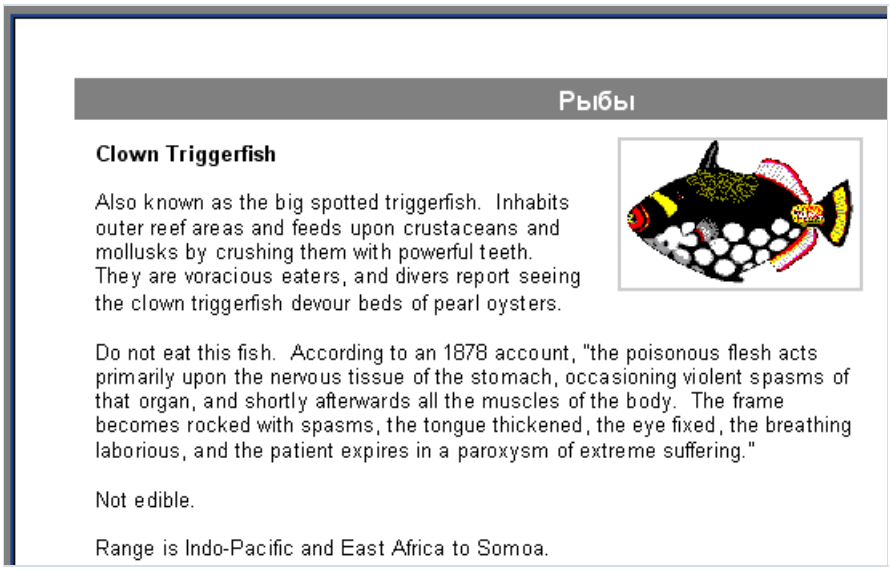
В некоторых случаях при оформлении отчета бывает необходимо сделать обтекание объектов (зачастую рисунков) текстом. Продемонстрируем такую возможность FastReport на примере с рыбами.

Добавим в отчет еще один объект "Текст" (обведен красным на рисунке) 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:



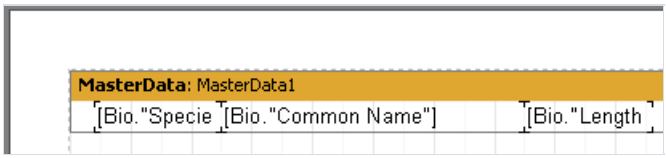
При построении отчета, когда текст не помещается в верхний объект, его не поместившаяся часть переносится в нижний. Так как объекты расположены вокруг рисунка, создается эффект обтекания рисунка текстом.

> 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 отчет следующего вида:



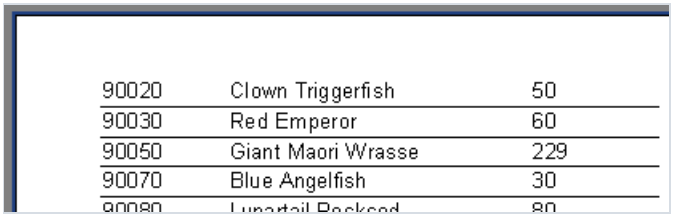
Разместим объекты на бэнде встык, а также уменьшим высоту бэнда до минимального размера.

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



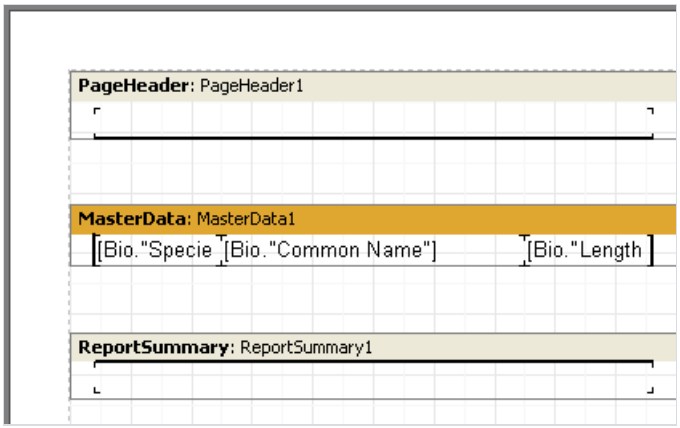
90020	Clown Triggerfish	50
90030	Red Emperor	60
90050	Giant Maori Wrasse	229
90070	Blue Angelfish	30
90080	Lunartail Rockcod	80

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

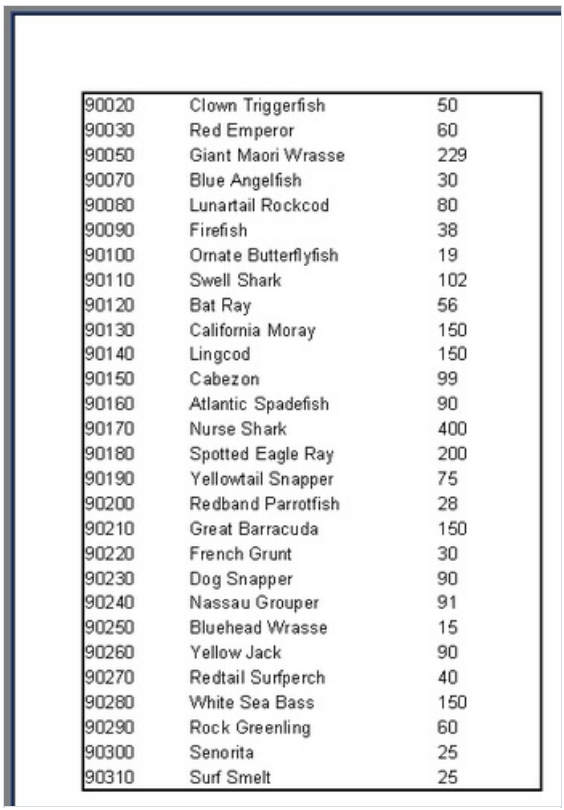


90020	Clown Triggerfish	50
90030	Red Emperor	60
90050	Giant Maori Wrasse	229
90070	Blue Angelfish	30
90080	Lunartail Rockcod	80

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



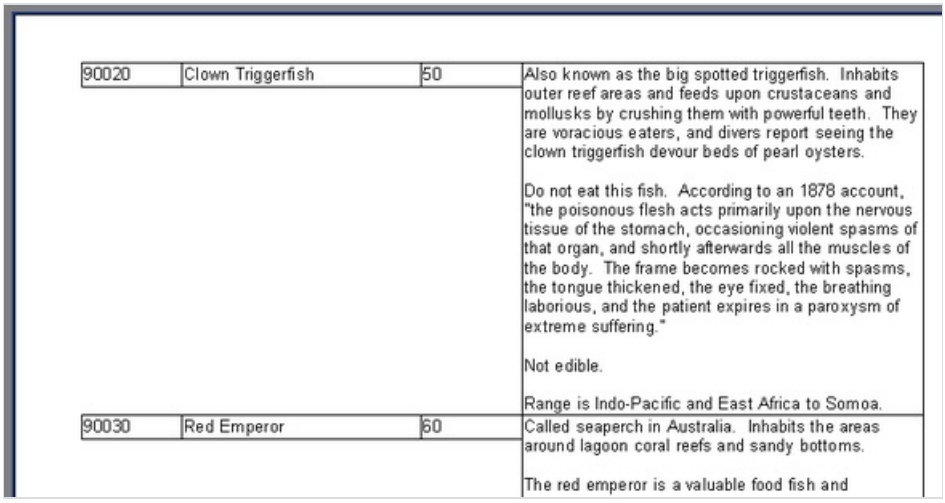
Как видно, мы добавили два объекта "Текст" и включили линии рамки у крайних объектов на дата-бэнде. В результате отчет будет выглядеть следующим образом:



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	Ornate 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 Surfperch	40
90280	White Sea Bass	150
90290	Rock Greenling	60
90300	Senorita	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 он 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 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 Somoa.
90030	Red Emperor	60	Called seaperch in Australia. Inhabits the areas around lagoon coral reefs and sandy bottoms. The red emperor is a valuable food fish and

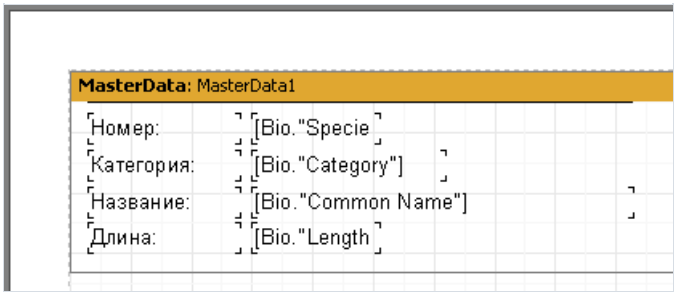
Немного не то, что нам нужно – хотелось бы, чтобы рамки соседних объектов тоже растягивались. FastReport позволяет легко решить эту проблему.

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:

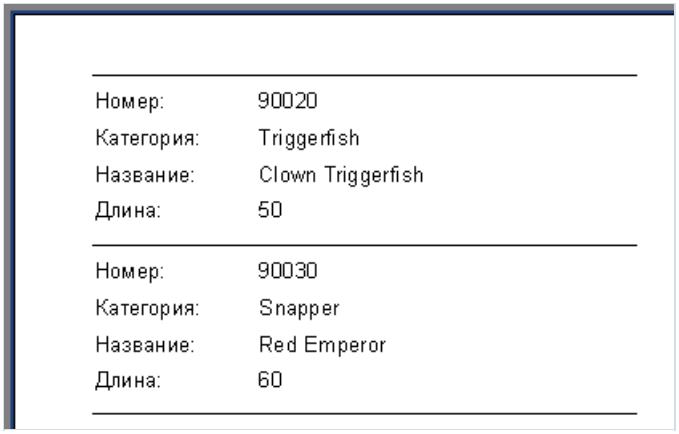
90020	Clown Triggerfish	50	Also known as the big spotted triggerfish. Inhabits outer reef areas and feeds upon crustaceans and mollusks by crushing them with powerful teeth. They are voracious eaters, and divers report seeing the clown triggerfish devour beds of pearl oysters. Do not eat this fish. According 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 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 Somoa.
90030	Red Emperor	60	Called seaperch in Australia. Inhabits the areas around lagoon coral reefs and sandy bottoms.

Печать этикеток

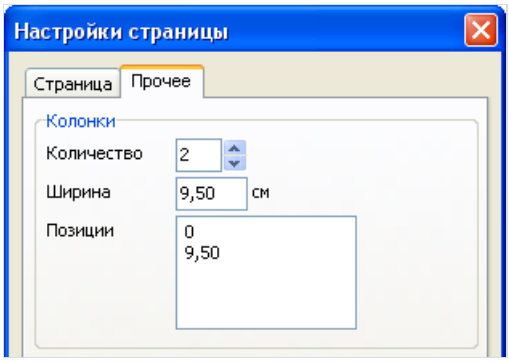
В отличие от табличных отчетов, данные в отчетах типа "этикетка" располагаются друг под другом. Рассмотрим пример подобного отчета, который выводит данные о рыбах (см. предыдущий пример) в виде этикеток. Отчет имеет следующую структуру:



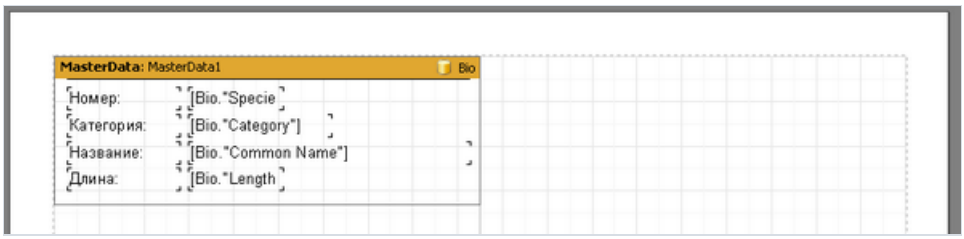
Если запустить отчет на выполнение, получим следующее:



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/Настройки 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 дизайнере тонкой вертикальной линией:



При этом печать отчета будет происходить следующим образом:

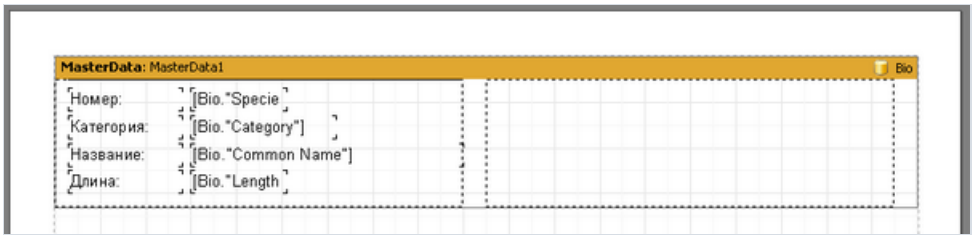
- FastReport будет выводить бэнд "Данные 1 уровня" до тех пор, пока на странице не закончится свободное место.
- После этого сформируется не новая страница, как в обычном отчете, а новая колонка на этой же странице, и вывод бэндов продолжится сверху. Но теперь все объекты будут смещены вправо на ширину 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:



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:



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

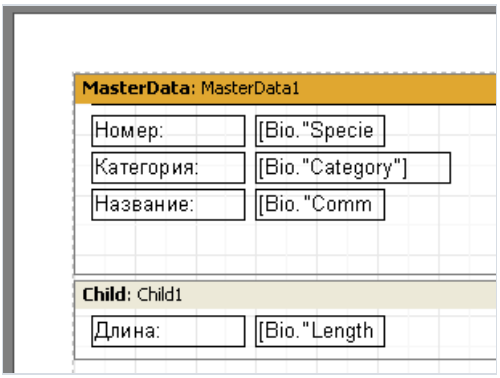
Child-бэнды

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:

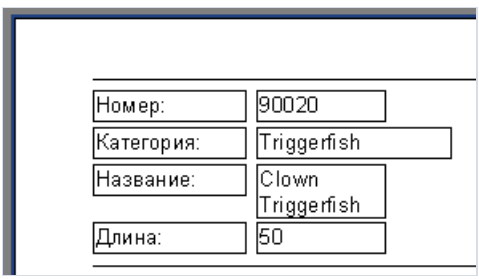


Мы видим, что объект в первом случае содержит длинный текст и поэтому он растянулся на две строки. При этом лежащий под ним объект, привязанный к полю Bio."Length (cm)", сместился ниже. Произошло это потому, что по умолчанию все объекты имеют включенное свойство "Смещение" (или `ShiftMode` = smAlways 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 – дочерний бэнд, или Child-бэнд. Он привязывается к основному бэнду и выводится после него. Модифицируем наш отчет:



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:

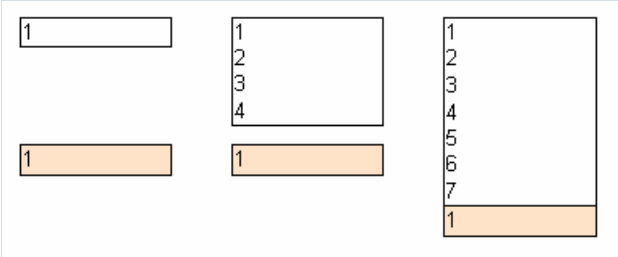


Как видно, теперь заголовок печатается там, где нужно. Для того, чтобы избежать переноса child-бэнда на следующую страницу (т.е. отрыва его от основного бэнда), установите у основного бэнда свойство "Не отрывать child" (`KeepChild` в инспекторе объектов).

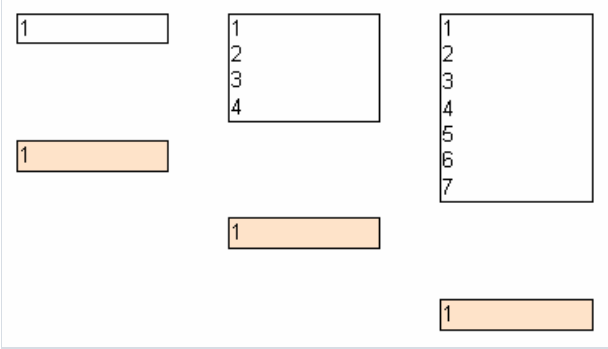
Смещение объектов

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.

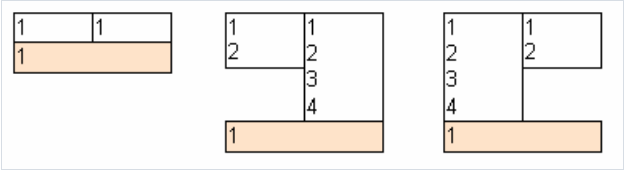
На рисунке ниже представлено три случая. Как мы видим, нижний объект со включенной опцией "Смещение при перекрытии" смещается только в последнем случае, когда в верхнем объекте много текста и он перекрывает нижний.



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 Customer и Orders. 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'

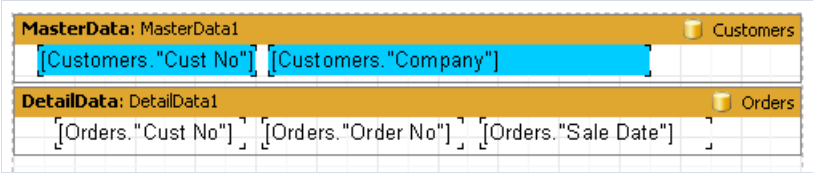
Table2:
DatabaseName = 'DBDEMOS'
TableName = 'Orders.db'

DataSource1:
DataSet = Table1

frxDBDataSet1:
DataSet = Table1
UserName = 'Customers'

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

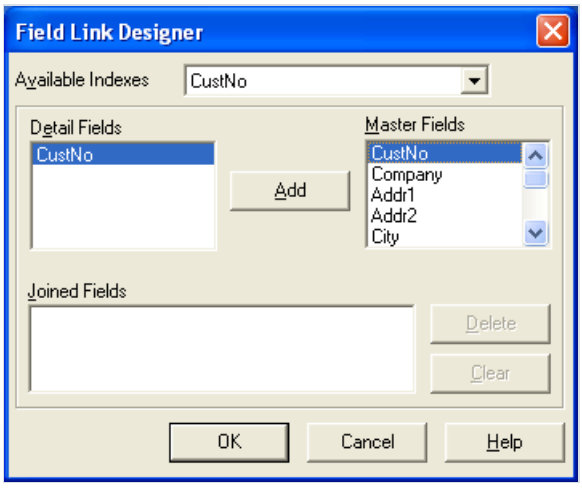
В дизайнере отчета подключим наши источники данных в окне "Отчет/Данные...". Положим на страницу бэнды "Данные 1 уровня" и "Данные 2 уровня":



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

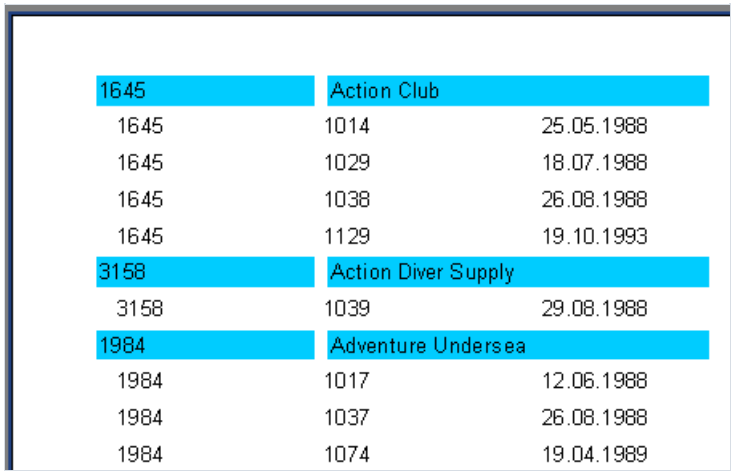
Если сейчас запустить отчет, мы увидим, что список заказов одинаковый для каждого клиента и содержит все записи из таблицы Orders. Это произошло потому, что мы не включили фильтрацию записей в таблице Orders.

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 OK.

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:

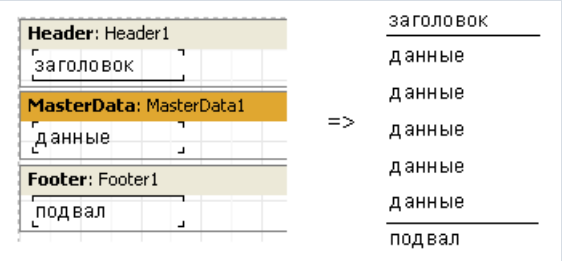


1645	Action Club	
1645	1014	25.05.1988
1645	1029	18.07.1988
1645	1038	26.08.1988
1645	1129	19.10.1993
3158	Action Diver Supply	
3158	1039	29.08.1988
1984	Adventure Undersea	
1984	1017	12.06.1988
1984	1037	26.08.1988
1984	1074	19.04.1989

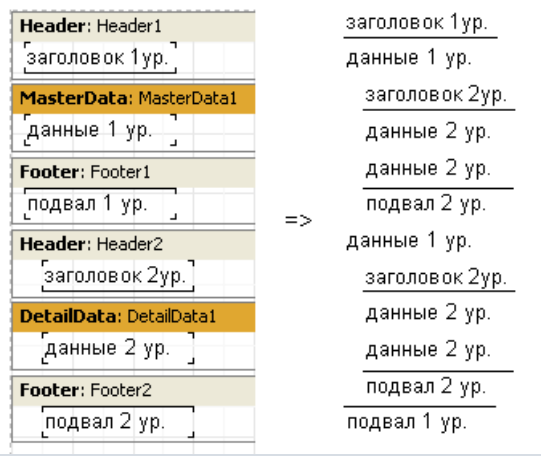
Аналогичным образом можно строить отчеты, содержащие до 6 уровней данных.

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:

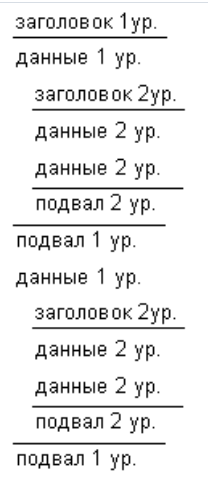


Рассмотрим печать заголовков и подвалов на примере отчета master-detail:



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.

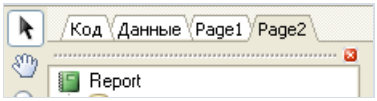
Однако, используя свойство дата-бэнда `FooterAfterEach` (или пункт контекстного меню "Footer после каждой записи"), можно вывести подвал после каждой строки данных. Это может оказаться полезным при печати отчетов типа master-detail. Предыдущий пример с включенным у бэнда "Данные 1 уровня" свойством `FooterAfterEach` будет выглядеть так:



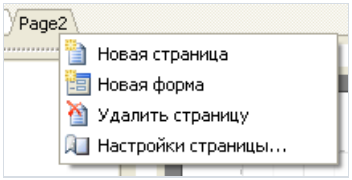
Многостраничные отчеты

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 дизайнере, 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 дизайнере появилась новая закладка:



Можно переключаться между страницами, нажав на нужную закладку мышью. Кроме того, закладки можно перетаскивать (drag&drop), тем самым 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:

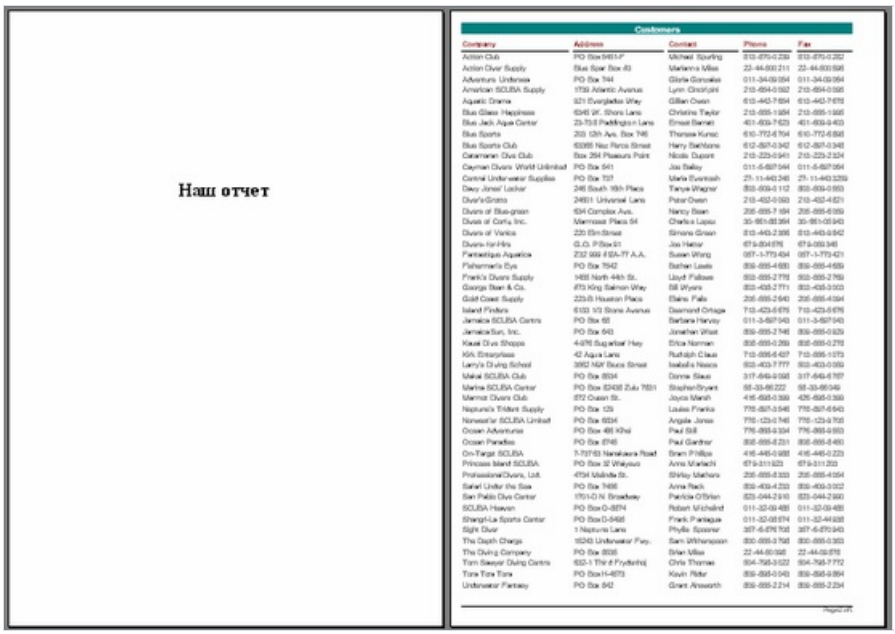


Количество страниц в отчете не ограничено. Как правило, дополнительные страницы используются для вывода титульного листа, либо в более сложных отчетах, содержащих данные из многих источников.

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:



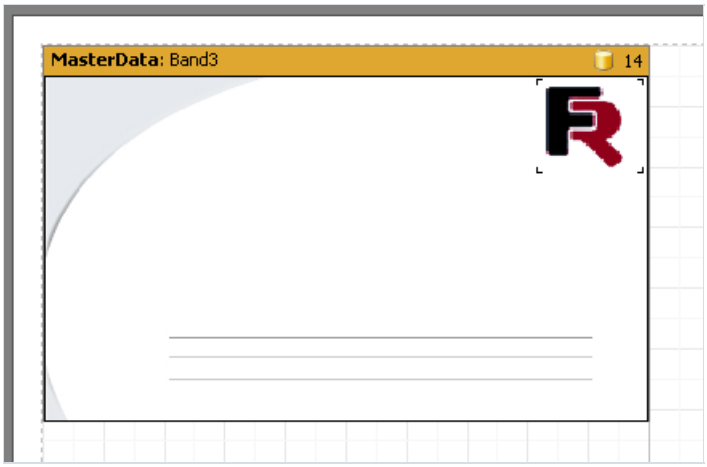
Необходимо отметить одну особенность многостраничного отчета. Если у второй страницы включить опцию "Печатать на предыдущей странице" (свойство `PrintToPreviousPage` в инспекторе объектов), то печать объектов второй страницы начнется не с нового листа, а на свободном месте предыдущей страницы. Это позволяет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

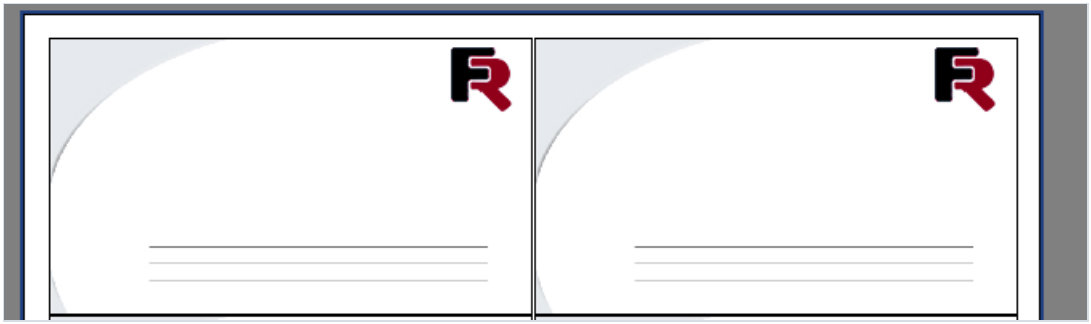
Свойства RowCount и PageCount

Иногда возникает необходимость вывести статичные данные несколько раз. В качестве примера можно рассмотреть печать незаполненных визиток или открыток. Для этого у бэндов данных есть свойство `RowCount`, а у страницы отчета `PageCount`.

Эти свойства устанавливают необходимое количество повторений бэнда/страницы не привязанных к данным.

На рисунках приведен пример такого использования, у бэнда свойство `RowCount` равно 14, что приведет к повторному выводу бэнда 14 раз.





Группировка, итоги

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
```

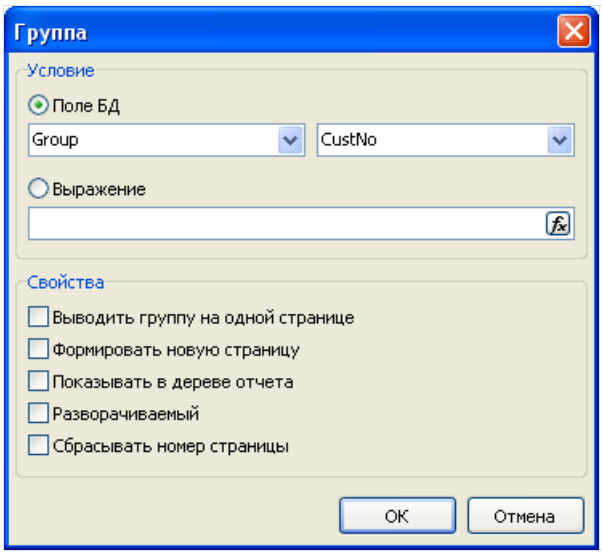
Как на основе этих данных построить многоуровневый отчет? В FastReport для этого есть специальный бэнд – "Заголовок группы". У бэнда задается условие (значение поля БД или выражение), при смене которого происходит вывод бэнда. Продемонстрируем это на примере.

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

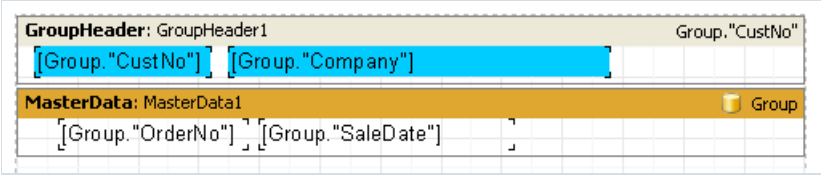
```
Query1:
DatabaseName = 'DBDEMOS'
SQL =
  select * from customer, orders
  where orders.CustNo = customer.CustNo
  order by customer.CustNo

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

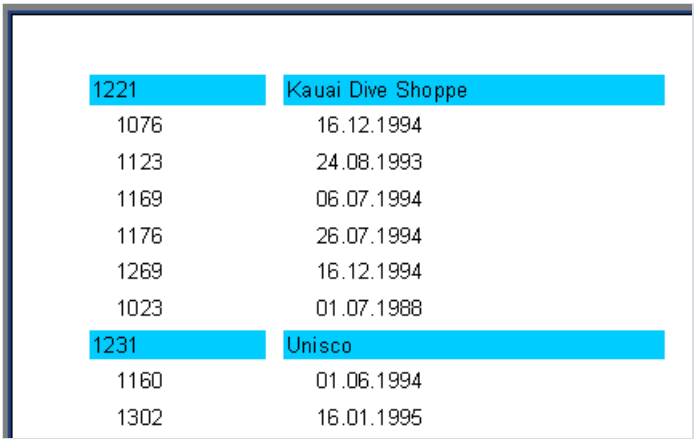
Зайдем в дизайнер и подключим наш источник данных к отчету. Добавим в отчет два бэнда: "Заголовок группы" и "Данные 1 уровня". В редакторе бэнда "Заголовок группы" укажем условие – поле данных Group.CustNo:



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



При запуске получится отчет следующего вида:



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:

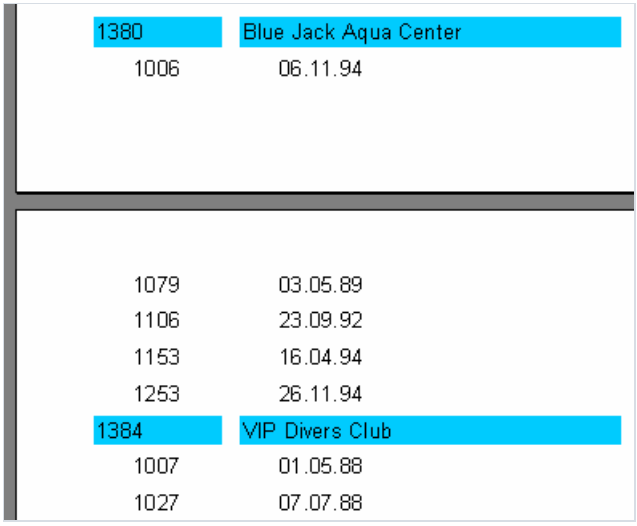
- требуется только одна таблица (запрос) для всего отчета;
- число уровней вложенности данных не ограничено;

- возможность дополнительной сортировки данных;
- более рациональное использование ресурсов СУБД – запрос возвращает только те данные, которые должны быть напечатаны, отсутствует фильтрация данных.

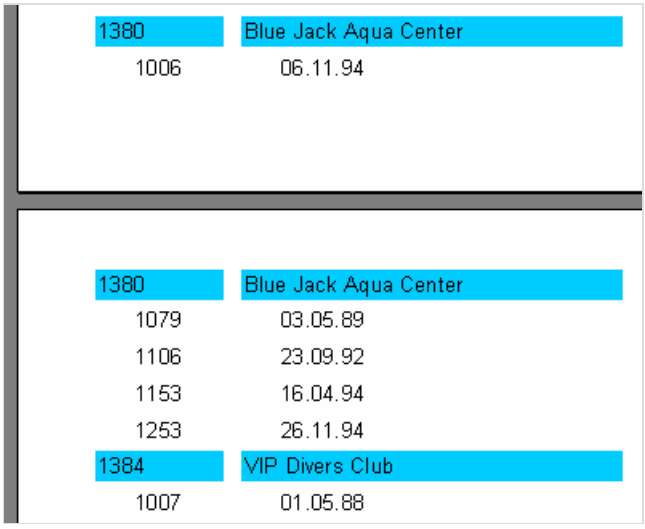
Единственный минус – необходимость написания запросов на языке SQL. Впрочем, знание основ SQL является обязательным для программиста, работающего с базами данных.

Другие особенности групп

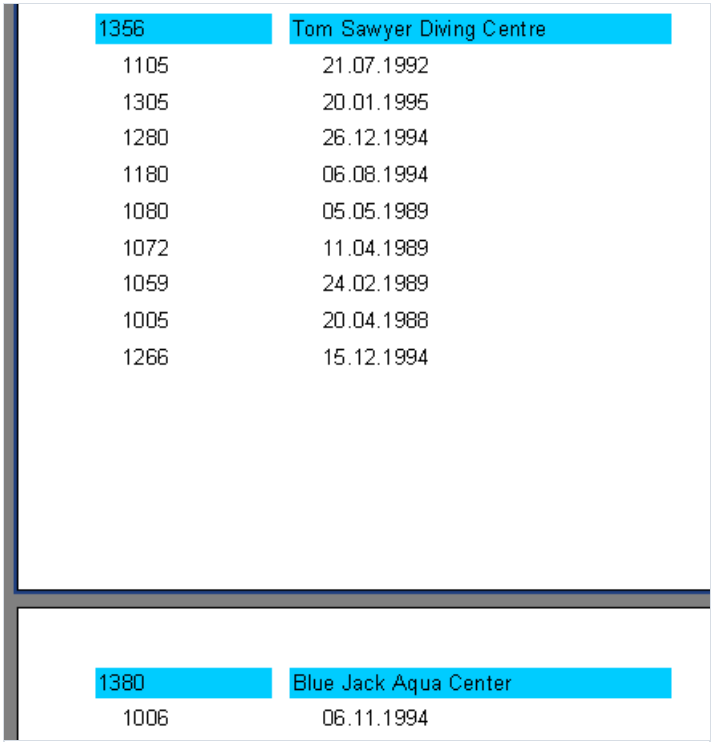
Обратим внимание на то, как группа переносится на следующую страницу:



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 "Выводить на новой странице" (или свойство `ReprintOnNewPage` в инспекторе объектов). При этом отчет будет выглядеть так:



Есть еще способ, позволяющий избежать разрыва группы. Для этого надо включить свойство заголовка группы "Держать вместе" (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:



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

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

Сброс нумерации страниц

У группы есть свойство `ResetPageNumbers` (пункт "Сбрасывать номер страницы" в контекстном меню), которое позволяет сбрасывать нумерацию страниц при печати группы. Для чего это нужно?

Допустим, вы сделали отчет с группировкой. В заголовке группы - наименование клиента, тело группы - заказы, сделанные клиентом. Теперь отпечатанный отчет нужно раздать разным клиентам - каждому свои листы. К сожалению, нумерация страниц в таком отчете сквозная, и какой-нибудь клиент получит листы с номерами, например, 50, 51, 52 (а где же предыдущие 49 страниц, спросит он?)

Чтобы избежать подобной ситуации, надо пронумеровать листы каждого клиента отдельно. То есть, в пределах одного отчета вы получите отдельную нумерацию страниц для каждой группы.

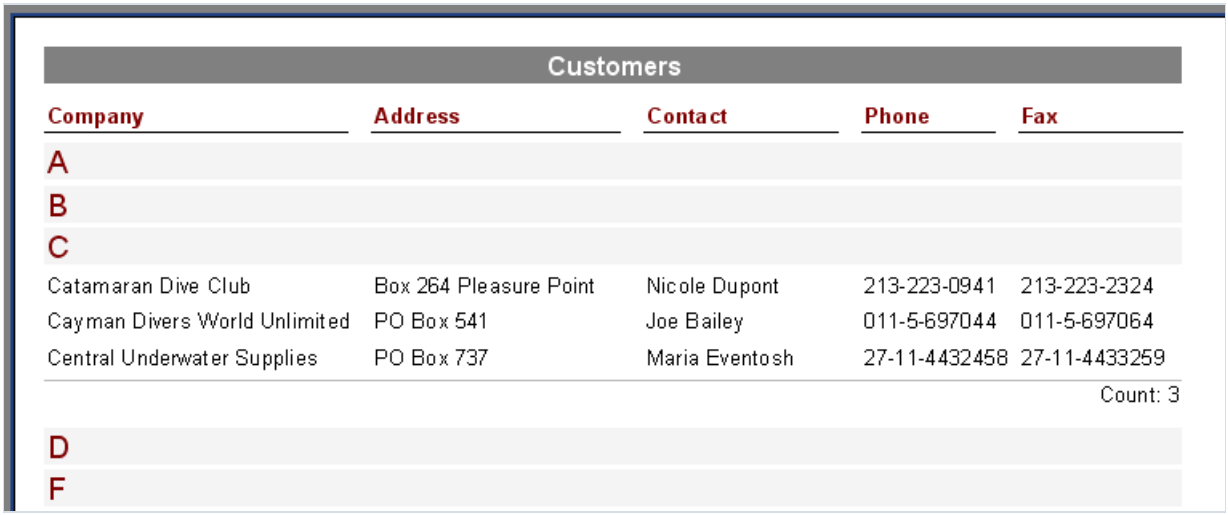
> Обратите внимание - при установке свойства `ResetPageNumbers` надо также включить свойство `StartNewPage` ("Формировать новую страницу"), чтобы каждая группа печаталась начиная с новой страницы.

Вывести номер страницы и общее число страниц в группе можно с помощью переменных `[Page]`, `[TotalPages]`, помещенных в объект "Текст".

Разворачиваемые (drill-down) группы

У заголовка группы есть свойство `DrillDown` (пункт "Разворачиваемый" в контекстном меню). Включение этого свойства позволяет сделать группу интерактивной. Это означает, что группа будет реагировать на щелчок мышью в окне предварительного просмотра. Щелкая мышью на заголовке группы, ее можно развернуть (показать все ее записи) или свернуть, оставив только заголовок и, при необходимости, подвал (это настраивается свойством `ShowFooterIfDrillDown`).

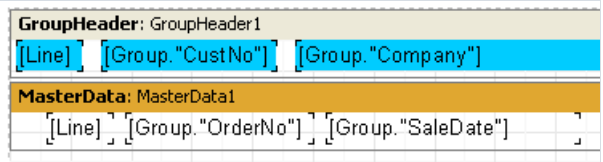
Например, так выглядит группа с одним развернутым заголовком:



Вы можете указать, надо ли выводить все группы свернутыми или развернутыми при запуске отчета. По умолчанию группы свернуты, это контролируется свойством `ExpandDrillDown`. Если группы необходимо развернуть, установите это свойство в True. Также вы можете развернуть или свернуть все группы, выбрав пункты "Развернуть все" или "Свернуть все" из контекстного меню в окне предварительного просмотра.

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").



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

1	1221	Kauai Dive Shoppe
1	1076	16.12.1994
2	1123	24.08.1993
3	1169	06.07.1994
4	1176	26.07.1994
5	1269	16.12.1994
6	1023	01.07.1988
2	1231	Unisco
1	1160	01.06.1994
2	1302	16.01.1995
3	1278	23.12.1994
4	1202	06.10.1994

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

1	1221	Kauai Dive Shoppe
1	1076	16.12.1994
2	1123	24.08.1993
3	1169	06.07.1994
4	1176	26.07.1994
5	1269	16.12.1994
6	1023	01.07.1988
2	1231	Unisco
7	1160	01.06.1994
8	1302	16.01.1995
9	1278	23.12.1994

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:

Функция	Описание
`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.

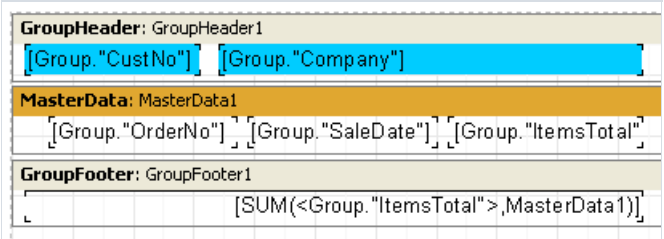
Функция `COUNT` имеет следующий синтаксис:

```
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:



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	
1076	16.12.1994	17781
1123	24.08.1993	13945
1169	06.07.1994	9471,95
1176	26.07.1994	4178,85
1269	16.12.1994	1400
1023	01.07.1988	4674
		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)]
```

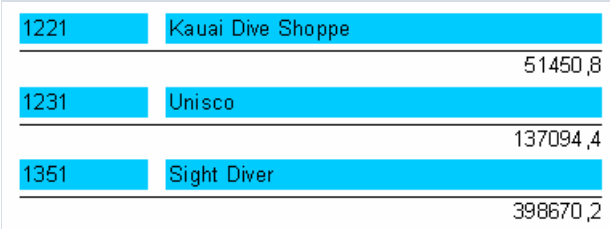
Это даст нам отчет следующего вида:

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)]
```

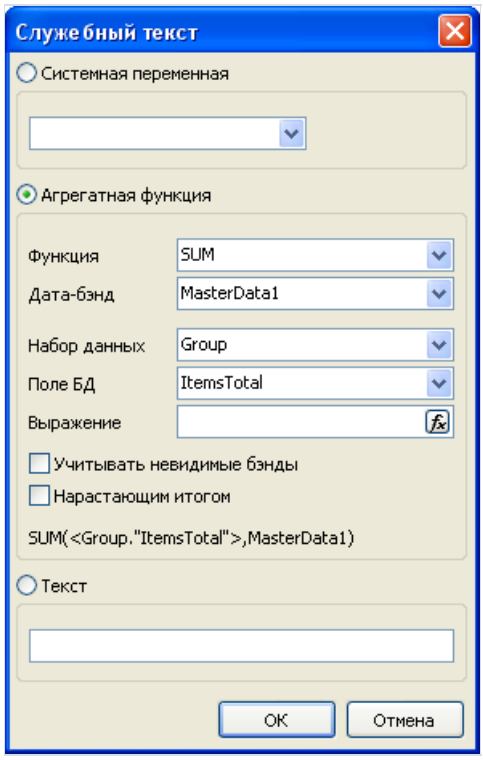
Значение 3 – это битовая комбинация 1 и 2, что означает, что нам надо учитывать невидимые бэнды и не сбрасывать сумму. В итоге получится следующее:



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.

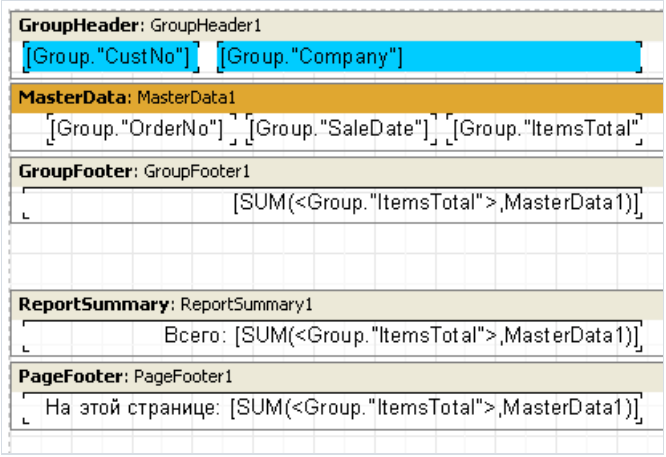


В редакторе надо последовательно выбрать тип функции, дата-бэнд, по которому она будет считаться, и поле БД или выражение, значение которого будет вычисляться. Также можно отметить флажки "Учитывать невидимые бэнды" и "Нарастающие итоги".

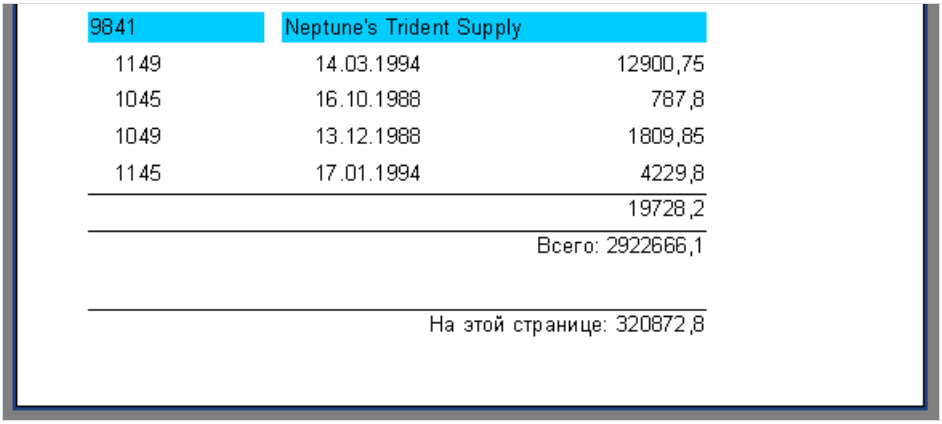
Второй способ – использовать объект "Текст" и кнопку  в его редакторе. При этом открывается дополнительное окно, аналогичное рассмотренному редактору объекта "Системный текст". При нажатии кнопки OK в текст объекта вставляется вызов агрегатной функции.

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.



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



Форматирование, выделение

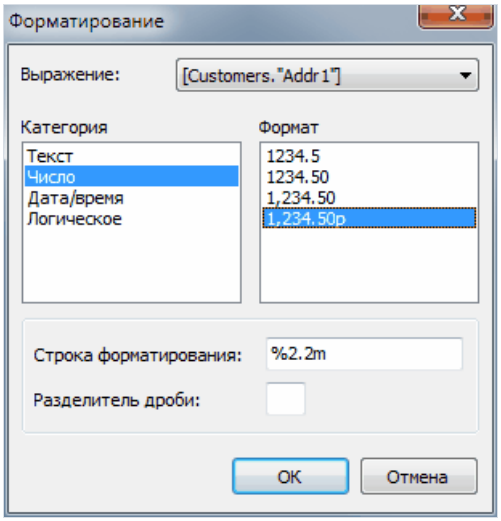
Форматирование значений

Обратим внимание на одну особенность при использовании агрегатных функций: они возвращают неформатированное числовое значение.

1176	26.07.94	4 178,85р.
1269	16.12.94	1 400,00р.
		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. Если оставить разделитель пустым, то будет использоваться разделитель из текущих региональных настроек системы.).

После нажатия клавиши OK 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.
		51 450,80р.

Обратите внимание на выпадающий список с выражениями в верхней части окна. Если в тексте объекта есть несколько выражений, можно указать формат для каждого из них.

Форматирование по месту

Форматирование "по месту" позволяет указать строку формата сразу после выражения.

> Этот способ применялся в ранних версиях FastReport для форматирования нескольких выражений, содержащихся в одном объекте "Текст". В версии FastReport 5 этот способ устарел, т.к. в редакторе формата можно указать формат для каждого выражения.

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)]
```

Для корректного вывода значений надо 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	$1 400,00
		Total: $51 450,80
		Number: 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:

`#nСтрокаФорматирования` – числовой формат

`#dСтрокаФорматирования` – формат даты/времени

`#bЛожь,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

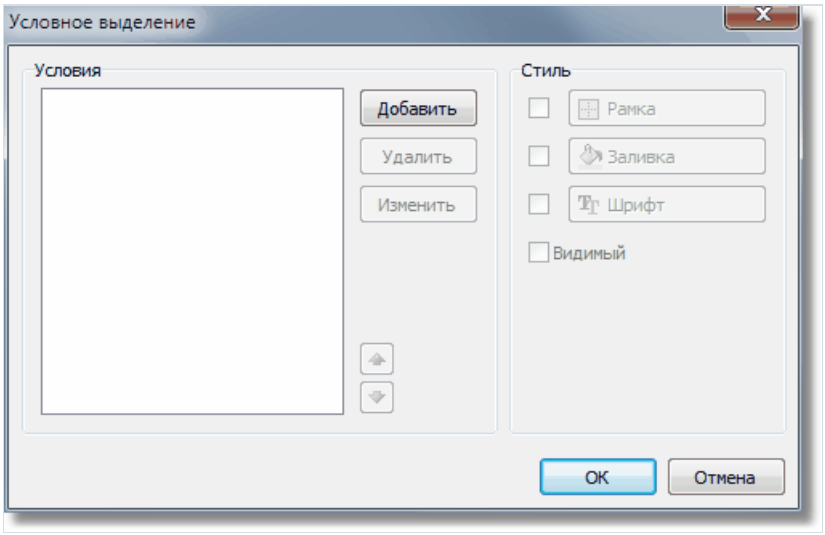
В строке для числового формата допускается указывать вместо точки запятую или тире, тогда этот символ будет использован как разделитель целой и дробной частей числа. Использование других разделителей не допускается.

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

Условное выделение

В объекте "Текст" предусмотрена возможность смены внешнего вида объекта в зависимости от заданных условий. Например, объект можно выделить красным цветом, если он содержит отрицательное значение.

Эта возможность называется "условное выделение". Для ее настройки выберите объект и нажмите кнопку  на панели инструментов "Текст". Вы увидите следующее диалоговое окно:



Здесь можно определить одно или несколько условий и задать стиль для каждого условия. Стиль может содержать одно или несколько свойств:

- рамка;
- заливка;
- параметры шрифта;
- видимость объекта.

Вы можете указать, какие свойства необходимо менять при срабатывании условия. Для этого используйте флажки в правой части окна. По умолчанию новый стиль имеет одну настройку – цвет текста.

Для того чтобы создать новое условие, нажмите кнопку "Добавить". Вы увидите редактор выражения. Здесь можно написать любое выражение, которое возвращает логический результат (да/нет). В большинстве случаев в выражении участвует текущее печатаемое значение, которое доступно через переменную `Value`.

Рассмотрим следующий пример: у нас есть объект "Текст", в котором печатается остаток товара на складе:

```
[Products."UnitsInStock"]
```

Мы хотим подсветить объект красным цветом, если количество товара = 0. Для этого создадим следующее условие:

```
Value = 0
```

Внимание: если в качестве скриптового языка выбран C++ Script (см. подробнее в главе Скрипт), условие должно быть написано на C++ Script:

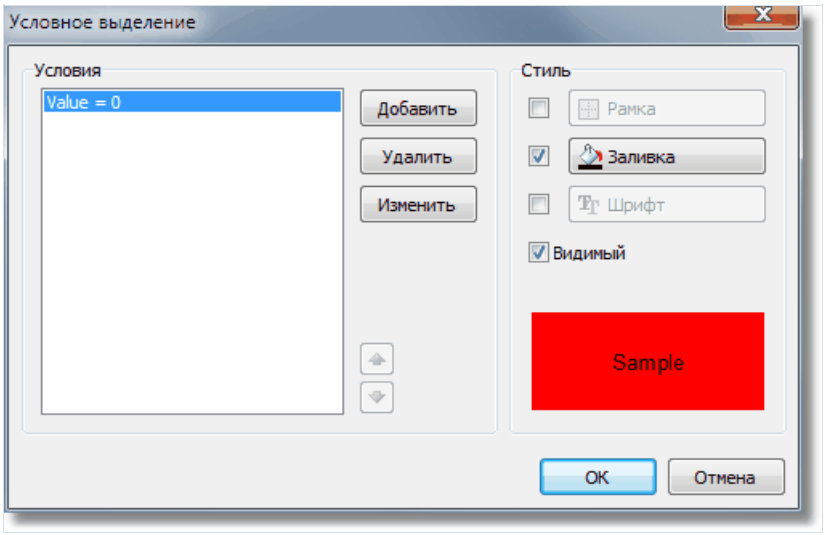
```
Value == 0
```

В данном случае мы использовали переменную `Value`, которая содержит печатаемое значение. Если в объекте есть несколько выражений, эта переменная будет содержать значение последнего выражения.

Вместо `Value` можно использовать поле из источника данных, например:

```
<Products."UnitsInStock"> = 0
```

Настроим стиль для данного условия так, чтобы использовалась только заливка, и укажем цвет заливки – красный:



При печати объекты, содержащие нулевое значение, будут красными. Теперь усложним наш пример, добавив еще одно условие. Если остаток товара меньше 10, он должен быть напечатан желтым цветом. Для этого откроем редактор условий и нажмем кнопку "Добавить". Второе условие будет выглядеть так:

```
Value < 10
```

В случае, когда указано несколько условий, FastReport проверяет все условия, начиная с первого. Если какое-то условие выполняется, FastReport применяет его стиль к объекту, и процесс завершается. Здесь важно расставить условия в правильном порядке. Так, порядок, который мы рассмотрели в этом примере, правильный:

1. Value = 0
2. Value < 10

Если условия поменять местами, то выделение будет работать неправильно.

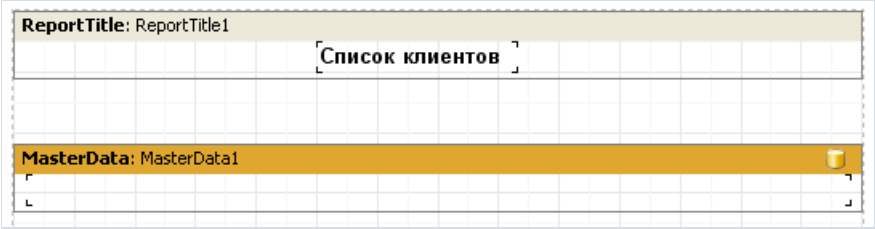
1. Value < 10
2. Value = 0

В данном случае условие "Value = 0" выполняться не будет, потому что при нулевом значении сработает первое условие. Для того чтобы поменять порядок условий, используйте кнопки и в редакторе условий.

Выделение строк через одну

С помощью условного выделения можно легко придать отчету более современный вид, "раскрасив" каждую нечетную 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:



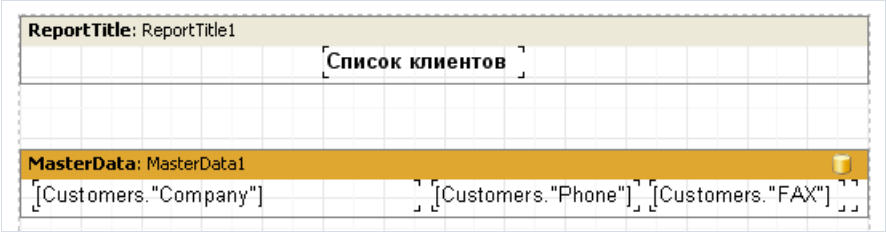
Этот объект будет выполнять роль подложки, которая будет менять цвет в зависимости от номера строки данных. Выделим объект и установим в редакторе выделения следующее условие:

```
<Line> mod 2 = 1
```

Внимание: если в качестве скриптового языка выбран C++Script (см. подробнее в главе Скрипт), условие должно быть написано на C++Script:

```
<Line> % 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		
Action Club	813-870-0239	813-870-0282
Action Diver Supply	22-44-500211	22-44-500596
Adventure Undersea	011-34-09054	011-34-09064
American SCUBA Supply	213-654-0092	213-654-0095
Aquatic Drama	613-442-7654	613-442-7678
Blue Glass Happiness	213-555-1984	213-555-1995

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 удается. В таком случае можно использовать объект "Вложенный отчет"  .

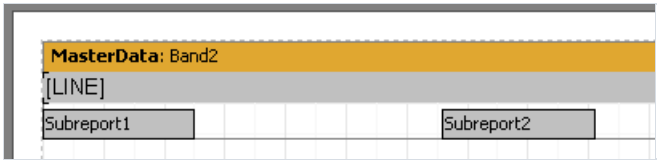
Вставив такой объект в отчет, мы увидим, что FastReport автоматически добавил новую страницу, связанную с этим объектом. Вложенный отчет по своей структуре очень похож на многостраничный. Единственное отличие – вложенный отчет выводится в заданном месте основного отчета, а не после него. При формировании 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".

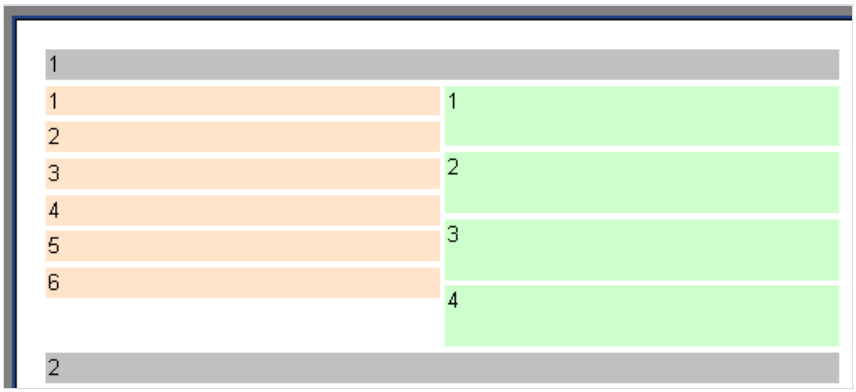
Следует отметить, что способность FastReport строить многократно вложенные отчеты позволяет неограниченно наращивать уровень вложенности данных. Напомним, что без использования объекта "Вложенный отчет" число уровней вложенности в FastReport ограничено – не более шести.

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:



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.

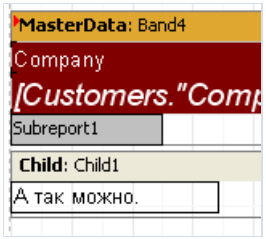
Нельзя класть объекты ниже объекта "Вложенный отчет":



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 дочерний (child) бэнд:

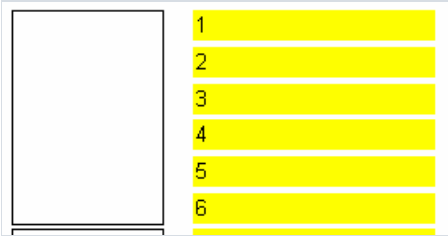


Это же касается случая, когда нужно вывести несколько вложенных отчетов друг под другом.

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:



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

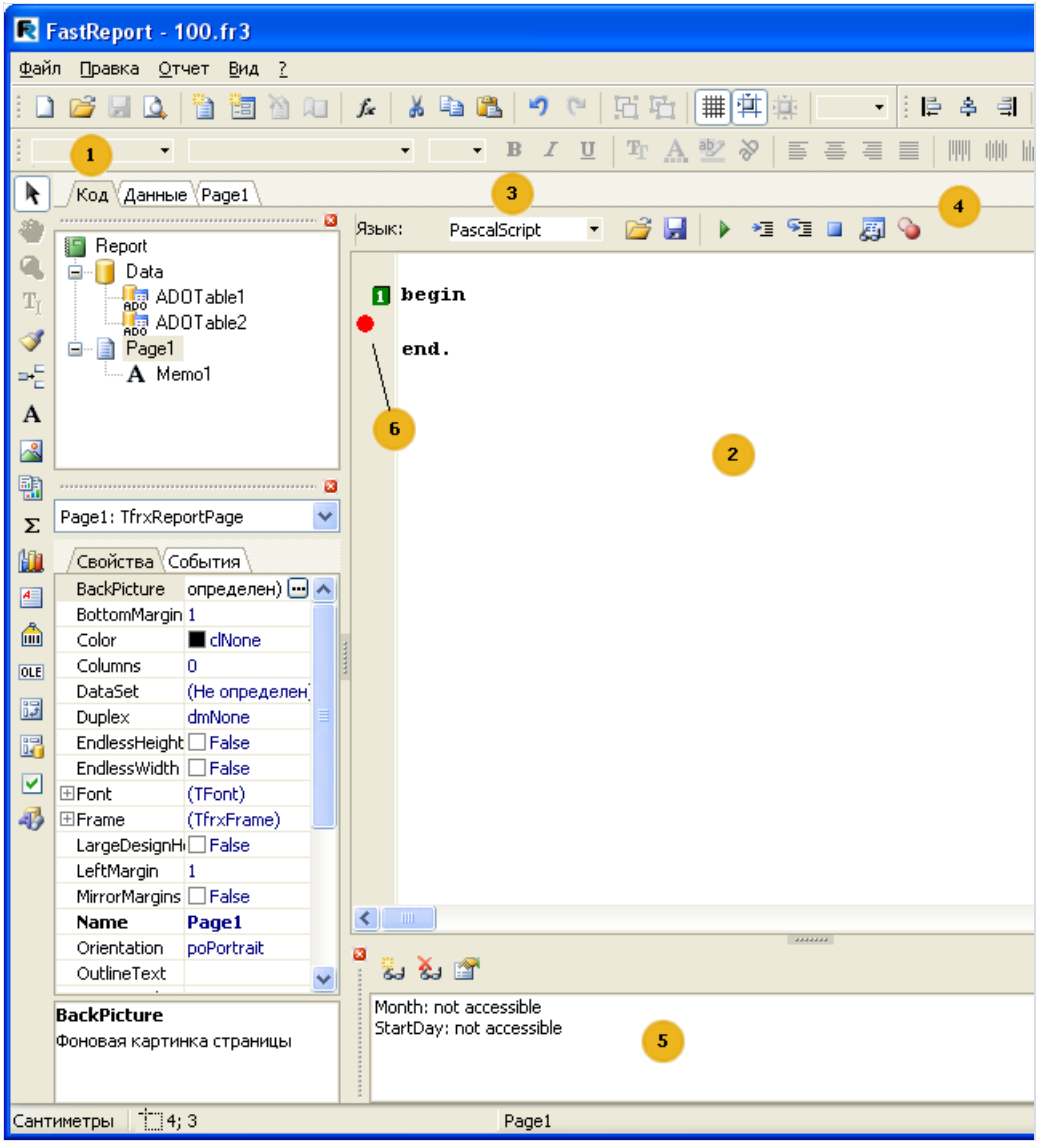
Возможности скриптового движка FastScript следующие:

- 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 дизайнере FastReport, который содержит редактор скриптов с подстветкой 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 скрипта;

4 – панель управления отладчика:

- запуск отчета на выполнение в режиме отладки;

- запуск отчета до строки, на которой стоит курсор (Run to cursor);

- выполнение очередной строки кода (Step into);

- прерывание работы скрипта;

- просмотр значений выражений (Evaluate);

 - установка/снятие точки останова.

5 - окно Watches для наблюдения за переменными;

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

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 установенную закладку
Ctrl+F	Поиск строки

Клавиша	Значение
Ctrl+R	Замена строки
F3	Повторный поиск/замена с позиции курсора
F4	Запуск отчета до строки, на которой стоит курсор (Run to cursor)
F5	Установка точки прерывания (Toggle breakpoint)
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)
Ctrl+пробел	Показывает выпадающий список с методами и свойствами объекта, имя которого набрано
Ctrl+Shift+Delete	Удаляет слово перед курсором целиком
Ctrl+Shift+Backspace	Удаляет слово после курсора целиком

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 языках PascalScript, C++Script. При создании нового отчета PascalScript 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.

Зайдем в дизайнер и нажмем кнопку "Новый отчет", чтобы FastReport автоматически создал пустой шаблон. Переключимся на закладку "Код" и напишем следующий скрипт:

PascalScript:

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

C++ Script:

```
{
  ShowMessage("Hello, World!");
}
```

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:

PascalScript:

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

C++Script:

```
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.

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

PascalScript:

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

C++Script:

```
{
  MyTextObject.Color = clRed
}
```

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:

PascalScript:

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

C++ Script:

```
if (<my variable> == 10) { ... }
```

Альтернативный вариант – использование функции `Get`:

PascalScript:

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

C++ Script:

```
if (Get("my variable") == 10) { ... }
```

Изменение значения такой переменной возможно только с помощью процедуры `Set`:

PascalScript:

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

C++ Script:

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

Стоит отметить, что для присвоения строкового значения нужно использовать дополнительные кавычки:

PascalScript:

```
Set('my variable', '''' + 'Строка' + '''');
```

C++ Script:

```
Set("my variable", "\"Строка\"");
```

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

PascalScript:

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

C++ Script:

```
if (<Page#> == 1) { ... }
```

Обращение к полям БД

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

PascalScript:

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

C++ Script:

```
if (<Table1."Field1"> == Null) { ... }
```

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

Использование агрегатных функций в скрипте

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:

PascalScript:

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

C++ Script:

```
string MyVariable;
{
  MyVariable = "Hello!";
}
```

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

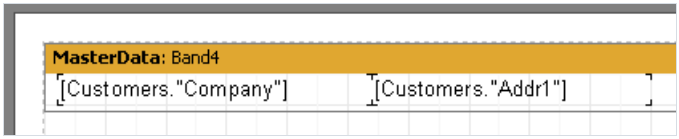
Имя переменной должно быть уникальным, т.е. не должно совпадать с именами объектов отчета, стандартных функций, констант. При любой ошибке в скрипте на экран будет выведено сообщение и отчет строиться не будет.

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 вызывается главная процедура скрипта. После этого начинается 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 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

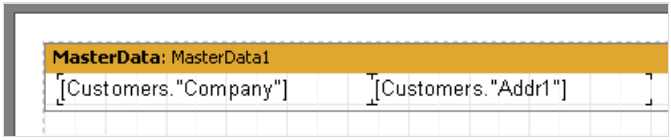
Продемонстрируем вышесказанное на практике. Создадим отчет – список клиентов, в котором будут представлены только компании, название которых начинается с буквы "A".

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

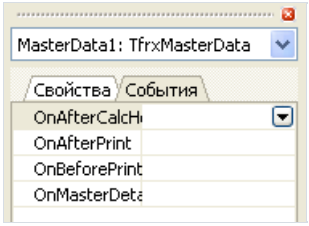
```
Table1:
DatabaseName = 'DBDEMOS'
TableName = 'customer.db'

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

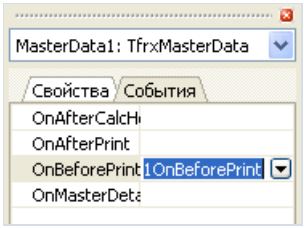
Зайдем в редактор отчета и создадим отчет следующего вида:



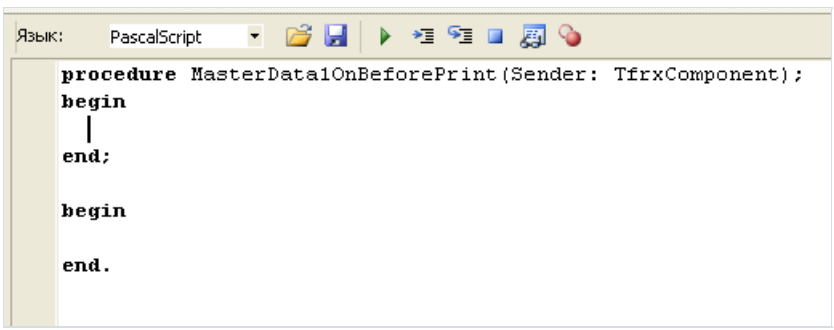
Выделим дата-бэнд и переключимся на закладку "События" в инспекторе объектов:



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":



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

PascalScript:

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

C++Script:

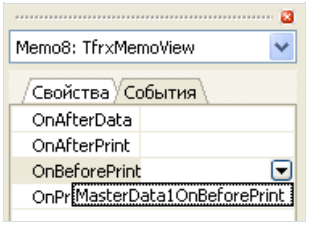
```
if (Copy(<Customers."Company">, 1, 1) == "A")
  MasterData1.Visible = true;
else
  MasterData1.Visible = false;
```

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.

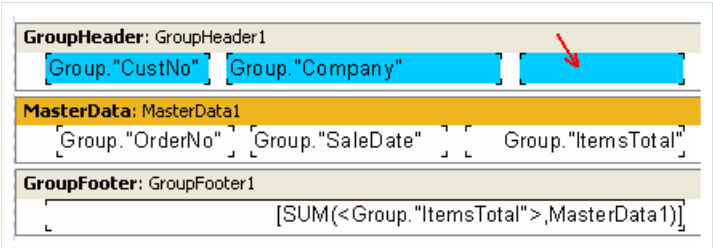
Продемонстрируем, как решить эту задачу двумя способами. Для начала создадим новый проект в Delphi, на форму положим компоненты `TQuery`, `TfrxReport`, `TfrxDBDataSet`. Настроим их следующим образом:

```
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 объекты следующим образом:



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:

PascalScript:

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

procedure frxReport1OnStartReport(Sender: TfrxComponent);
begin
  List := TStringList.Create;
end;

procedure frxReport1OnStopReport(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
  if not Engine.FinalPass then
    List.Add(FloatToStr(SUM(<Group."ItemsTotal">,MasterData1)));
  Inc(i);
end;

begin
end.
```

C++ Script:

```
TStringList List;
int i;

void frxReport1OnStartReport(TfrxComponent Sender)
{
  List = TStringList.Create();
}

void frxReport1OnStopReport(TfrxComponent Sender)
{
  List.Free();
}

void Page1OnBeforePrint(TfrxComponent Sender)
{
  i = 0;
}

void GroupHeader1OnBeforePrint(TfrxComponent Sender)
{
  if (Engine.FinalPass)
    Memo8.Text = "Sum: " + List[i];
}

void GroupFooter1OnBeforePrint(TfrxComponent Sender)
{
  List.Add(FloatToStr(SUM(<Group."ItemsTotal">,MasterData1)));
  i++;
}

{
}
```

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	4 674,00р.
1076	16.12.94	17 781,00р.
1123	24.08.93	13 945,00р.
1169	06.07.94	9 471,95р.
1176	26.07.94	4 178,85р.
1269	16.12.94	1 400,00р.
		51 450,80р.

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:

PascalScript:

```
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.
```

C++ Script:

```
void GroupHeader1OnBeforePrint(TfrxComponent Sender)
{
  if (Engine.FinalPass)
    Memo8.Text = "Sum:" + Get(<Group."CustNo">);
}

void GroupFooter1OnBeforePrint(TfrxComponent Sender)
{
  Set(<Group."CustNo">,
    FloatToStr(SUM(<Group."ItemsTotal">,MasterData1)));
}

{
}
```

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`:

PascalScript:

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

C++ Script:

```
if (Value > 3000)
  Memo1.Color = clRed;
```

вместо того, чтобы писать что-то вроде:

PascalScript:

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

C++ Script:

```
if (<Table1."Field1"> > 3000)
  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.

Событие OnAfterData отлично подходит для вычисления высоты и ширины таких объектов как "Текст", т.е. если в скрипте отчета нужно получить реальное значение высоты объекта (растягиваемый объект), а в объекте "Текст" используется выражение, то можно использовать такой скрипт в событии OnAfterData:

PascalScript:

```
var
  MemoWidth: Extended;
begin
  MemoWidth := TfrxMemoView(Sender).CalcWidth;
end;
```

C++ Script:

```
float MemoWidth;
  MemoWidth = TfrxMemoView(Sender).CalcWidth;
```

Если данный код поместить в событие OnBeforePrint, то результатом будет высота объекта, в котором содержится выражения, а не его значение.

Служебные объекты

Помимо объектов, имеющихся 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.
`SecondScriptcall`	Boolean	Равно True, если при переносе объектов событие объекта вызывается повторно (происходит при переносе объекта "Текст" с включенным свойством `SuppressRepeated`).

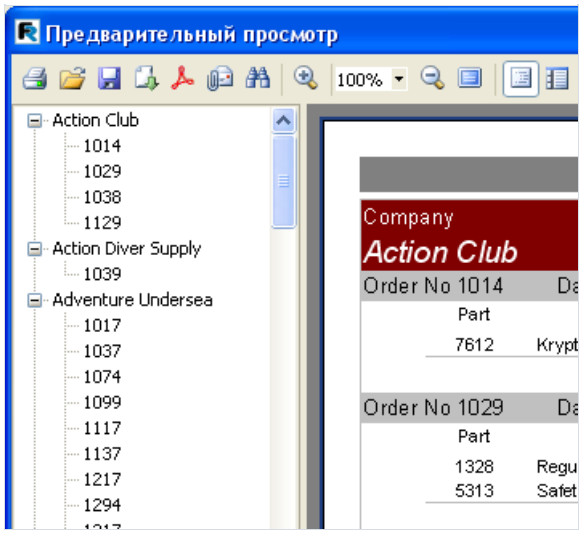
Методы:

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).

Метод	Описание
`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`.

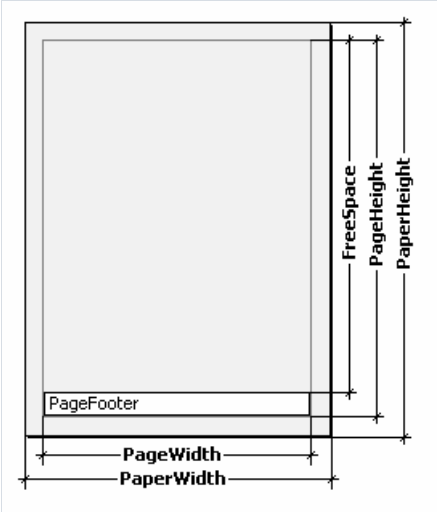
Рассмотрим методы этого объекта.

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 видны 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?

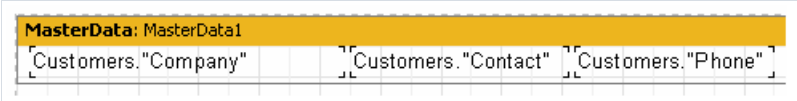
- Ядро FastReport выводит бэнды на страницу до тех пор, пока на ней остается свободное место, достаточное для вывода бэнда.
- Когда свободного места не остается, печатается бэнд "Подвал страницы" (если он есть) и формируется новая пустая страница.

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:



можно напечатать его таким образом:



Это результат работы скрипта, назначенного событию OnBeforePrint бэнда:

PascalScript:

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

C++ Script:

```
void MasterData1OnBeforePrint(TfrxComponent Sender)
{
  Engine.CurX = Engine.CurX + 5;
}
```

Манипуляция свойством `CurY` позволяет, например, напечатать бэнды внахлест:



Соответствующий скрипт:

PascalScript:

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

C++ Script:

```
void MasterData1OnBeforePrint(TfrxComponent Sender)
{
  Engine.CurY = Engine.CurY - 15;
}
```

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:

PascalScript:

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

C++ Script:

```
void MasterData1OnAfterPrint(TfrxComponent Sender)
{
  if (<Line> == 2)
    Engine.NewPage();
}
```

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]
```

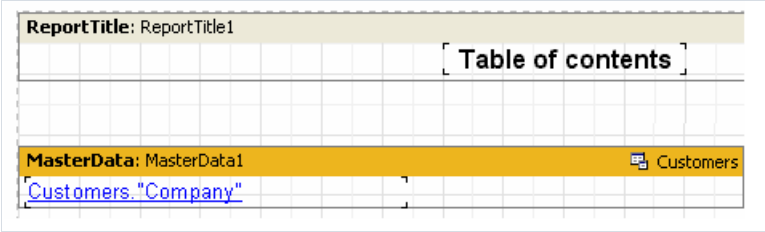
В последнем случае, при построении отчета FastReport раскроет выражение, находящееся в квадратных скобках.

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 таблица Customer.db.

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

Первая страница:

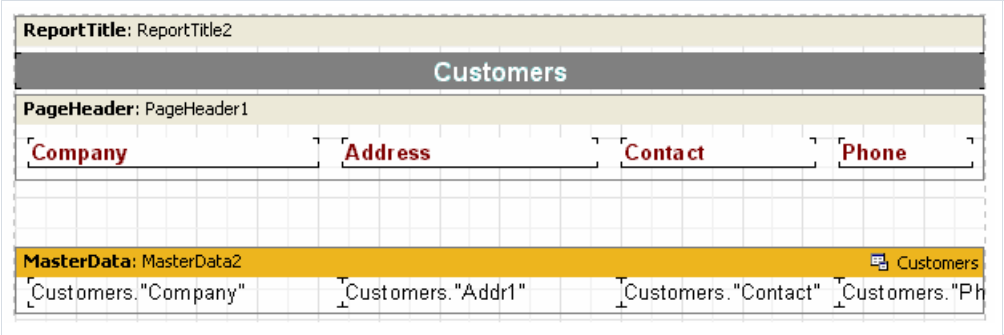


В свойство URL объекта "Текст", который лежит на дата-бэнде, поместим строку:

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

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

Вторая страница:



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

PascalScript:

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

C++ Script:

```
void MasterData2OnBeforePrint(TfrxComponent Sender)
{
  Engine.AddAnchor(<Customers."Company">);
}
```

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

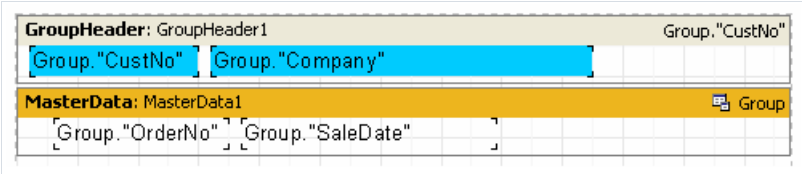
Объект Outline, как уже упоминалось, представляет собой дерево отчета, которое может быть показано в окне предварительного просмотра. При щелчке на элементе дерева произойдет переход на страницу отчета, которая связана с элементом дерева.

> Для работы с Outline необязательно использовать скрипт, т.к. некоторые бэнды имеют механизм, позволяющий формировать дерево автоматически.

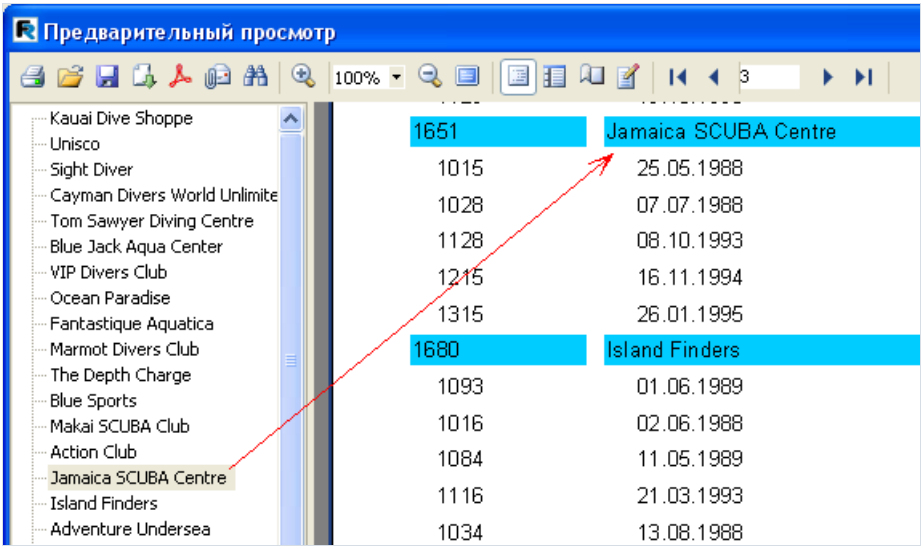
Рассмотрим два примера использования Outline, с помощью бэндов и из скрипта.

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 ранее.

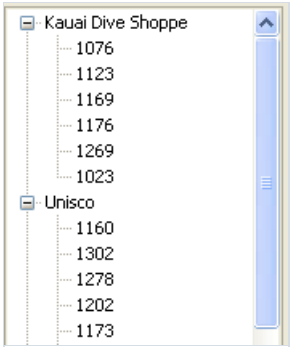


Укажем значение свойства бэнда `GroupHeader1.OutlineText` равным <Group."Company">. При запуске отчета мы увидим следующее:



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`:

PascalScript:

```
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.
```

C++ Script:

```
void GroupHeader1OnBeforePrint(TfrxComponent Sender)
{
  Outline.LevelRoot;
  Outline.AddItem(<Group."Company">);
}

void MasterData1OnBeforePrint(TfrxComponent Sender)
{
  Outline.AddItem(<Group."OrderNo">);
  Outline.LevelUp;
}

{
}
```

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

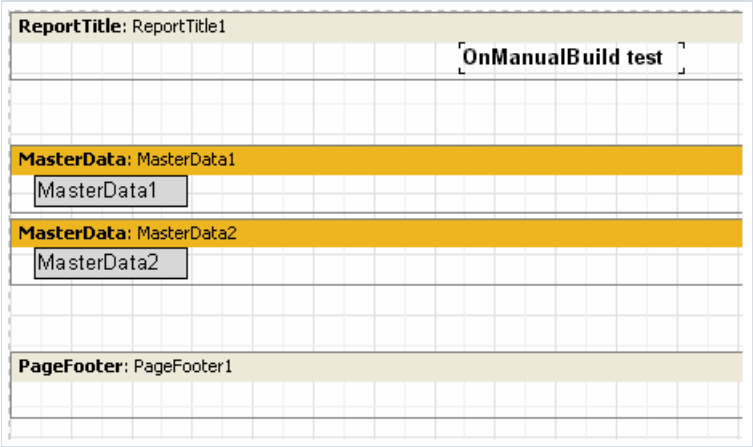
Построением отчета обычно занимается ядро FastReport. Оно выводит бэнды отчета в определенной последовательности столько раз, сколько имеется данных, формируя таким образом готовый отчет. Иногда необходимо вывести отчет нестандартной формы, который ядро FastReport сформировать не в состоянии.

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:



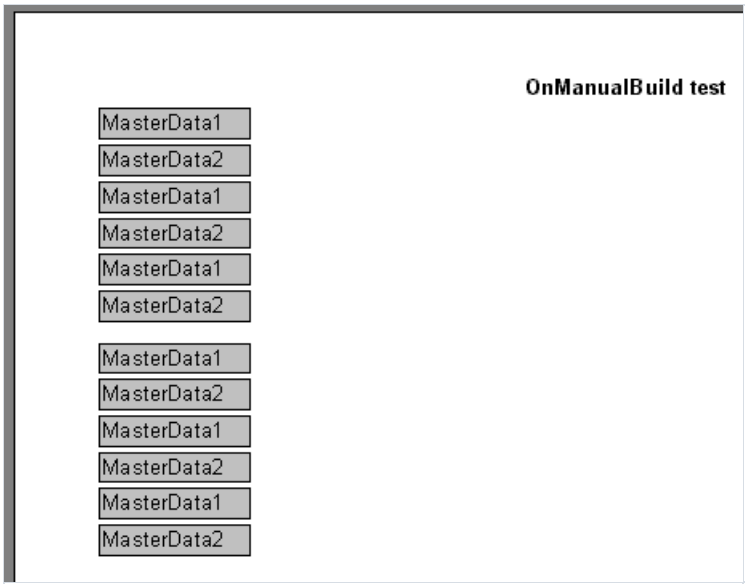
Обработчик выведет эти бэнды в чередующемся порядке, каждый по 6 раз. После шести бэндов будет сделан небольшой промежуток.

PascalScript:

```
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;
```

C++ Script:

```
void Page1OnManualBuild(TfrxComponent Sender)
{
  int i;
  for (i = 1; i <= 6; i++)
  {
    // выводим бэнды друг за другом
    Engine.ShowBand(MasterData1);
    Engine.ShowBand(MasterData2);
    // делаем небольшой промежуток
    if (i == 3)
      Engine.CurY = Engine.CurY + 10;
  }
}
```



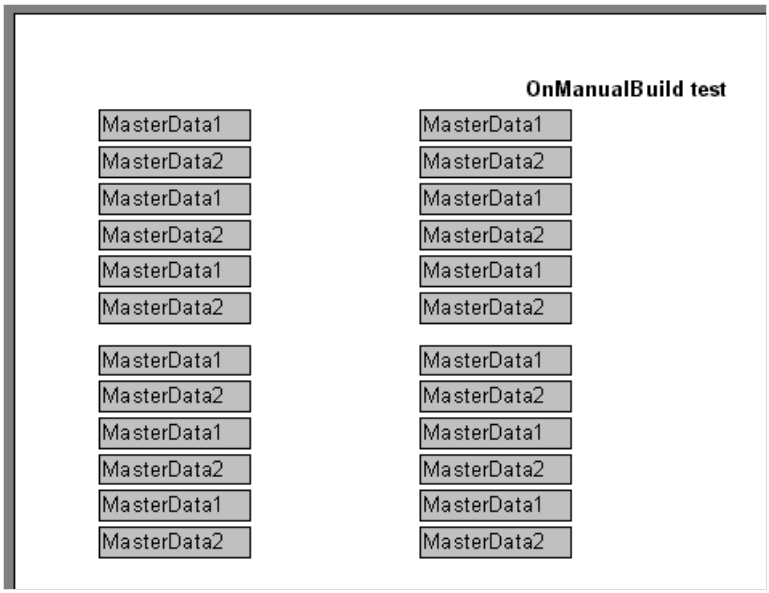
Следующий пример выведет две группы бэндов рядом друг с другом.

PascalScript:

```
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;
```

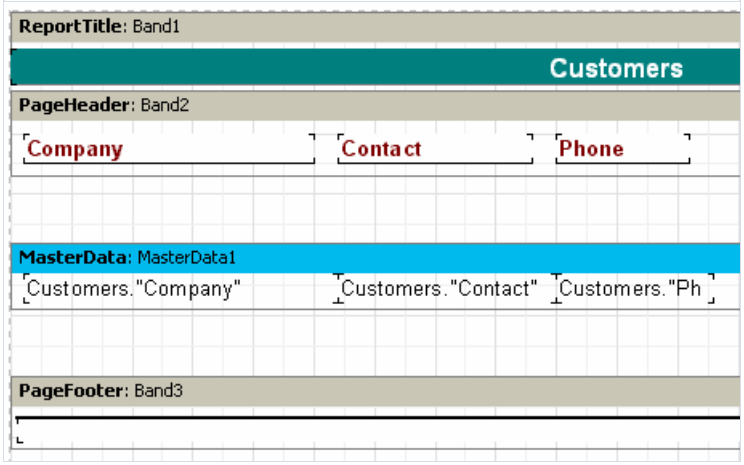
C++Script:

```
void Page1OnManualBuild(TfrxComponent Sender)
{
  int i, j;
  Extended SaveY;
  SaveY = Engine.CurY;
  for (j = 1; j <= 2; j++)
  {
    for (i = 1; i <= 6; i++)
    {
      Engine.ShowBand(MasterData1);
      Engine.ShowBand(MasterData2);
      if (i == 3)
        Engine.CurY = Engine.CurY + 10;
    }
    Engine.CurY = SaveY;
    Engine.CurX = Engine.CurX + 200;
  }
}
```



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.



Скрипт события следующий:

PascalScript:

```
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;
```

C++Script:

```
void Page1OnManualBuild(TfrxComponent Sender)
{
  TfrxDataSet DataSet;
  DataSet = MasterData1.DataSet;
  DataSet.First();
  while (!DataSet.Eof)
  {
    Engine.ShowBand(MasterData1);
    DataSet.Next();
  }
}
```

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. Получить ссылку на нужный источник можно и 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:

PascalScript:

```
var
  Band: TfrxReportTitle;
  Memo: TfrxMemoView;
begin
  Band := TfrxReportTitle.Create(Page1);
  Band.Height := 20;
  Memo := TfrxMemoView.Create(Band);
  Memo.SetBounds(10, 0, 100, 20);
  Memo.Text := 'This memo is created in code';
end.
```

C++ Script:

```
TfrxReportTitle Band;
TfrxMemoView Memo;
{
  Band = TfrxReportTitle.Create(Page1);
  Band.Height = 20;
  Memo = TfrxMemoView.Create(Band);
  Memo.SetBounds(10, 0, 100, 20);
  Memo.Text = "This memo is created in code";
}
```

Запустим отчет:



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

Сводные 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.

Строим кросс-отчет

Перейдем от теории к практике. Построим простой кросс-отчет, показывающий зарплату сотрудников за четыре года. Для этого нам понадобится таблица crosstest, которая находится в папке FastReport DEMOS\MAIN. 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:

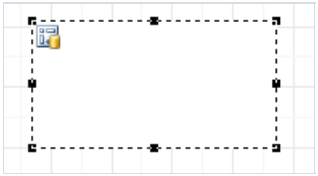
```
Table1:
DatabaseName = 'c:\Program Files\FastReports\FastReport 6\Demos\Main'
TableName = 'crosstest.db'
```

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

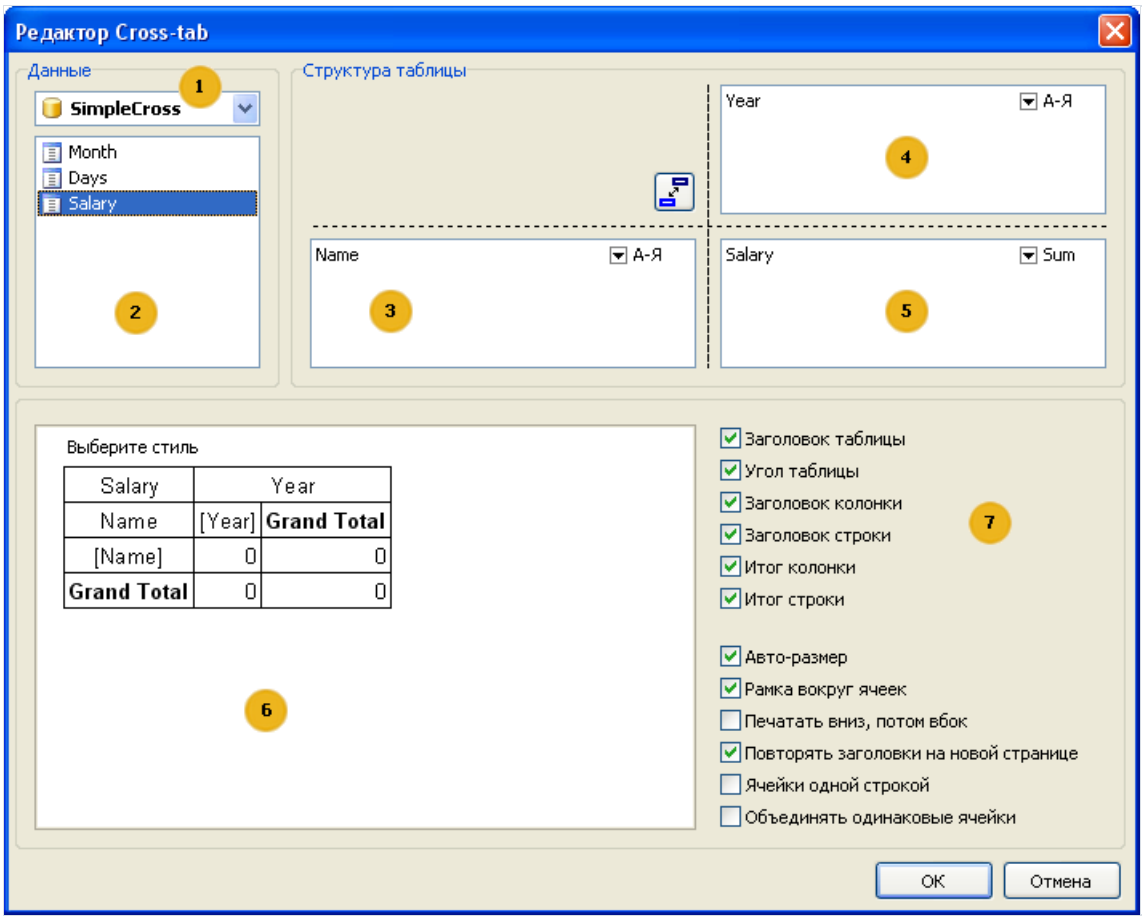
```
frxDBDataSet1:
DataSet = Table1
UserName = 'SimpleCross'
```

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" :



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



Цифрами отмечены:

1 – выпадающий список доступных источников данных;

2 – список полей в выбранном источнике данных. Поля из этого списка можно перетаскивать в списки 3, 4, 5;

3 – список полей, которые образуют заголовок строки;

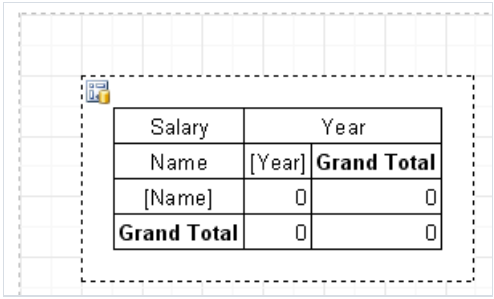
4 – список полей, которые образуют заголовок столбца;

5 – список полей, которые образуют ячейку таблицы;

6 – здесь отображается структура будущей таблицы.

7 – настройки объекта.

Как видно, действовать здесь придется только мышью. В нашем случае достаточно перетащить поля из списка 2 в списки 3, 4, 5, как показано на рисунке. Пока больше делать ничего не будем - закроем редактор кнопкой OK. Мы увидим, что объект отображает свою структуру на странице отчета:

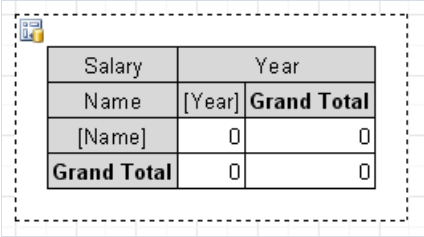


Если сейчас запустить отчет, мы увидим следующее:

Salary	Year				
Name	1999	2000	2001	2002	**Grand Total**
Ann	3300	2700	3100	1700	10800
Ben	3900	2100		1800	7800
Catherine	6100	3200			9300
Den		3999	8100		12099
Grand Total	13300	11999	11200	3500	39999

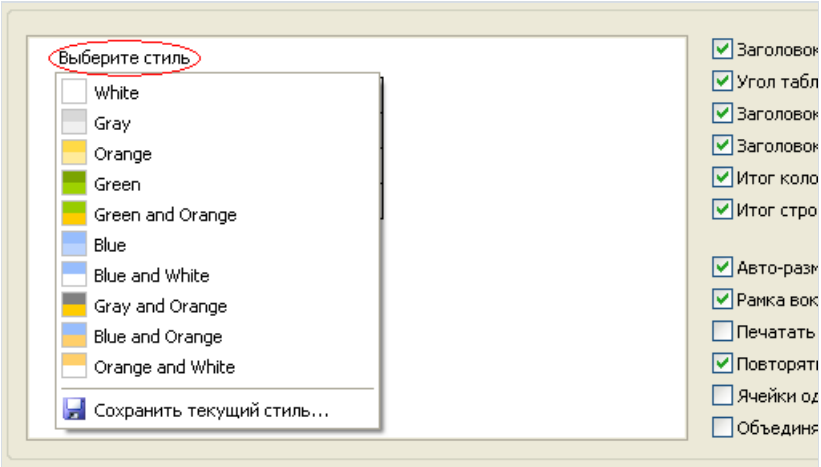
Внешний вид таблицы

Продолжим изучение объекта. Первое, что нам захочется сделать – это сменить цвет заголовков и поменять английские надписи на русские. Сделать это очень просто. Чтобы сменить цвет заголовка, последовательно щелкните на объектах заголовка и выберите нужный цвет кнопкой на панели инструментов. У нас должно получиться следующее:



Salary	Year	
Name	[Year]	**Grand Total**
[Name]	0	0
Grand Total	0	0

Также можно воспользоваться готовыми стилями. Для этого зайдите в редактор объекта и нажмите кнопку с надписью "Выберите стиль":

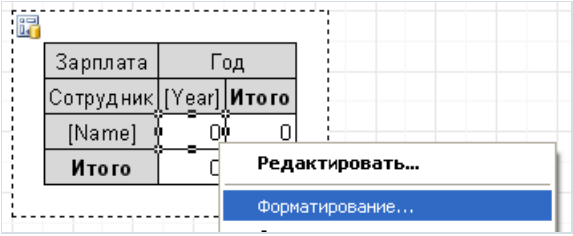


Чтобы сменить текст надписей, дважды щелкните на ячейке – вы увидите редактор текста, в котором наберите нужный текст. После этого наш объект будет выглядеть так:



Зарплата	Год	
Сотрудник	[Year]	**Итого**
[Name]	0	0
Итого	0	0

Осталось задать формат, в котором выводятся денежные значения. Для этого щелкните на первом объекте, представляющем ячейку (он находится на пересечении [Year] и [Name]), вызовите его контекстное меню правой кнопкой мыши и выберите пункт "Форматирование...".



Затем выберите нужный формат. Получается вот что:

Зарплата	Год				
Сотрудник	1999	2000	2001	2002	**Итого**
Ann	3 300,00р.	2 700,00р.	3 100,00р.	1 700,00р.	10 800,00р.
Ben	3 900,00р.	2 100,00р.		1 800,00р.	7 800,00р.
Catherine	6 100,00р.	3 200,00р.			9 300,00р.
Den		3 999,00р.	8 100,00р.		12 099,00р.
Итого	13 300,00р.	11 999,00р.	11 200,00р.	3 500,00р.	39 999,00р.

Хотите сказать, что зарплата ваших сотрудников измеряется в долларах? ;) Это легко поправить, указав другую строку форматирования: `$%2.2n`.

Использование функций

В нашем примере мы вывели в строке "Итого" сумму зарплат каждого сотрудника за четыре года. Вы можете использовать следующие функции:

`SUM` – сумма

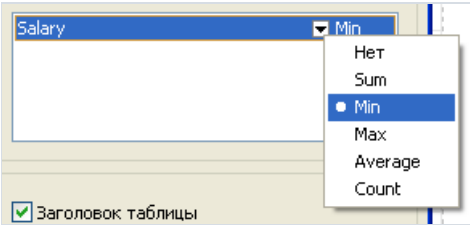
`MIN` – минимальное значение

`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:

Зарплата	Год				
Сотрудник	1999	2000	2001	2002	**Минимум**
Ann	3 300,00р.	2 700,00р.	3 100,00р.	1 700,00р.	1 700,00р.
Ben	3 900,00р.	2 100,00р.		1 800,00р.	1 800,00р.
Catherine	6 100,00р.	3 200,00р.			3 200,00р.
Den		3 999,00р.	8 100,00р.		3 999,00р.
Минимум	3 300,00р.	2 100,00р.	3 100,00р.	1 700,00р.	1 700,00р.

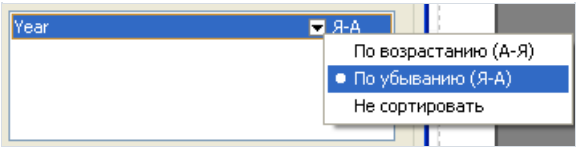
Сортировка значений

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 сортировки:

- по возрастанию;
- по убыванию;
- отсутствие сортировки. В этом 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 зайдем в редактор кросс-объекта и выберем элемент колонки "Year". Чтобы сменить сортировку, щелкнем на значок со стрелкой вниз:



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

Зарплата	Год				
Сотрудник	2002	2001	2000	1999	**Итого**
Ann	1 700,00р.	3 100,00р.	2 700,00р.	3 300,00р.	10 800,00р.
Ben	1 800,00р.		2 100,00р.	3 900,00р.	7 800,00р.
Catherine			3 200,00р.	6 100,00р.	9 300,00р.
Den		8 100,00р.	3 999,00р.		12 099,00р.
Итого	3 500,00р.	11 200,00р.	11 999,00р.	13 300,00р.	39 999,00р.

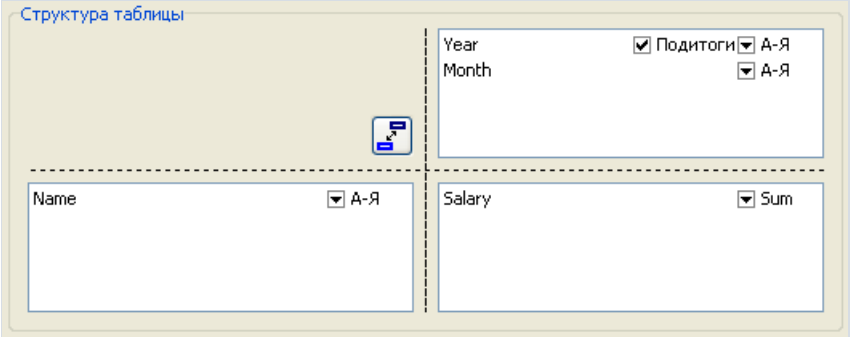
Таблица с составными заголовками

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. Таблица содержит данные следующего характера:

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

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.

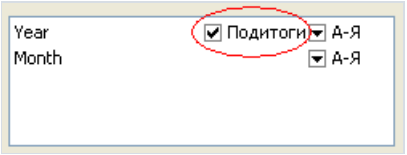
Какого вида отчет мы хотим получить? Он должен быть похож на отчет из предыдущего примера, но с разбивкой годов на месяцы. Следовательно, настроить кросс-объект надо таким же образом, только добавив в заголовок столбца поле "Month":



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

Зарплата	Год, Месяц															
Сотрудник	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		1900	2000		3900		2100		2100				0	1800	1800	7800
Catherine		3000	3100		6100			3200	3200				0		0	9300
Den					0	3999			3999	4000	4100		8100		0	12099
Итого	1000	4900	6200	1200	13300	5299	3500	3200	11999	4000	5600	1600	11200	3500	3500	39999

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 "Подитоги":



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: выделите нужный объект и впишите в него следующий текст:

```
Итого за [Value]
```

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

Зарплата	Год, Месяц								
Сотрудник	1999					2000			
	2	10	11	12	**Итого за 1999**	1	2	3	**Итого за 2000**
Ann	1000		1100	1200	3300	1300	1400		2700
Ben		1900	2000		3900		2100		2100
Catherine		3000	3100		6100			3200	3200
Den					0	3999			3999
Итого	1000	4900	6200	1200	13300	5299	3500	3200	11999

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? Рассмотрим 3 способа управления размерами ячеек.

Первый способ – 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:

Зарплата					
Сотрудник	1999				
	2	10	11	12	**Итого за 1999**
Ann	1000		1100	1200	3300
Ben		1900	2000		3900
Catherine		3000	3100		6100
Den					0
Итого	1000	4900	6200	1200	13300

Однако такой способ можно использовать далеко не всегда – что, если сами значения строк/столбцов достаточно длинные, ведь их нельзя исправить вставкой разрыва строки вручную.

Второй способ - использовать 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 пикселам. 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**

Наконец, третий способ - установить самому нужные размеры ячеек. Для этого нужно отключить свойство "Авто-размер" в редакторе объекта (или `AutoSize` в инспекторе). Теперь размер всех элементов таблицы можно менять вручную. Сделать это очень просто - при наведении мыши на элементы таблицы указатель

мыши меняет форму, предлагая изменить ширину или высоту. Вот пример того, что можно сделать:



Зарплата	Год, Месяц		
Сотрудник	[Year]		Итого
	[Month]	**Итого за [Year]**	
[Name]	0	0	0
Итого	0	0	0

Учтите, что при отключении свойства "Авто-размер" перестает работать подбор размера ячеек. Если вы установили недостаточную ширину ячейки, при печати текст может быть обрезан:



Зарплата	Го				
Сотрудник	1999				
	2	10	11	12	**Итого за 1999**
Ann	1 000,0 Оп		1 100,0 Оп	1 200,0 Оп	3 300,00р
Ben		1 900,0 Оп	2 000,0 Оп		3 900,00р
Catherine		3 000,0 Оп	3 100,0 Оп		6 100,00р
Den					0,00р.
Итого	1 000,0 Оп	4 900,0 Оп	6 200,0 Оп	1 200,0 Оп	13 300,00 р

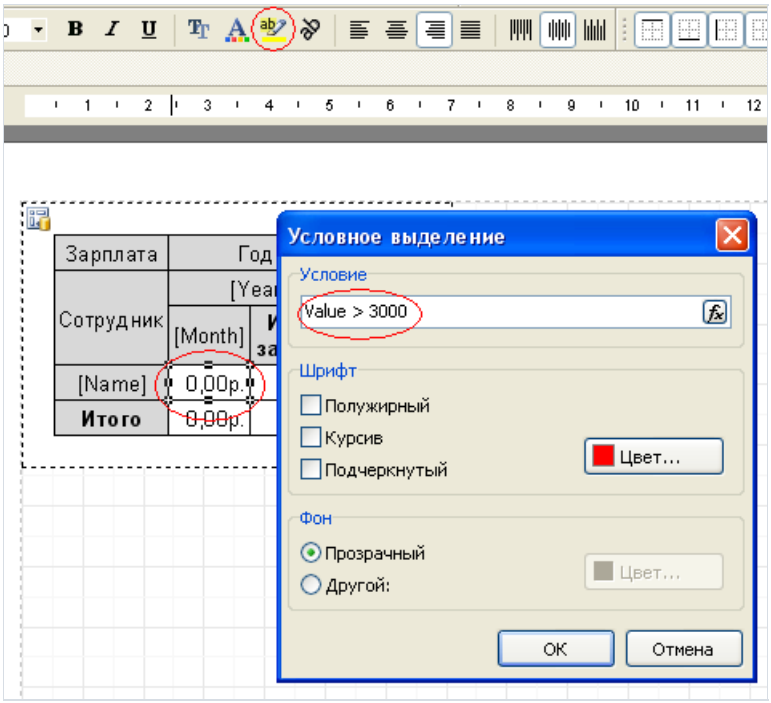
В таком случае просто увеличьте размер соответствующих ячеек.

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 щелкнем на объекте, который представляет ячейку таблицы, и нажмем кнопку на панели инструментов. Откроется окно редактора выделения, в котором надо задать следующее условие:

```
Value > 3000
```



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

Зарплата	Год, Месяц					
Сотрудник	1999					
	2	10	11	12	**Итого за 1999**	1
Ann	1 000,00р.		1 100,00р.	1 200,00р.	3 300,00р.	1 300,00р.
Ben		1 900,00р.	2 000,00р.		3 900,00р.	
Catherine		3 000,00р.	3 100,00р.		6 100,00р.	
Den					0,00р.	3 999,00р.
Итого	1 000,00р.	4 900,00р.	6 200,00р.	1 200,00р.	13 300,00р.	5 299,00р.

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:

Pascal script:

```
procedure Cross1OnPrintCell(Memo: TfrxMemoView;
  RowIndex, ColumnIndex, CellIndex: Integer;
  RowValues, ColumnValues, Value: Variant);
begin
  if ColumnIndex = 2 then
    Memo.Color := clRed;
end;
```

C++ Script:

```
void Cross1OnPrintCell(TfrxMemoView Memo, int RowIndex, int ColumnIndex, int CellIndex,
  Variant RowValues, Variant ColumnValues, Variant Value)
{
  if (ColumnIndex == 2) { Memo.Color = clRed; }
}
```

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:

Pascal script:

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

C++ Script:

```
void Cross1OnPrintColumnHeader(TfrxMemoView Memo,
  Variant HeaderIndexes, Variant HeaderValues, Variant Value)
{
  if ((VarToStr(HeaderValues[0]) == "1999") &&
     (VarToStr(HeaderValues[1]) == "11"))
  {
    Memo.Color = clRed;
  }
}
```

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):

Pascal script:

```
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;
```

C++ Script:

```
void Cross1OnPrintCell(TfrxMemoView Memo,
  int RowIndex, int ColumnIndex, int CellIndex,
  Variant RowValues, Variant ColumnValues, Variant Value)
{
  if ((VarToStr(ColumnValues[0]) == "1999") &&
     (VarToStr(ColumnValues[1]) == "11"))
  {
    Memo.Color = clRed;
  }
}
```

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. FastReport 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:

Pascal script:

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

C++ Script:

```
void Cross1OnPrintColumnHeader(TfrxMemoView Memo,
  Variant HeaderIndexes, Variant HeaderValues, Variant Value)
{
  if ((HeaderIndexes[0] == 0) && (HeaderIndexes[1] == 2)) { Memo.Color = clRed; }
}
```

Управление размером строк и 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:

Pascal script:

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

C++ Script:

```
void Cross1OnCalcWidth(int ColumnIndex, variant ColumnValues, Extended &Width)
{
  if ((VarToStr(ColumnValues[0]) == "1999") &&
    (VarToStr(ColumnValues[1]) = "11"))
    {
      Width = 100;
    }
}
```

Результат:

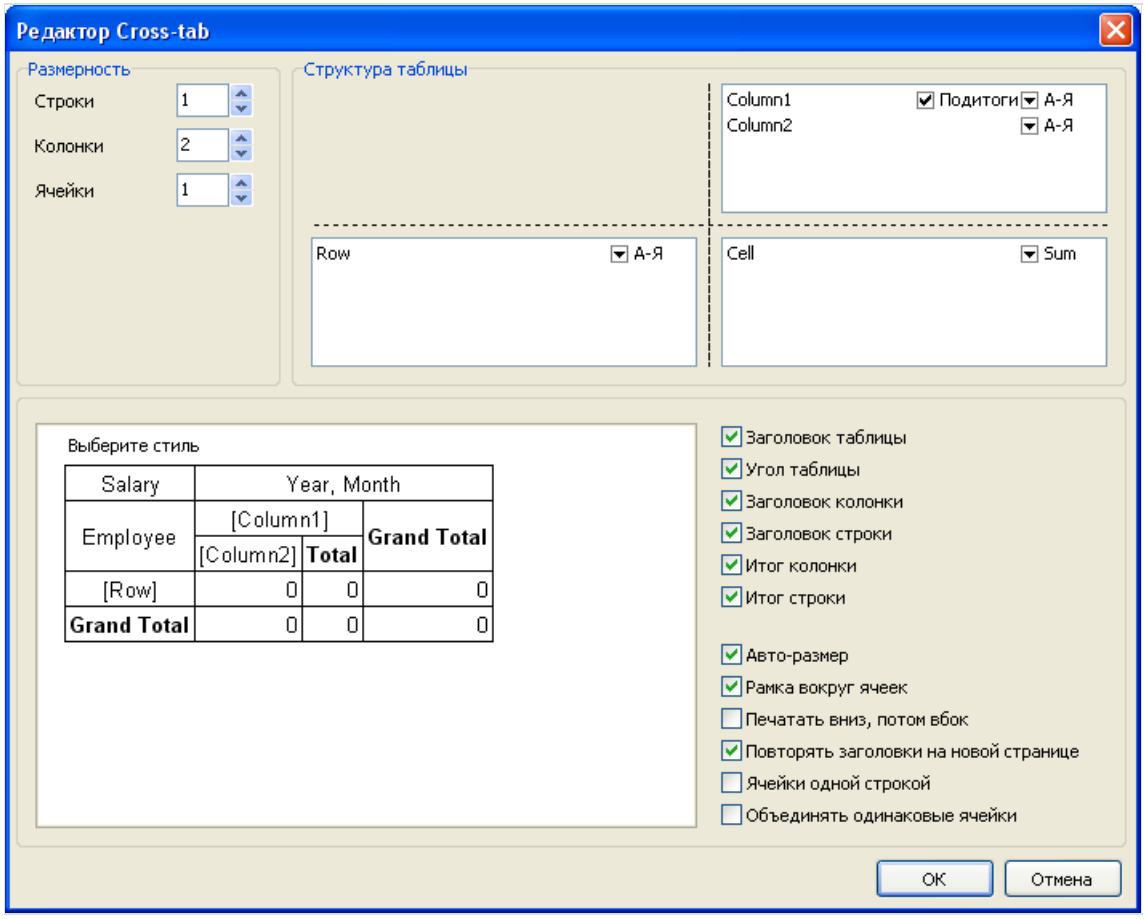
	1999				
	2	10	11	12	**Total**
Ann	1000		1100	1200	**3300**
Ben		2100	2200		**4300**
Catherine		3000	3100		**6100**
Den					**0**
Grand Total	**1000**	**5100**	**6400**	**1200**	**13700**

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:

PascalScript:

```
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;
```

C++ Script:

```
void Cross1OnBeforePrint(TfrxComponent Sender)
{
  Cross1.AddValue(["Ann"], [2001, 2], [1500]);
  Cross1.AddValue(["Ann"], [2001, 3], [1600]);
  Cross1.AddValue(["Ann"], [2002, 1], [1700]);
  Cross1.AddValue(["Ben"], [2002, 1], [2000]);
  Cross1.AddValue(["Den"], [2001, 1], [4000]);
  Cross1.AddValue(["Den"], [2001, 2], [4100]);
}
```

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.

> 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:

Salary	Year, Month						
Employee	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.

Добавление объектов в таблицу

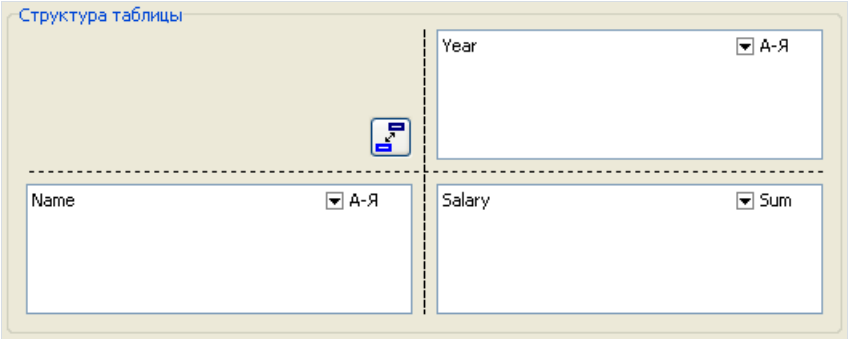
В таблицу можно вставлять посторонние объекты. Для чего это может быть нужно? Например, чтобы выделить какие-нибудь значения ячеек. Можно, конечно, использовать условное выделение (мы рассматривали подобный пример) - но не всегда его возможностей бывает достаточно.

Рассмотрим пример, в котором каждое значение ячейки представлено в виде маленькой шкалы, которая отображает уровень зарплаты. Вот что должно получиться в результате:

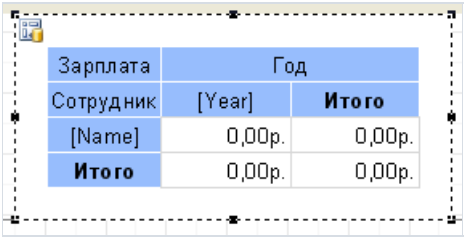
Зарплата	Год				
Сотрудник	1999	2000	2001	2002	**Итого**
Ann	3 300,00р.	2 700,00р.	3 100,00р.	1 700,00р.	10 800,00р.
Ben	3 900,00р.	2 100,00р.		1 800,00р.	7 800,00р.
Catherine	6 100,00р.	3 200,00р.			9 300,00р.
Den		3 999,00р.	8 100,00р.		12 099,00р.
Итого	13 300,00р.	11 999,00р.	11 200,00р.	3 500,00р.	39 999,00р.

Красным помечены значения менее 100, желтым - менее 3000, зеленым - более 3000.

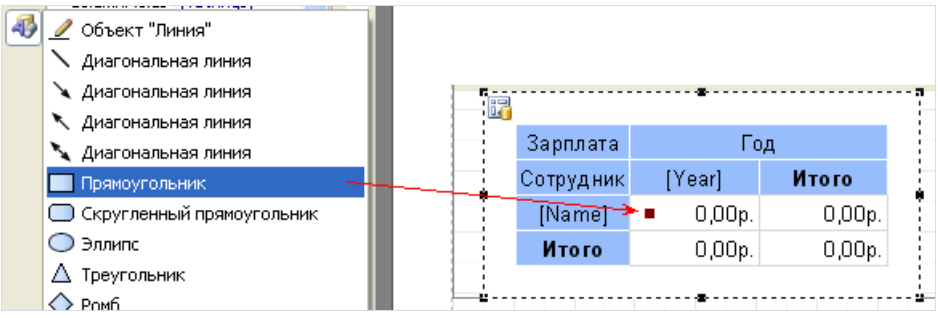
Приступим к созданию отчета. Положим на лист объект "Кросс-таблица БД" и настроим его содержимое:



Настроим внешний вид таблицы. Для этого выберем цвет заголовков, поменяем английские надписи на русские (Зарплата, Сотрудник, Год, Итого) и отключим свойство "Авто-размер" (`AutoSize`). В результате должна получиться такая таблица:

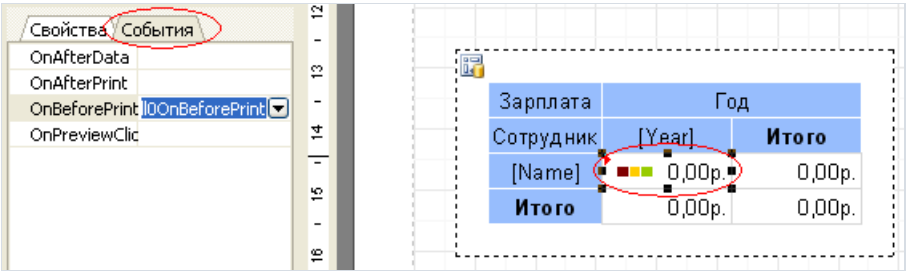


Теперь добавим элементы, которые будут отображать шкалу, в таблицу. Для этого выберите объект "Рисование/Прямоугольник" в панели объектов и положите его внутрь ячейки таблицы:



Таким же образом добавьте еще два прямоугольника.

Теперь создадим скрипт, который будет показывать нужное количество прямоугольников и раскрашивать их в один из цветов. Для этого выделите саму ячейку и в инспекторе объектов создайте обработчик события OnBeforePrint:



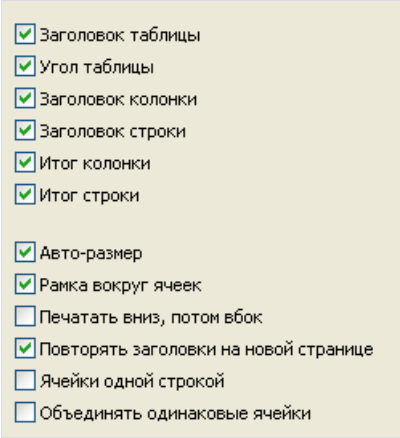
В обработчике напишем следующее (обратите внимание на названия объектов - вставленные в таблицу объекты имеют именно такие имена):

```
procedure DBCross1Cell0OnBeforePrint(Sender: TfrxComponent);
begin
  // Value - это текущее значение ячейки
  if Value < 100 then
  begin
    // это первый объект
    DBCross1Object1.Color := clMaroon; // красный
    // это второй объект
    DBCross1Object2.Color := clWhite;
    // это третий объект
    DBCross1Object3.Color := clWhite;
  end
  else if Value < 3000 then
  begin
    DBCross1Object1.Color := $00CCFF;  // желтый
    DBCross1Object2.Color := $00CCFF;
    DBCross1Object3.Color := clWhite;
  end
  else
  begin
    DBCross1Object1.Color := $00CC98;  // зеленый
    DBCross1Object2.Color := $00CC98;
    DBCross1Object3.Color := $00CC98;
  end;
end;
```

Это все - если запустить отчет, мы увидим таблицу, приведенную в начале этого раздела.

Другие полезные настройки

Рассмотрим настройки таблицы, которые могут оказаться полезными. Все эти настройки доступны в редакторе объекта.



Верхняя группа настроек определяет, показывать или нет те или иные элементы таблицы.

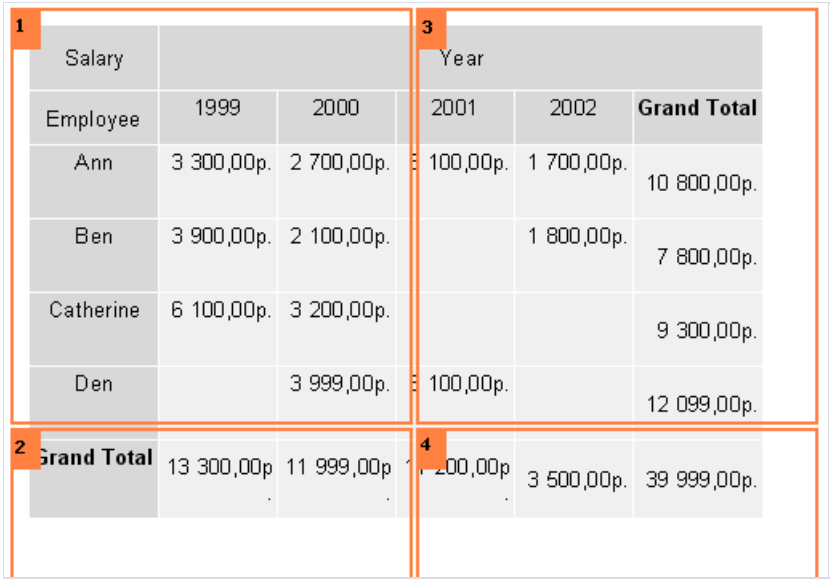
Опция "Авто-размер" нами уже была рассмотрена, она позволяет отключать автоматический подбор размеров таблицы и делать это вручную.

Опция "Рамка вокруг ячеек" включает линии рамки у ячеек, которые являются внешними. Это позволяет сделать рамку вокруг всего блока ячеек (не таблицы!). Вот пример такой таблицы (все рамки у ячеек выключены, рисуется только внешняя):

Зарплата			Год		
Сотрудник	1999	2000	2001	2002	**Сумма**
Ann	3 300,00р.	2 700,00р.	3 100,00р.	1 700,00р.	10 800,00р.
Ben	3 900,00р.	2 100,00р.		1 800,00р.	7 800,00р.
Catherine	6 100,00р.	3 200,00р.			9 300,00р.
Den		3 999,00р.	8 100,00р.		12 099,00р.
Сумма	13 300,00р.	11 999,00р.	11 200,00р.	3 500,00р.	39 999,00р.

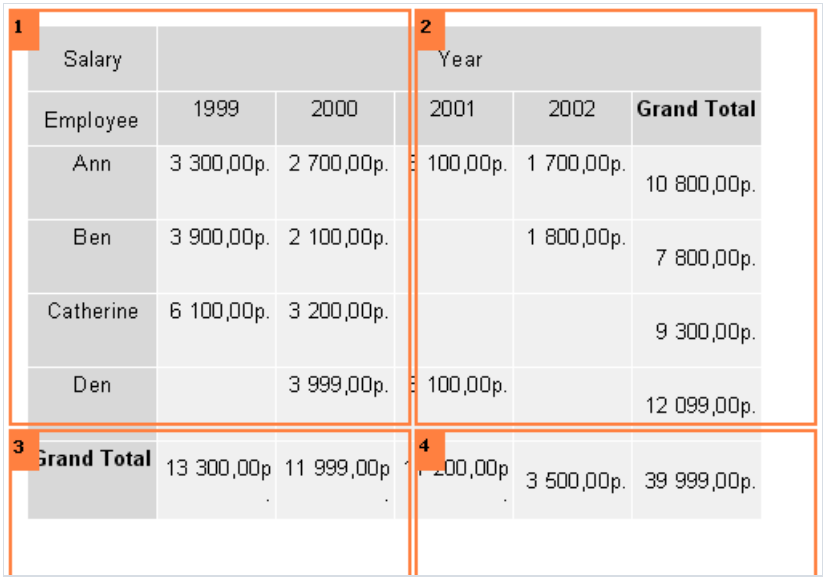
Опция "Печатать вниз, потом вбок" определяет, как разбивать большую таблицу на страницы. Вот пример разбивки, с данной опцией и без - обратите внимание на нумерацию страниц.

1. "Печатать вниз, потом вбок" - включена:



1 Salary	3 Year			
Employee	1999	2000	2001	2002	**Grand Total**
Ann	3 300,00р.	2 700,00р.	100,00р.	1 700,00р.	10 800,00р.
Ben	3 900,00р.	2 100,00р.		1 800,00р.	7 800,00р.
Catherine	6 100,00р.	3 200,00р.			9 300,00р.
Den		3 999,00р.	100,00р.		12 099,00р.
2 **Grand Total**	13 300,00р.	11 999,00р.	4 200,00р.	3 500,00р.	39 999,00р.

2. "Печатать вниз, потом вбок" - выключена:



1 Salary	2 Year			
Employee	1999	2000	2001	2002	**Grand Total**
Ann	3 300,00р.	2 700,00р.	100,00р.	1 700,00р.	10 800,00р.
Ben	3 900,00р.	2 100,00р.		1 800,00р.	7 800,00р.
Catherine	6 100,00р.	3 200,00р.			9 300,00р.
Den		3 999,00р.	100,00р.		12 099,00р.
3 **Grand Total**	13 300,00р.	11 999,00р.	4 200,00р.	3 500,00р.	39 999,00р.

Опция "Повторять заголовки на новой странице" определяет, надо ли печатать заголовки таблицы на всех страницах, на которые разбивается большая таблица.

Опция "Ячейки одной строкой" используется, если в вашей таблице два или более значения в ячейке. Опция определяет, надо ли печатать ячейки рядом (в одной строке) или друг под другом.

Опция "Объединять одинаковые ячейки" позволяет объединять две или несколько ячеек в одной строке, если они имеют одинаковые значения. Пример такой таблицы:

Дни	Год				
Сотрудник	1999	2000	2001	2002	**Сумма**
Ann	3		4	2	12
Ben	4	2		2	8
Catherine	6	3			9
Den		4	7		11
Сумма	13	12	11	4	40

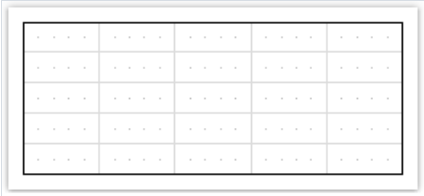
Также из инспектора объектов доступны следующие настройки:

- `AddWidth`, `AddHeight` - количество пустого места, которое будет добавлено при расчете размера ячейки. Эти свойства позволяют расширить ячейку на заданное значение. При этом свойство `AutoSize` должно быть True;

- `NextCross` - указатель на объект "Кросс-таблица", который будет выведен справа от данной таблицы;
- `NextCrossGap` - промежуток между двумя таблицами.

Табличные отчеты

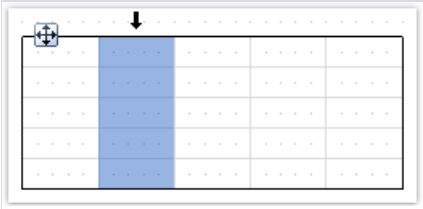
Объект "Таблица" состоит из строк, колонок и ячеек и представляет собой упрощенный аналог таблицы Microsoft Excel. Он выглядит следующим образом:



Настройка колонок

Вы можете удалить или вставить колонки с помощью контекстного меню. Для этого:

- выделите таблицу или любой ее элемент и поместите указатель мыши над нужной колонкой. Форма указателя поменяется на маленькую черную стрелку:

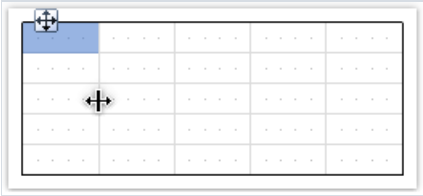


- нажмите левую кнопку мыши, чтобы выделить колонку;
- нажмите правую кнопку мыши, чтобы показать контекстное меню колонки;
- если вам нужно выделить несколько соседних колонок, нажмите левую кнопку и, не отпуская ее, двигайте мышь влево или вправо, чтобы выделить соседние колонки.

Управление размером колонок

Вы можете указать ширину колонки одним из следующих способов:

- выделите таблицу или любой ее элемент и поместите указатель мыши на границе между двумя колонками. Форма указателя поменяется на горизонтальный разделитель:



Нажмите левую кнопку мыши и потяните мышь, чтобы изменить размеры колонки;

- выделите колонку и укажите нужную ширину в свойстве "Ширина" (`Width`). Это свойство доступно в окне "Инспектор объектов".

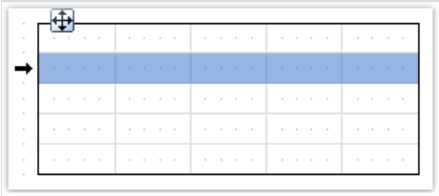
Вы также можете включить свойство колонки "Авторазмер" (`AutoSize`). При запуске отчета ширина колонки будет подобрана автоматически. Для того чтобы ограничить ширину колонки, можно указать свойства "Минимальная ширина" (`MinWidth`) и "Максимальная ширина" (`MaxWidth`).

> Ширина колонки не должна быть больше, чем ширина страницы.

Настройка строк

Строки настраиваются аналогичным образом. Чтобы выделить строку, сделайте следующее:

- выделите таблицу или любой ее элемент и поместите указатель мыши слева от нужной строки. Форма указателя поменяется на маленькую черную стрелку:

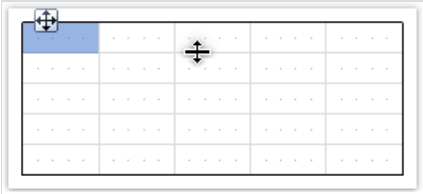


- нажмите левую кнопку мыши, чтобы выделить строку;
- нажмите правую кнопку мыши, чтобы показать контекстное меню строки;
- если вам нужно выделить несколько соседних колонок, нажмите левую кнопку и, не отпуская ее, двигайте мышь вверх или вниз, чтобы выделить соседние строки.

Управление размером строк

Вы можете указать высоту строки одним из следующих способов:

- выделите таблицу или любой ее элемент и поместите указатель мыши на границе между двумя строками. Форма указателя поменяется на вертикальный разделитель:



Нажмите левую кнопку мыши и потяните мышь, чтобы изменить размеры колонки;

- выделите строку и укажите нужную высоту в свойстве "Высота" (`Height`). Это свойство доступно в окне "Свойства".

Вы также можете включить свойство строки "Авторазмер" (`AutoSize`). При запуске отчета высота строки будет подобрана автоматически. Для того чтобы ограничить высоту строки, можно указать свойства "Минимальная высота" (`MinHeight`) и "Максимальная высота" (`MaxHeight`).

> Высота строки не должна быть больше, чем высота страницы.

Настройка ячеек

Ячейка представляет собой текстовый объект. По сути, класс ячейки наследуется от объекта "Текст". Все, сказанное выше про объект "Текст", относится и к ячейке таблицы.

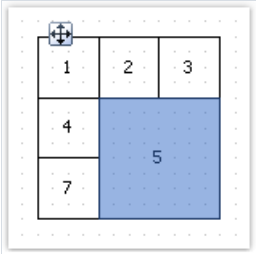
- Редактировать текст ячейки можно так же, как объект "Текст". Кроме того, вы можете перетащить (drag&drop) элемент из окна "Данные" внутрь ячейки.
- Рамка и заливка ячеек могут быть настроены с помощью панели инструментов "Рамка и заливка".

Чтобы вызвать контекстное меню ячейки, щелкните на ней правой кнопкой мыши.

Объединение ячеек

Вы можете объединять соседние ячейки таблицы - при этом получается одна большая ячейка. Для того, чтобы это сделать:

- выделите начальную ячейку с помощью мыши;
- нажмите левую кнопку мыши и, не отпуская ее, двигайте мышь, чтобы выделить группу ячеек;
- на выделенной области нажмите правую кнопку мыши, чтобы показать контекстное меню ячейки. В контекстном меню ячейки выберите пункт "Объединить ячейки".



Для того чтобы разбить ячейку, вызовите ее контекстное меню и выберите пункт "Разбить ячейку".

Объекты в ячейках

В ячейку можно добавлять другие объекты отчета, например, рисунки. Следующие объекты добавлять в ячейку нельзя:

- "Таблица";
- "Кросс-таблица";
- "Вложенный отчет".

Для того чтобы добавить объект в ячейку, просто перетащите его внутрь ячейки. Вы можете свободно перемещать объект между ячейками, а также вынести его обратно за пределы таблицы.



Ячейка служит контейнером для помещенных в нее объектов. Это значит, что вы можете использовать свойство "Выравнивание" (`Align`) у объектов внутри ячейки. Это позволит изменять размеры объектов при изменении размеров ячейки.

Графики, диаграммы

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.

> Также можно использовать библиотеку TeeChartPro, которая приобретается отдельно.

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

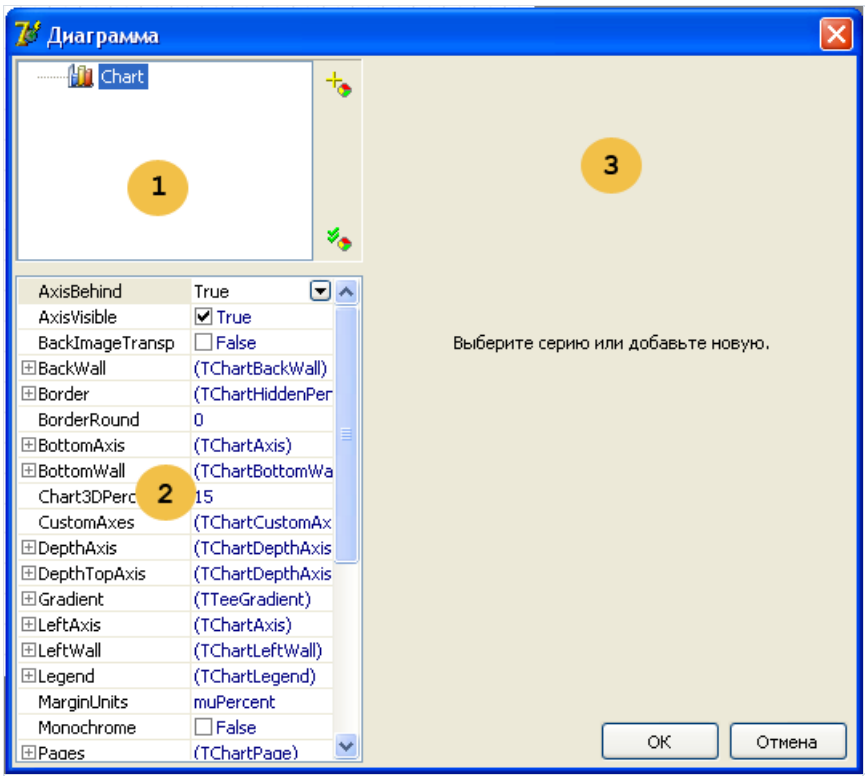
```
Name        Area        Population
Argentina   2 777 815   32 300 003
Bolivia     1 098 575    7 300 000
...
```

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

```
Table1:
DatabaseName = 'DBDEMOS'
TableName = 'coutry.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 – 18x8см. Чтобы настроить объект, вызовем его редактор двойным щелчком мыши.



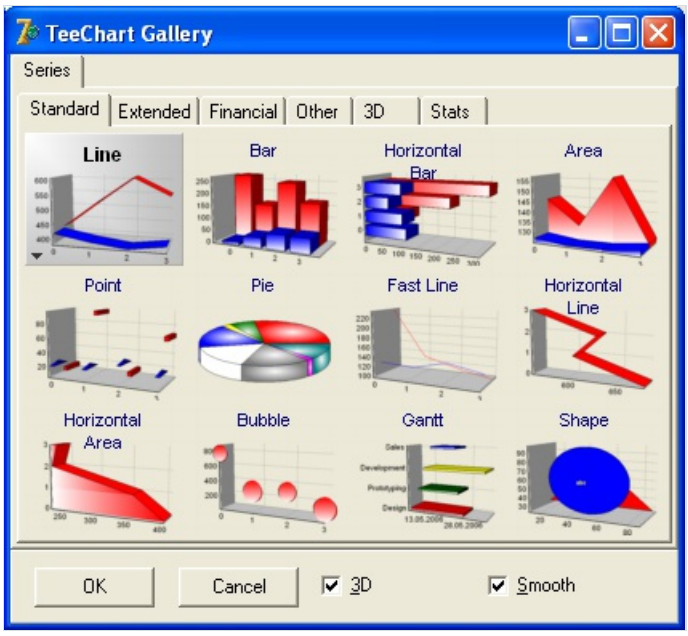
Цифрами на рисунке обозначены:

1 – структура диаграммы. Диаграмма может содержать одну или несколько серий (series).

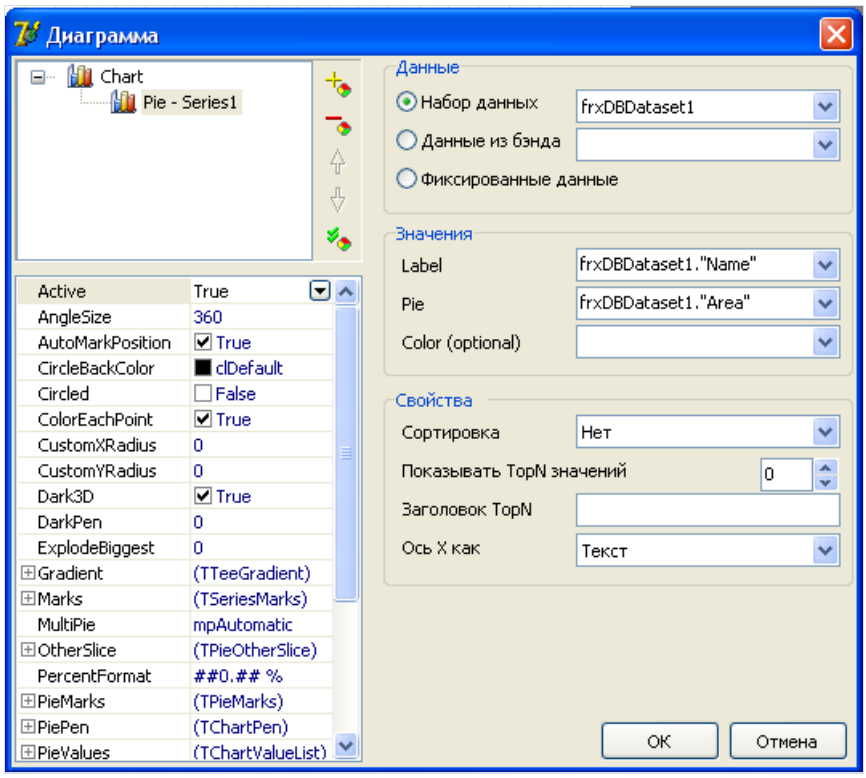
2 –инспектор объектов, который отображает свойства выбранного в окне 1 элемента. Таким образом можно произвести тонкую настройку свойств диаграммы.

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:



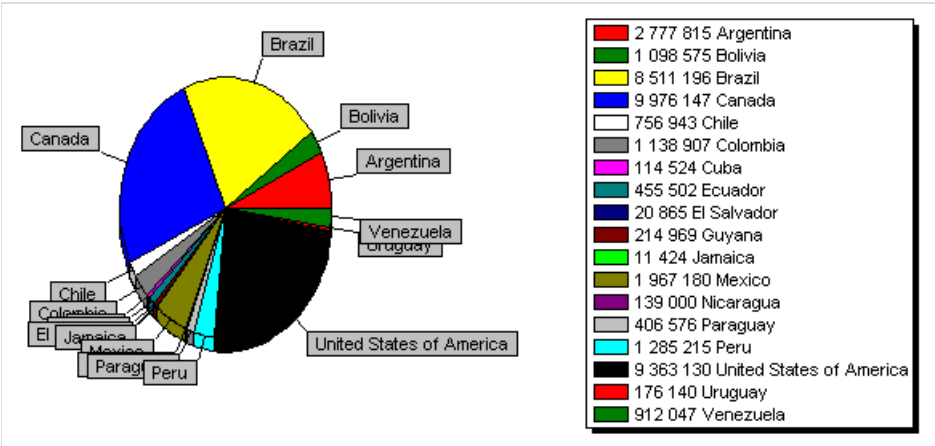
После добавления серии панель 3 стала активной. Здесь надо указать, какие данные будут использоваться при построении диаграммы. Сначала выберем набор данных из выпадающего списка "Набор данных". Поля "Label" и "Pie" заполним следующим образом – их также можно выбрать из выпадающих списков:



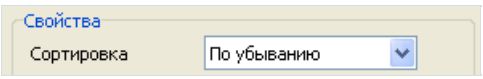
Кнопки с стрелками вверх и вниз позволяют перемещать серии диаграмм и задают им порядок отрисовки, при необходимости можно задать имя серии просто кликнув на ней мышкой.

В нашем случае (с круговой диаграммой) значения "Label" используются для отображения поясняющих надписей, а для построения диаграммы используются только значения "Pie". Можно также выбрать значение для "Color", это позволить установить для каждого "куска" диаграммы нужный цвет.

Пока закончим настройку, закрыв редактор кнопкой OK. 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:

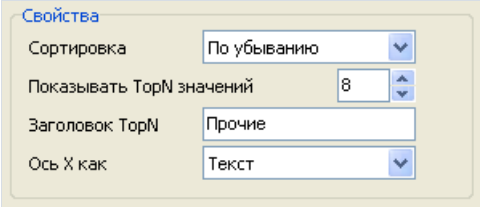


Если теперь запустить отчет, мы увидим, что данные в поясняющей таблице отсортированы.

Ограничение количества значений в диаграмме

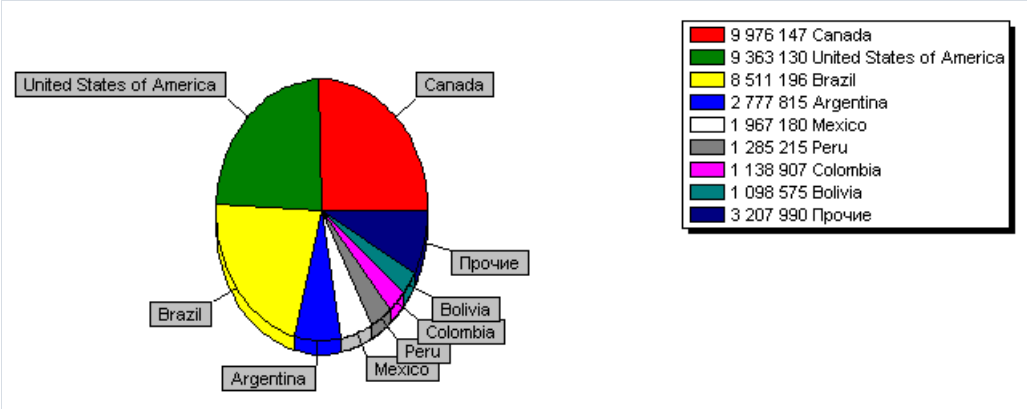
Наша диаграмма выглядит довольно перегруженной – слишком много мелких значений, которые все равно не видны на диаграмме. FastReport позволяет ограничить количество значений в диаграмме. При этом все значения, которые не уложились в заданный предел, выведутся в виде одного значения, которое представляет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

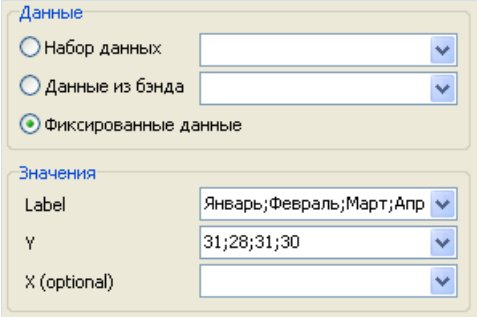
- `ColorEachPoint` – раскрашивать каждое значение разным цветом.
- `ExplodeBiggest` – выделять наибольшее значение (только для серии типа "круговая диаграмма").
- `Marks` – настройки внешнего вида поясняющих подсказок.
- `ValueFormat` – строка форматирования значений.

Следует отметить, что все возможности диаграмм полностью раскрываются в пакете TeeChart Pro, который приобретается отдельно. Данный пакет позволяет выводить большое количество разнообразных типов диаграмм и имеет удобные редакторы всей диаграммы и каждой серии, что позволяет легко настраивать внешний вид.

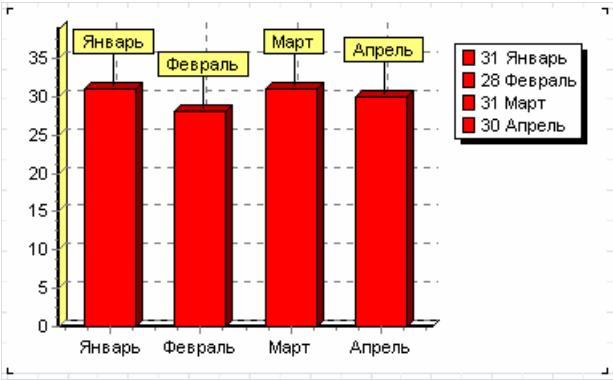
Диаграмма с фиксированными данными

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:



При запуске отчета мы увидим следующий результат:



Заполнение диаграммы из скрипта

Рассмотрим заполнение предыдущей диаграммы данными из скрипта. Для этого в редакторе диаграммы оставим пустыми поля X, Y. В скрипте отчета напишем следующее:

PascalScript:

```
begin
  Chart1.SeriesData[0].Source1 := 'Январь;Февраль;Март;Апрель';
  Chart1.SeriesData[0].Source2 := '31;28;31;30';
end.
```

C++Script:

```
{
  Chart1.SeriesData[0].Source1 = "Январь;Февраль;Март;Апрель";
  Chart1.SeriesData[0].Source2 = "31;28;31;30";
}
```

"SeriesData[0]" в данном случае позволяет задать параметры для первой серии в диаграмме. Если диаграмма имеет несколько серий, то обращаться к ним можно через "SeriesData[номер_серии]".

Печать диаграммы, построенной в Delphi

Если вы уже построили диаграмму в коде Delphi и хотите ее распечатать в отчете, вам понадобится объект "Рисунок". Расположите его в нужном месте отчета и напишите следующий обработчик события TfrxReport.OnBeforePrint:

```
procedure TForm1.frxReport1BeforePrint(Sender: TfrxReportComponent);
begin
  if Sender.Name = 'Picture1' then
    TfrxPictureView(Sender).Picture.Assign(
      Chart1.TeeCreateMetafile(False,
        Rect(0, 0, Round(Sender.Width), Round(Sender.Height))));
end;
```

где "Picture1" – имя объекта "Рисунок", "Chart1" – ваша делфийская диаграмма.

Отчеты с картами

Компонент "Карта" (MapObject) предназначен для отображения двумерных графических карт в формате ESRI shapefile (.shp/.dbf), Open Street Map (.osm), GPS-трек (GPS Exchange File, .gpx). Подробнее о форматах можно прочитать здесь:

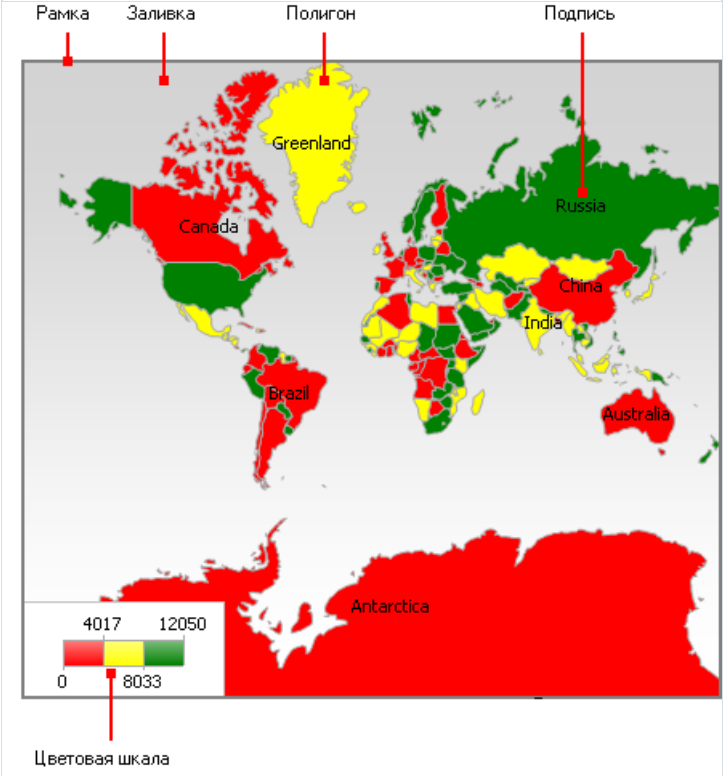
http://ru.wikipedia.org/wiki/Shapefile

https://ru.wikipedia.org/wiki/OpenStreetMap

https://ru.wikipedia.org/wiki/GPX

Элементы карты

Объект "Карта" состоит из следующих элементов:



Один объект "Карта" может отображать один или несколько слоев. Каждый слой содержит отдельную карту.

Управление отображением

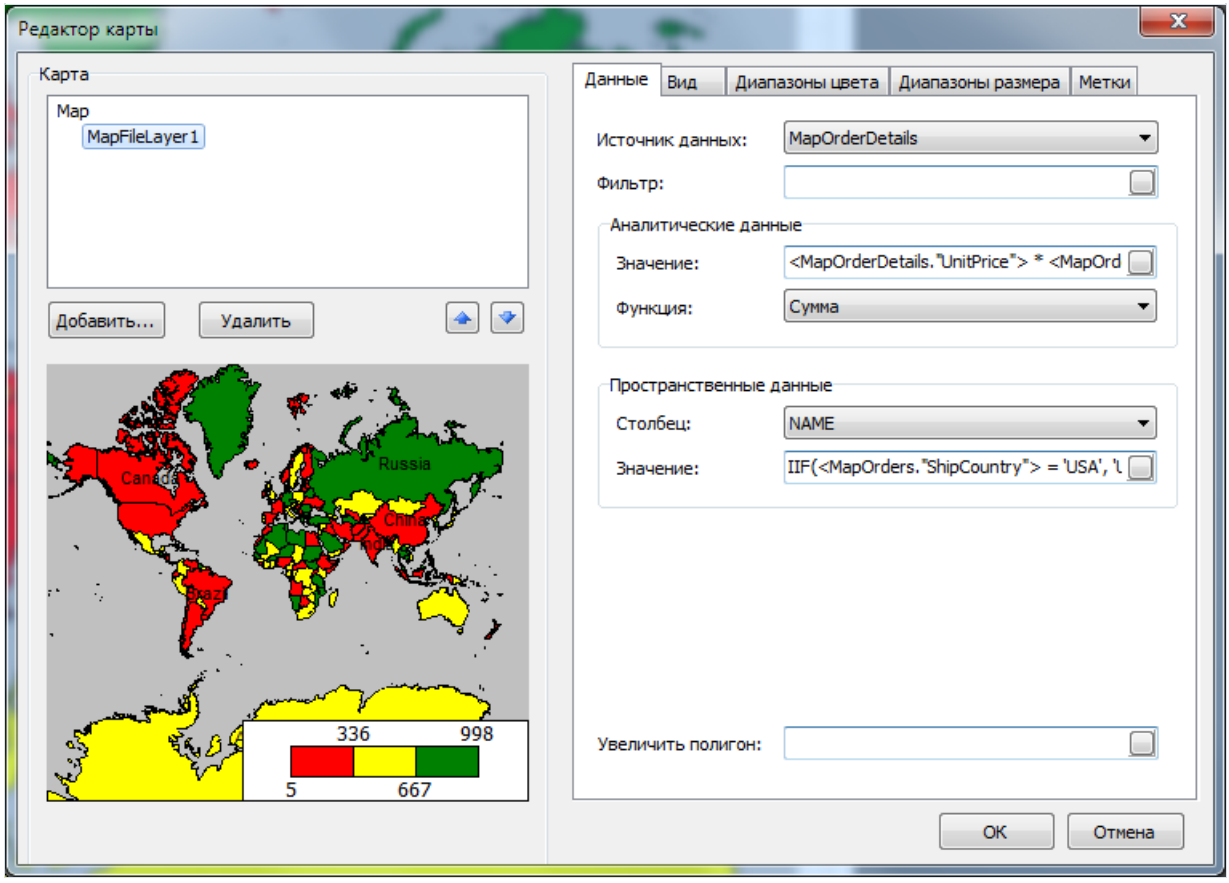
В режиме дизайнера и в окне просмотра готового отчета вы можете управлять отображением карты с помощью мыши:

- колесо мыши меняет масштаб карты;
- нажав левую кнопку мыши, можно двигать карту;
- кликнув внутри полигона, можно настроить его свойства в инспекторе объектов.

Минимальное и максимальное значения масштабирования задаются в свойствах `MinZoom`, `MaxZoom`. Эти значения можно задать в инспекторе объектов.

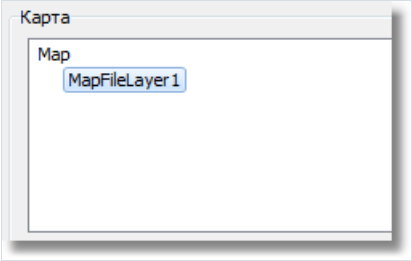
Редактор карты

Объект "Карта" содержит большое количество настроек, которыми можно управлять, вызвав редактор объекта. Для этого сделайте двойной щелчок мышью на объекте или выберите пункт "Редактировать..." в его контекстном меню:

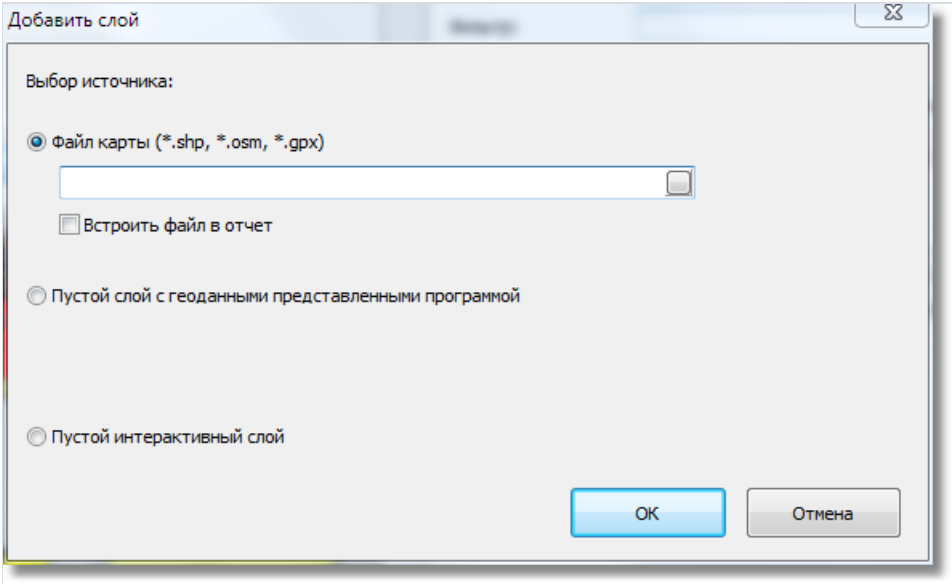


Добавление слоёв

Объект "Карта" может содержать один или несколько слоев. Список слоев отображается в левом верхнем углу редактора:



Для добавления нового слоя нажмите кнопку "Добавить...". Будет показано следующее окно:



На этом шаге нужно выбрать тип слоя:

- карта из файла (.shp/.dbf, .osm, .gpx). Это наиболее часто используемый тип карт. К примеру, вы можете напечатать карту мира и выделить цветом страны, в которых продажи были больше определенного значения;
- картографические данные из приложения. Ваше приложение должно предоставлять географические координаты (пару значений - широта и долгота), которые будут отображены в виде точки на карте. Точка может иметь подпись, а также быть разного размера и/или цвета, в зависимости от некоторых данных.

На практике этот тип карты используется в качестве второго слоя (первый слой, базовый, берется из файла карты). Например, базовый слой отображает карту какой-либо страны, а второй слой - точки с названиями городов, в которых были продажи. Размеры и цвет точки можно настроить таким образом, чтобы был понятен уровень продаж в данном городе.

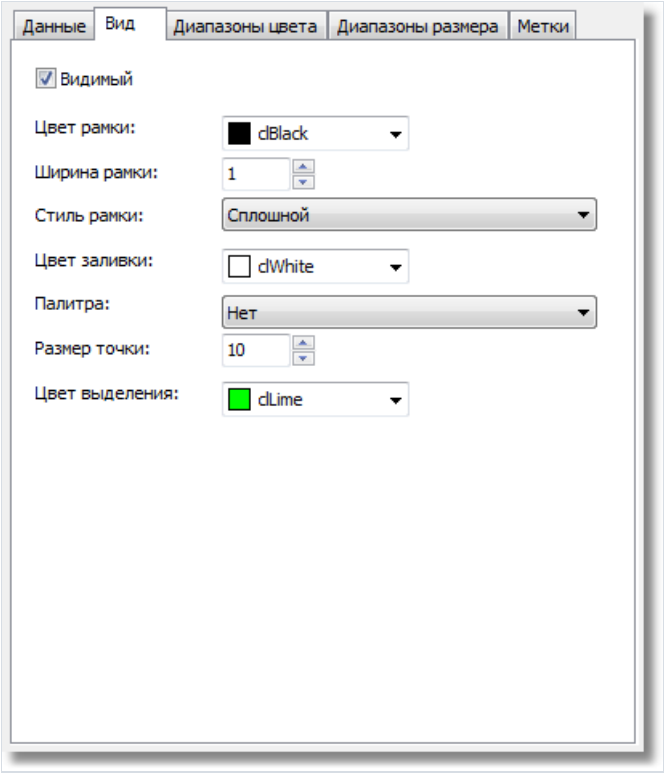
Если вы выбрали слой на основе файла карты, укажите дополнительно, как хранить картографические данные:

- данные внедряются в файл отчета. При этом отчет может сильно увеличиться в размерах.
- файл отчета ссылается на файл карты, внедрения не происходит. Этот режим полезен, если у вас есть несколько отчетов, использующих одни и те же карты.

> Карты большого объема (более 30Мб) или с большим количеством полигонов (более 20000) серьезно замедлят работу отчета.

Настройка внешнего вида

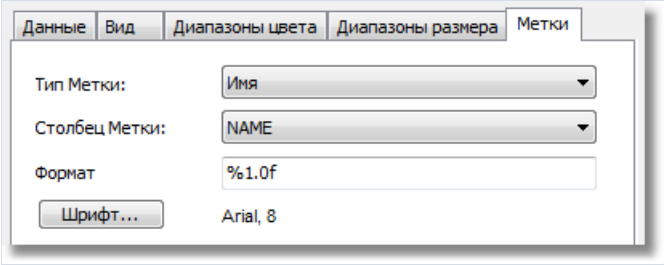
Внешний вид слоя можно настроить, выбрав слой и переключившись на закладку "Внешний вид":



Здесь можно настроить цвет и стиль рамки полигонов карты, а также выбрать цветовую палитру. Если вы настроили выделение полигонов цветом в зависимости от аналитических данных (об этом позднее), то палитра будет игнорирована.

Настройка отображаемых значений

На карте могут отображаться надписи, например, названия стран на карте мира. Вы можете настроить тип и внешний вид надписей на закладе "Надписи":



Если выбран тип слоя - карта из файла .shp, то необходимо указать поле, из которого будет взята надпись. Как правило, это поле "NAME". Для карты мира, входящей в состав демо-программы FastReport, можно выбирать из следующих полей:

`NAME` (например, Russia)

`ABBREV` (например, Rus.)

`ISO_A2` (например, RU)

`ISO_A3` (например, RUS)

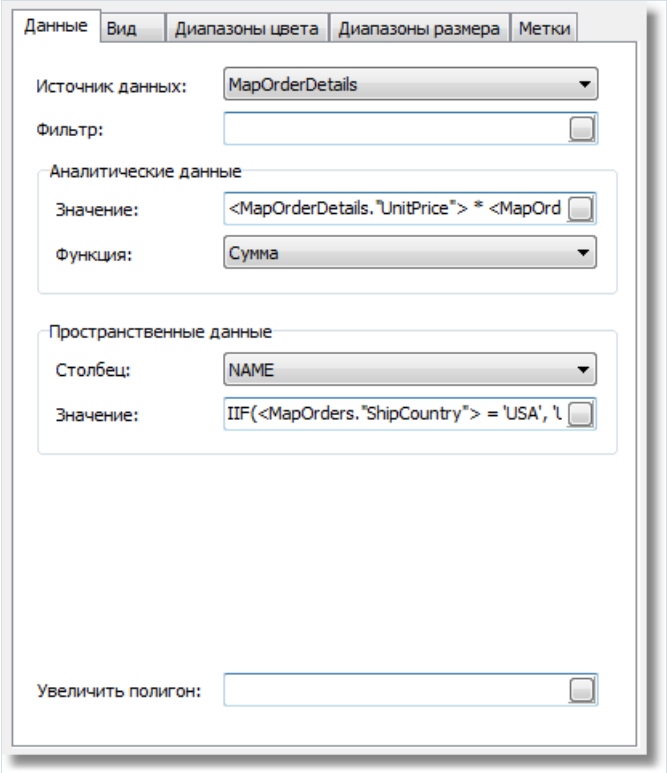
Для других карт список полей будет отличаться.

Подключение к данным

Большинство отчетов используют объект "Карта" не сам по себе, а для отображения аналитической информации. Например, это может быть объем продаж в разных странах. Для этого слой надо подключить к данным. Сделать это можно в редакторе карты, выбрав слой и переключившись на закладку "Данные".

Подключение к данным различается в зависимости от типа слоя (из файла карты, или геоданные из приложения):

Если тип слоя указан как карта из файла, закладка "Данные" выглядит следующим образом:



В этом случае приложение должно предоставить следующие данные:

- название (например, название страны);
- числовое значение (например, уровень продаж в данной стране).

Допустим, у вас имеется таблица Sales со следующими полями и данными:

```
Country  SalesTotal
USA          500000
Germany     1200000
Russia       300000
```

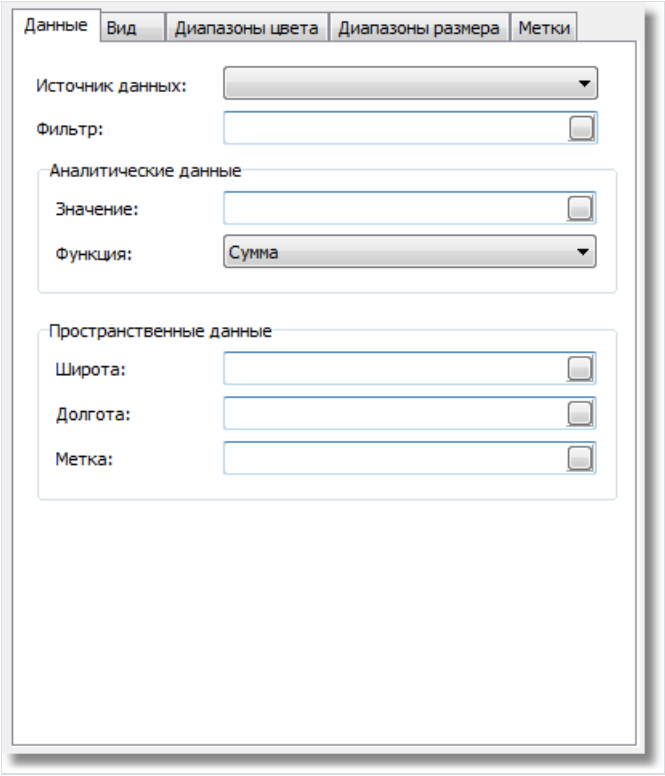
В данном случае нужно настроить данные следующим образом:

- источник данных - Sales
- пространственные данные, столбец - выберите то поле, которое в файле карты отвечает за название страны. Как правило, это поле "NAME".

- пространственные данные, значение - [Sales.Country]
- аналитические данные, значение - [Sales.SalesTotal]
- аналитические данные, функция - "Сумма". Функция используется, если для данной страны есть несколько записей с разными значениями.

Поле "Увеличить полигон" позволяет увеличить полигон с указанным именем на весь размер объекта "Карта". Например, чтобы увеличить страну Россия на карте мира, укажите в этом поле значение "Russia" (с кавычками).

Если тип слоя указан как данные, предоставляемые приложением, закладка "Данные" выглядит следующим образом:



В этом случае приложение должно предоставить следующие данные:

- пространственные данные - широта и долгота;
- метка (например, название города);
- числовое значение (например, уровень продаж в данном городе).

Допустим, у вас имеется таблица Sales со следующими полями и данными:

```
Latitude  Longitude  CityName  SalesTotal
48.13641  11.57753   Munchen        50000
50.94165  6.95505    Koln           36000
```

В этом случае нужно настроить данные следующим образом:

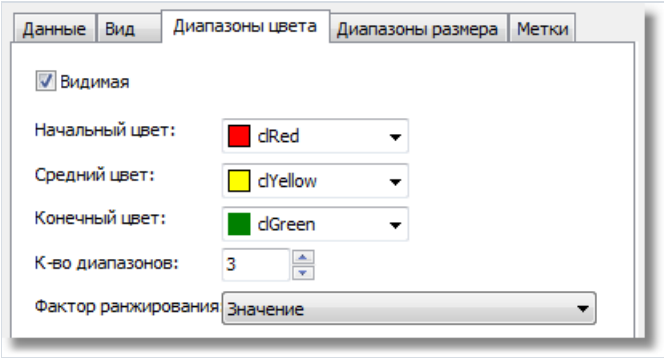
- источник данных - Sales
- пространственные данные, широта - [Sales.Latitude]
- пространственные данные, долгота - [Sales.Longitude]

- пространственные данные, метка - [Sales.CityName]
- аналитические данные, значение - [Sales.SalesTotal]
- аналитические данные, функция - "Сумма". Функция используется, если для данного города есть несколько записей с разными значениями.

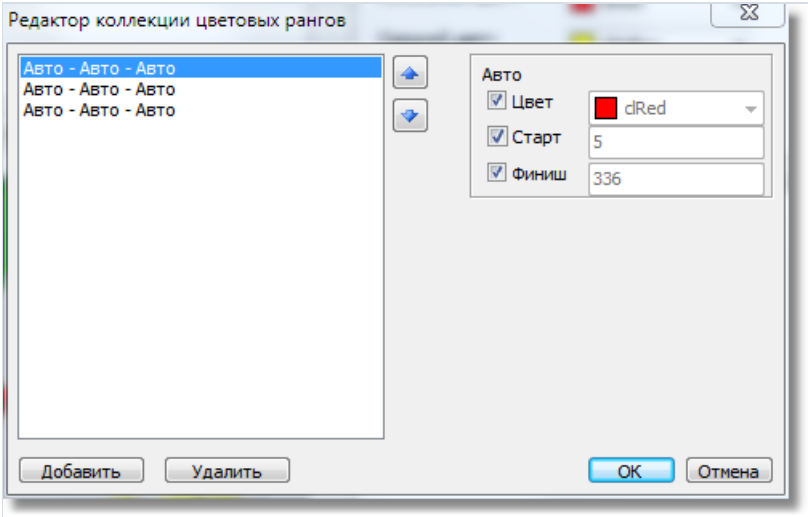
Выделение данных цветом

После того как слой подключен к данным, возникает вопрос - в каком виде выводить аналитическую информацию (например, объемы продаж в разных странах)? Самый простой способ - настроить отображение надписей так, чтобы помимо названия страны выводились и цифры продаж (см. раздел "Настройка отображаемых значений").

Однако, гораздо более наглядный способ - это раскраска стран в определенные цвета в зависимости от объема продаж. Для этого надо настроить цветовую шкалу. Сделать это можно на закладке "Диапазоны цвета":

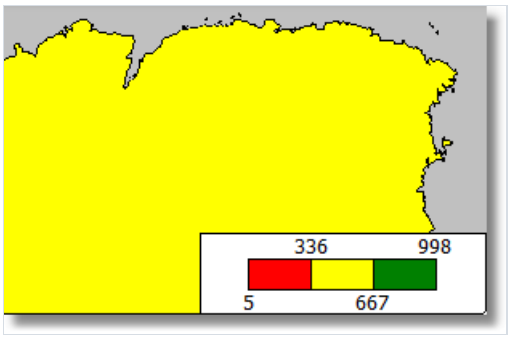


Цветовая шкала представляет собой набор значений: минимальное значение; максимальное значение; цвет. Таких наборов (диапазонов) может быть несколько. Для настройки цветовой шкалы надо указать, сколько диапазонов она содержит, после этого настроить минимальное и максимальное значение в каждом диапазоне, а также цвет:

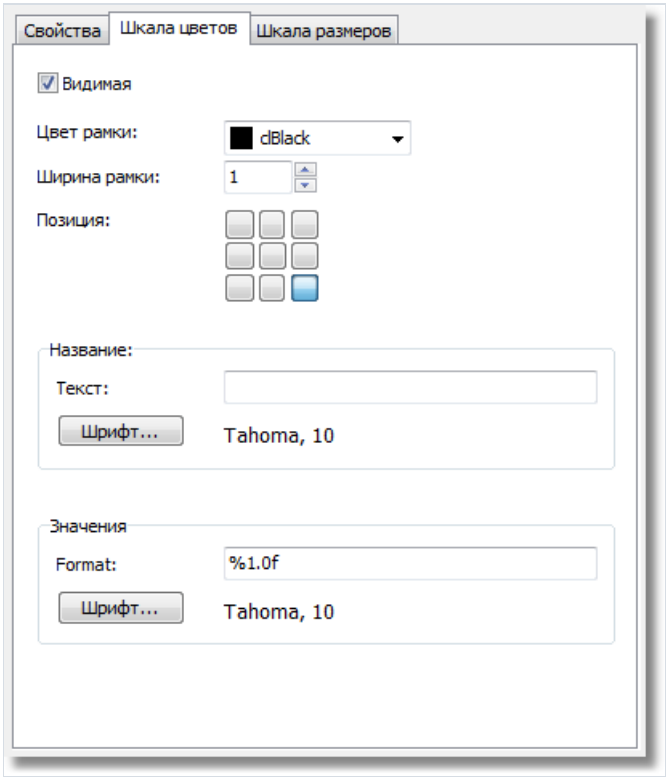


По умолчанию все значения установлены в "Авто". В этом случае FastReport рассчитает минимальное и максимальное значения для каждого диапазона автоматически, а цвет возьмет из предустановок "Начальный цвет", "Средний цвет" и "Конечный цвет". Этот режим можно использовать в большинстве случаев. Настройка "Фактор ранжирования" меняет распределение значений по шкале.

Если цветовая шкала настроена, то в нижней части карты появляется индикатор - полоска из нескольких разноцветных прямоугольников:

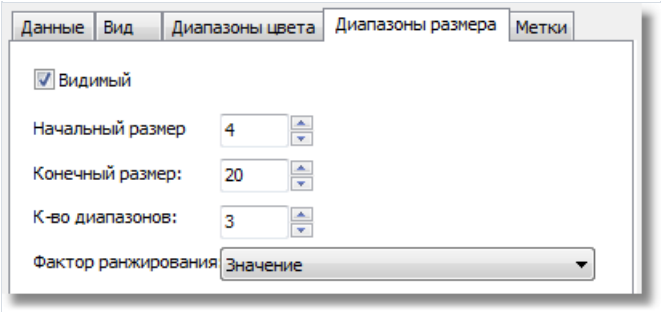


Внешний вид и расположение индикатора можно настроить, выбрав в списке слоев элемент "Карта" и переключившись на закладку "Цветовая шкала":



Выделение данных размером

Если тип слоя указан как данные, предоставляемые приложением, то данные будут отображаться в виде точки с надписью. Размер точки можно привязать к данным примерно таким же способом, как это делается при выделении цветом. Сделать это можно на закладке "Диапазоны размера":



Шкала размеров представляет собой набор значений: минимальное значение; максимальное значение; размер в пикселах. Таких наборов (диапазонов) может быть несколько. Для настройки шкалы надо указать, сколько диапазонов она содержит, после этого настроить минимальное и максимальное значение в каждом диапазоне, а также размер.

По умолчанию все значения установлены в "Авто". В этом случае FastReport рассчитает минимальное и максимальное значения для каждого диапазона автоматически, а размер возьмет из предустановок "Начальный размер", "Конечный размер". Настройка "Фактор ранжирования" меняет распределение значений по шкале.

Интерактивные отчеты

Готовый отчет FastReport можно сделать интерактивным. Это значит, что он будет реагировать на действия пользователя (а именно, на нажатие кнопки мыши). Вы можете использовать следующие виды взаимодействия:

- при нажатии на элемент отчета (бэнд или объект) выполняется какое-либо действие. Например, строится детальный отчет и показывается в отдельном окне;
- сбоку окна просмотра отображается структура отчета, которую можно использовать для навигации по отчету.

Гиперссылки

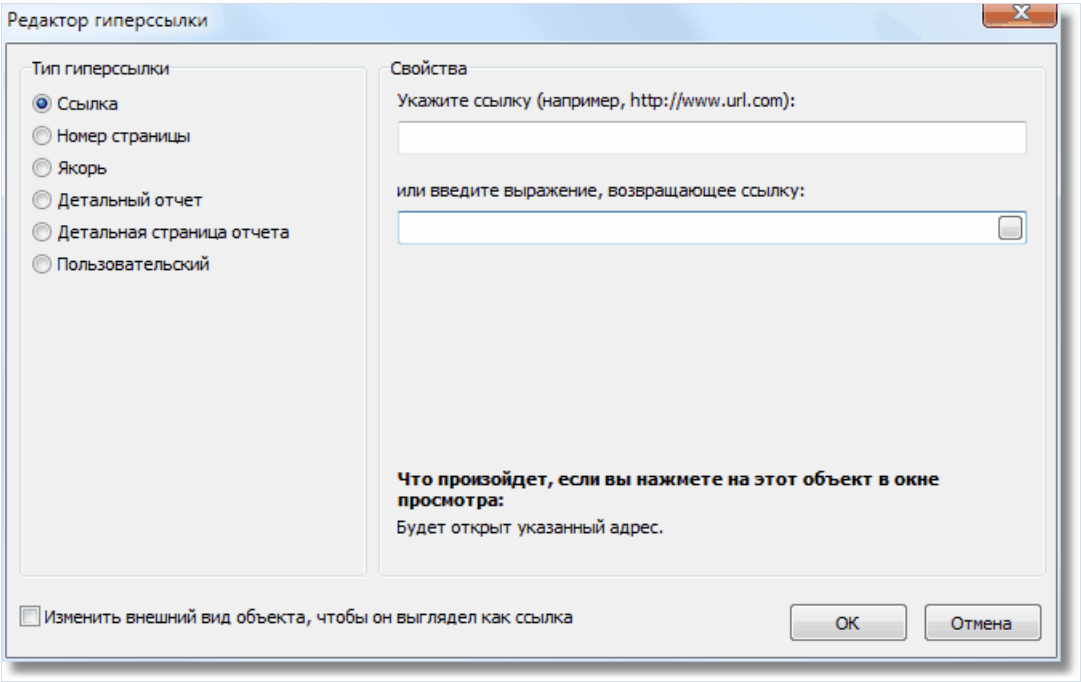
В этом разделе речь пойдет о свойстве "Гиперссылка" (`Hyperlink`), которое имеется практически у всех объектов отчета. Используя это свойство, можно определить реакцию объекта на нажатие мыши в окне предварительного просмотра.

При нажатии на такой объект может быть выполнено одно из следующих действий:

- переход на адрес URL;
- отсылка e-mail;
- выполнение какой-нибудь системной команды;
- переход на страницу отчета с указанным номером;
- переход на закладку, определенную в другом объекте отчета;
- запуск детального отчета в отдельном окне просмотра;
- реакция, определенная пользователем в скрипте.

Настройка гиперссылки

Для настройки гиперссылки выделите объект, который будет реагировать на нажатие мыши, и щелкните на нем правой кнопкой мыши. В контекстном меню выберите пункт "Гиперссылка...". Откроется окно редактора гиперссылки:



Выбрать тип гиперссылки можно в левой части окна. После того как вы настроили ссылку, можно отметить флажок "Изменить внешний вид объекта..." внизу окна. Внешний вид объекта изменится при этом следующим образом:

- будет установлен синий цвет текста и подчеркивание;
- будет установлена форма курсора (свойство `Cursor`) в виде руки.

В некоторых случаях гиперссылку нужно показывать в окне просмотра, но не нужно печатать на принтере. Это легко сделать, настроив свойство объекта `Visibility`. Это можно сделать в окне "Свойства".

Ссылка на адрес URL

Используя этот тип ссылки, вы можете:

- перейти на заданный адрес Интернет;
- выполнить какую-либо системную команду, например, "mailto:" для отправки электронной почты.

Вы можете указать значение ссылки двумя способами:

- указать значение напрямую, например, "https://быстрыеотчеты.рф/";
- указать выражение, которое возвращает значение ссылки. Это выражение будет вычислено в момент построения отчета, когда обрабатывается данный объект.

Ссылка на номер страницы

Используя этот тип ссылки, вы можете организовать навигацию по страницам готового отчета. Чаще всего используется переход на первую страницу отчета. Для этого в качестве значения ссылки укажите номер страницы (в данном случае 1).

Вы можете указать номер страницы двумя способами:

- указать номер напрямую, например, 1;
- указать выражение, которое возвращает номер страницы. Это выражение будет вычислено в момент построения отчета, когда обрабатывается данный объект.

Ссылка на якорь

Используя этот тип ссылки, вы можете переходить на якорь, определенный программно. Якорь имеет имя и определенную позицию в готовом отчете (номер страницы и позиция на странице). При переходе на якорь по его имени вы перемещаетесь на указанную позицию.

Чтобы использовать этот тип ссылки, вы должны добавить якорь программно вызовом метода `Engine.AddAnchor`. Подробнее см. раздел Скрипт/Якоря. Имя якоря нужно указать в окне настройки гиперссылки. Это можно сделать двумя способами:

- указать имя якоря напрямую;
- указать выражение, которое вернет имя якоря. Например, это может быть поле данных. Это выражение будет вычислено при запуске отчета, в момент обработки данного объекта.

Ссылка на детальный отчет

Используя этот тип ссылки, вы можете выполнять другой отчет и показывать его в отдельном окне просмотра.

В настройках данного типа ссылки вы должны указать следующую информацию:

- имя детального отчета;
- имя переменной в отчете, которой будет передано значение из гиперссылки;
- значение гиперссылки, которое надо передать в переменную отчета.

При нажатии на ссылку произойдет следующее:

- будет загружен указанный отчет;
- в переменную отчета будет передано значение гиперссылки;
- отчет будет построен и запущен в отдельном окне просмотра.

Значение переменной отчета можно указать двумя способами:

- указать значение напрямую;
- указать выражение, которое вернет значение. Это выражение будет вычислено при запуске отчета, в момент обработки данного объекта.

Ссылка на детальную страницу

Этот тип ссылки работает аналогичным образом, за исключением того, что в качестве детального отчета используется другая страница в текущем отчете. Для этого ваш отчет должен содержать как минимум две страницы: одну с основным отчетом, другую - с детальным.

В настройках данного типа ссылки вы должны указать следующую информацию:

- имя страницы в этом отчете;
- имя переменной в отчете, которой будет передано значение из гиперссылки;
- значение гиперссылки, которое надо передать в переменную отчета.

При нажатии на ссылку произойдет следующее:

- в переменную отчета будет передано значение гиперссылки;
- указанная страница отчета будет построена и показана в отдельном окне просмотра.

Значение переменной отчета можно указать двумя способами:

- указать значение напрямую;
- указать выражение, которое вернет значение. Это выражение будет вычислено при запуске отчета, в момент обработки данного объекта.

Когда вы выбираете страницу отчета, ее свойство `Visible` сбрасывается в false. Это означает, что при построении основного отчета эта страница будет пропущена.

Прочая ссылка

Используя этот тип ссылки, вы можете определить собственную реакцию на нажатие мыши. Для этого используется обработчик события "OnClick" объекта, который может быть написан в скрипте отчета.

Отчеты для матричных принтеров

Ранее мы рассматривали отчеты, которые предназначены для печати на обычных принтерах (струйном, лазерном). Печать такого отчета на матричном принтере займет очень много времени. FastReport позволяет создавать специальные отчеты для матричного принтера, где на печать выводятся только символы стандартного шрифта, без графических элементов. За счет этого печать производится очень быстро.

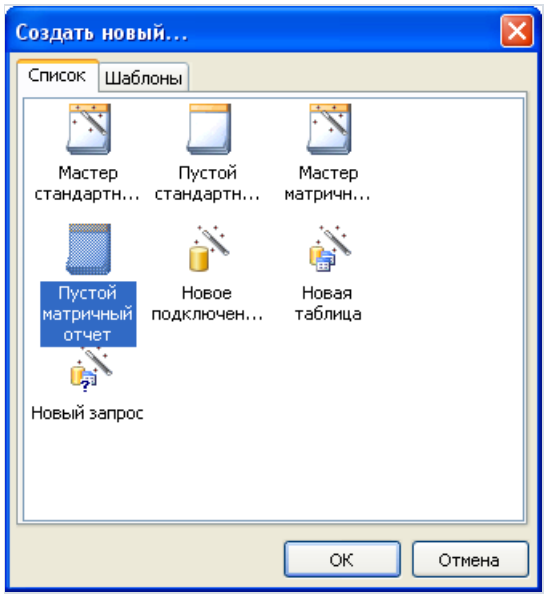
Рассмотрим создание отчета типа «Список», предназначенного для матричной печати. Ранее мы создавали такой отчет, см. главу Отчет "Список клиентов". Для отчета нам понадобятся те же данные.

Итак, создадим новый проект в Delphi, на форму положим компоненты `TTable`, `TfrxDBDataSet`, `TfrxReport`, `TfrxDotMatrixExport` и настроим их свойства:

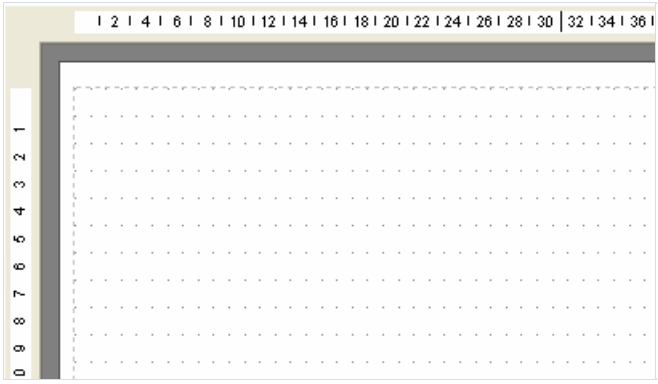
```
TTable:
DatabaseName = 'DBDEMOS'
TableName = 'Customer.db'

TfrxDBDataSet:
DataSet = Table1
UserName = ‘Customers’
```

Зайдем в дизайнер отчета и выберем пункт меню “Файл/Новый...”. Откроется окно, в котором перечислены мастера отчетов. Нам нужно выбрать мастер «Пустой матричный отчет»:



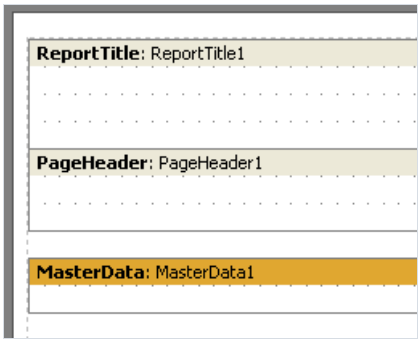
При нажатии кнопки OK вы увидите пустую страницу, которая размечена под матричный шрифт:



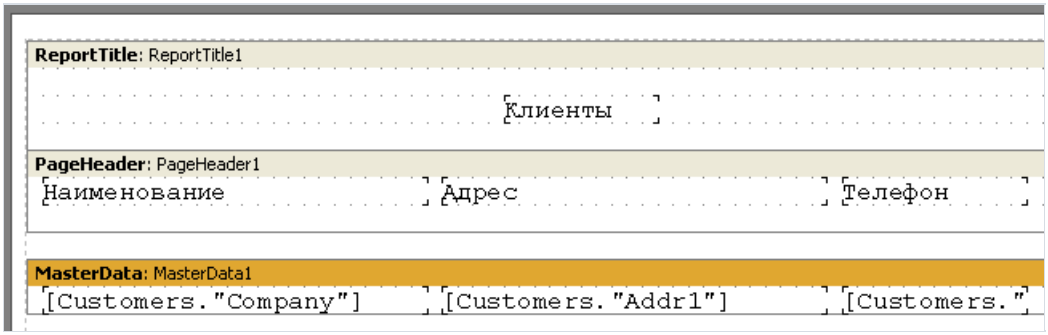
Список объектов, доступных для вставки, изменился – теперь это объекты «Бэнд», «Текст», «Линия», «ESC-Команда», «Вложенный отчет» и «Кросс-таблица». Другие объекты в матричном отчете использовать нельзя.



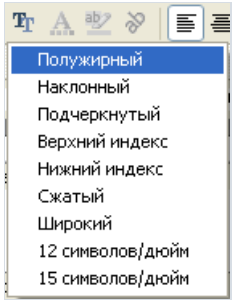
Разместим на странице отчета бэнды Report title, Page header, Master data:



На бэндах разместим объекты «Текст» следующим образом:



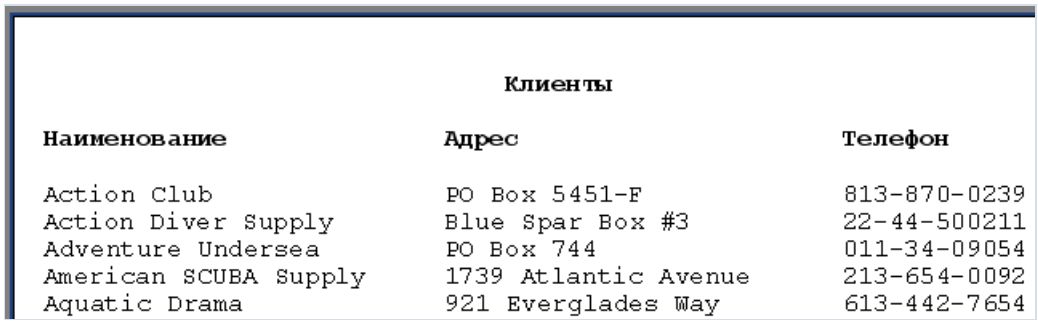
Принцип размещения матричных объектов такой же, как и в обычном отчете. Отличие в том, что объекты жестко привязаны к сетке, и для них нельзя задать другой размер шрифта или его цвет. А вот некоторые атрибуты шрифта менять можно, для этого выделите объект «Текст» и нажмите кнопку на панели инструментов:



Как видите, здесь можно задать атрибуты шрифта, специфичные для матричной печати. Эти атрибуты есть у страницы отчета и у всех матричных объектов, кроме бэндов.

> В дизайнере и предварительном просмотре отображаются только атрибуты «Полужирный», «Наклонный», «Подчеркнутый». На печать выводится полный набор атрибутов.

Изменим вид нашего отчета, задав стиль «Полужирный» для заголовков. Отчет готов, можно запускать предварительный просмотр:



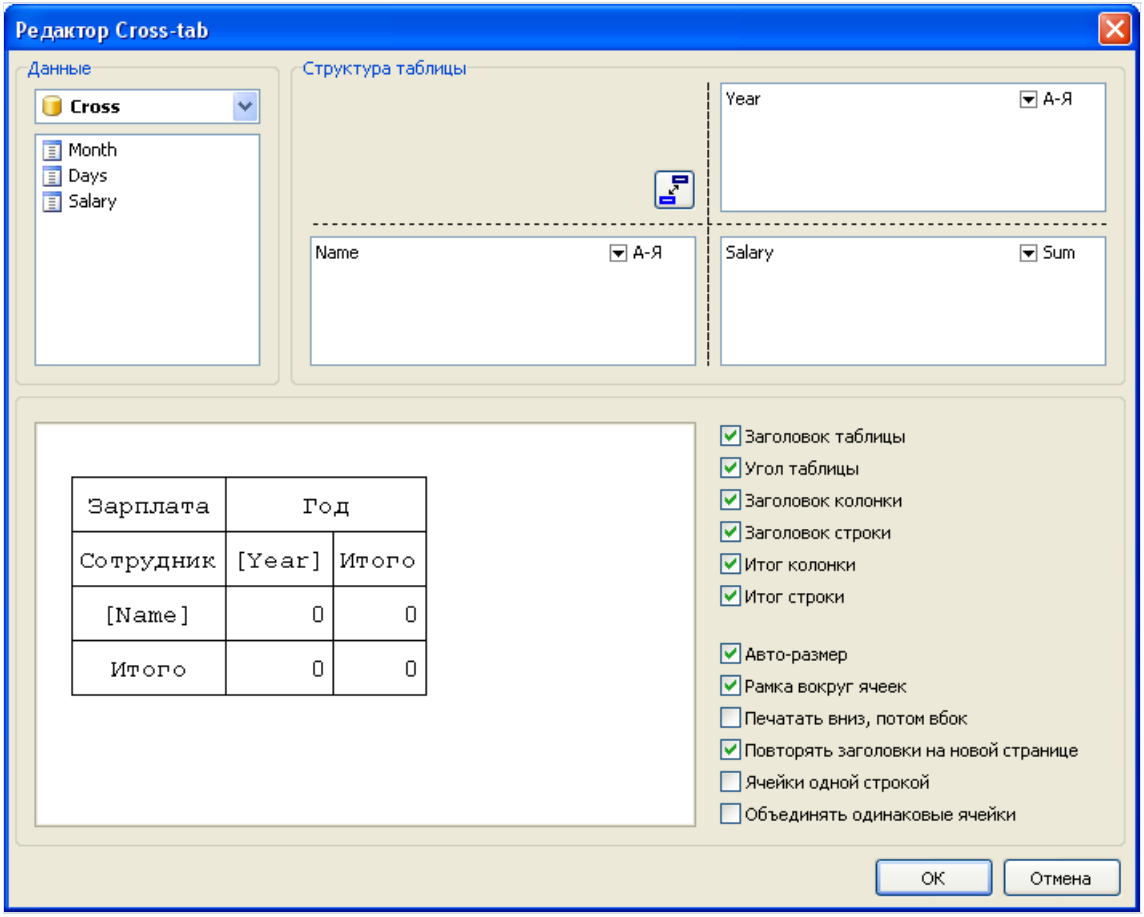
Клиенты

Наименование	**Адрес**	**Телефон**
Action Club	PO Box 5451-F	813-870-0239
Action Diver Supply	Blue Spar Box #3	22-44-500211
Adventure Undersea	PO Box 744	011-34-09054
American SCUBA Supply	1739 Atlantic Avenue	213-654-0092
Aquatic Drama	921 Everglades Way	613-442-7654

Кросс-таб отчет в матричном виде

Количество объектов для матричного отчета ограничено только теми, что могут быть отображены в текстовом виде. Среди них – объект «Кросс-таб». Рассмотрим создание простого кросс-отчета, аналогичного ранее построенному в главе Таблица с составными заголовками.

Для создания матричного отчета выполняем те же шаги – вызываем мастер «Пустой матричный отчет». На страницу отчета кладем компонент «Кросс-таблица БД» и настраиваем его структуру:

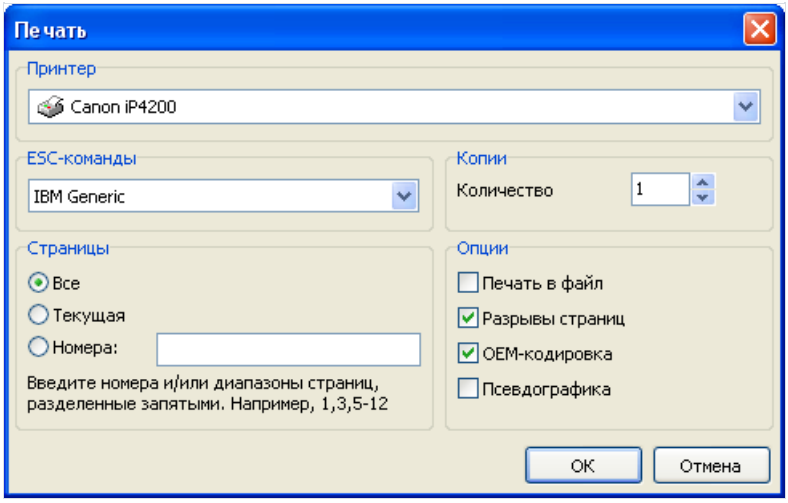


Как видно, редактор показывает структуру будущей таблицы в матричном режиме. Стиль ячеек кросса можно задать, пользуясь кнопкой в дизайнере. В остальном работа ничем не отличается от ранее описанной. На экране построенный отчет будет выглядеть следующим образом:

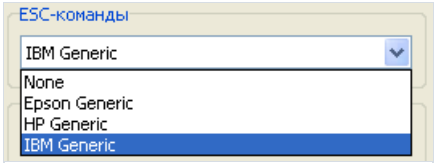
Зарплата	Год				
Сотрудник	1999	2000	2001	2002	Итого
Ann	3300	2700	3100	1700	10800
Ben	3900	2100		1800	7800
Catherine	6100	3200			9300
Den		3999	8100		12099
Итого	13300	11999	11200	3500	39999

Печать матричных отчетов

Для того, чтобы распечатать матричный отчет в текстовом режиме (т.е. с максимальной скоростью), необходимо на форму вашего проекта положить компонент `TfrxDotMatrixExport`  из палитры компонент "FastReport". Этот компонент отвечает за преобразование отчета в текстовый вид и последующую печать в текстовом режиме. При этом он подменяет стандартный диалог печати:



Диалог печати похож на стандартный, но в него добавлена специфика матричных принтеров. Так, перед печатью необходимо выбрать систему команд принтера (ESC-команды). Доступны следующие команды:

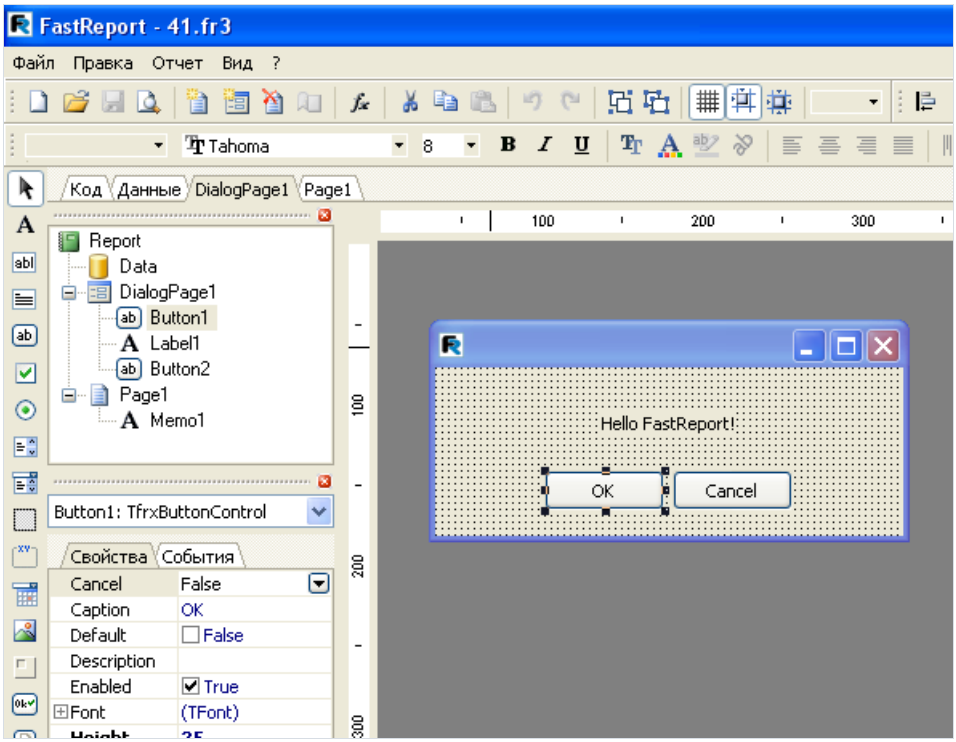


Также имеется набор флажков, задающих опции матричной печати:

- Печать в файл – определяет, нужно ли направлять поток печати в файл на жестком диске. Если флажок установлен, то при печати появится окно с запросом имени файла;
- Разрывы страниц – определяет, надо ли посылать управляющую команду «Разрыв страницы» по достижении конца страницы. Если флажок снят, это позволяет печатать на рулонной бумаге;
- OEM-кодировка – определяет, надо ли делать перекодировку символов;
- Псевдографика – определяет, как рисовать вертикальные и горизонтальные линии. Если флажок отключен, линии рисуются с помощью символов -, |, +.

Диалоговые формы

В отчете, помимо обычных страниц отчета, может быть несколько диалоговых форм. 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:



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" модуль "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.

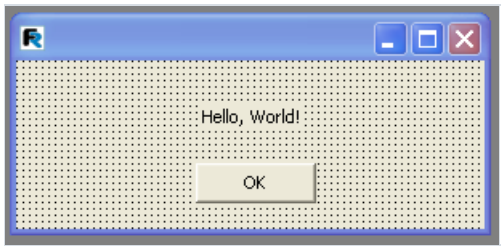
Элемент	Название	Описание
	TfrxBevelControl	Элемент управления предназначен для оформления диалоговой формы.
	TfrxImageControl	Элемент управления представляет собой картинку в формате `BMP`, `ICO`, `WMF`, `EMF`.

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:

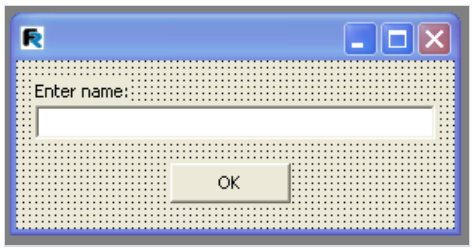


Если нажать кнопку OK, отчет будет построен и показан. Если же закрыть окно кнопкой X, отчет строиться не будет.

Таков алгоритм работы FastReport: при наличии в отчете диалоговых форм отчет будет построен только в том случае, если каждая форма была закрыта кнопкой OK, 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:



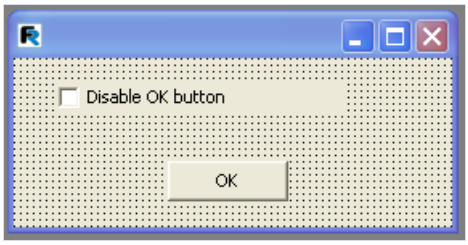
На странице отчета расположим объект "Текст" со следующим текстом внутри:

```
You've 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 напишем следующий скрипт:

PascalScript:

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

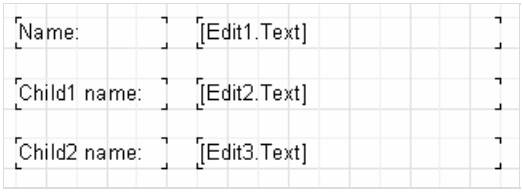
C++ Script:

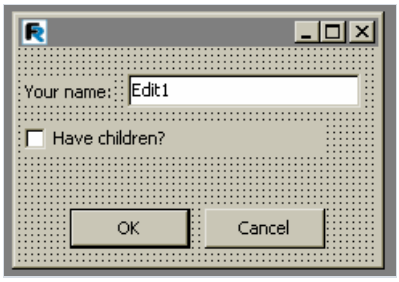
```
void CheckBox1OnClick(TfrxComponent Sender)
{
  Button1.Enabled = !CheckBox1.Checked;
}
```

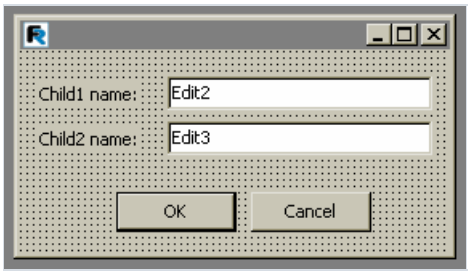
Как видим, код ничем не отличается от того, что мы привыкли видеть в Delphi. Запустив отчет, увидим, что кнопка реагирует на изменение состояния флажка.

Несколько диалоговых форм

Рассмотрим, как работает отчет с двумя диалоговыми формами. Создадим отчет с двумя диалогами и одной страницей:







У кнопок OK и Cancel настроим свойства `ModalResult` (mrOk и mrCancel соответственно).

Теперь запустим отчет. Нам будет сначала предложено ответить на вопросы из первого диалога (имя, есть ли дети), затем, при нажатии кнопки OK – из второго (имена детей).

После нажатия кнопки OK во втором диалоге отчет будет построен. Так работает ядро FastReport – при наличии нескольких диалоговых окон они показываются в порядке их создания, причем каждый последующий диалог будет показан после того, как в предыдущем диалоге была нажата кнопка OK (со свойством `ModalResult` = mrOk). Если какой-нибудь из диалогов будет отменен (кнопкой Cancel или крестиком на заголовке окна), построение отчета будет прекращено.

Управление формами отчета

В предыдущем примере обе формы диалога показываются независимо от того, отметили мы галочку "Have children" или нет. Покажем, как скрыть второй диалог в случае, если этот флажок снят. Для этого создадим обработчик OnClick у кнопки OK на первой форме диалога (сделайте двойной щелчок на кнопке, чтобы создать обработчик):

PascalScript:

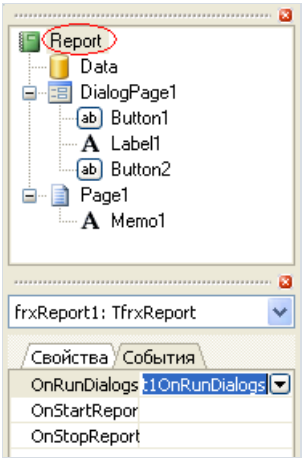
```
procedure Button1OnClick(Sender: TfrxComponent);
begin
  DialogPage2.Visible := CheckBox1.Checked;
end;
```

C++Script:

```
void Button1OnClick(TfrxComponent Sender)
{
  DialogPage2.Visible = CheckBox1.Checked;
}
```

Этот код скрывает вторую диалоговую форму (DialogPage2), если флажок не отмечен. Если запустить отчет на исполнение, мы увидим, что все работает как надо.

Другой способ управления формами заключается в использовании события отчета OnRunDialogs. Для создания обработчика этого события выберите объект `Report` в дереве отчета или в инспекторе объектов, и переключитесь на закладку "События" в инспекторе. Двойной щелчок на событии OnRunDialogs создаст нужный обработчик:



В обработчике напишем следующий код:

PascalScript:

```
procedure frxReport1OnRunDialogs(var Result: Boolean);
begin
  Result := DialogPage1.ShowModal = mrOk;
  if Result then
  begin
    if CheckBox1.Checked then
      Result := DialogPage2.ShowModal = mrOk;
  end;
end;
```

C++Script:

```
void frxReport1OnRunDialogs(bool &Result);
{
  Result = DialogPage1.ShowModal == mrOk;
  if (Result)
  {
    if (CheckBox1.Checked)
      Result = DialogPage2.ShowModal == mrOk;
  }
}
```

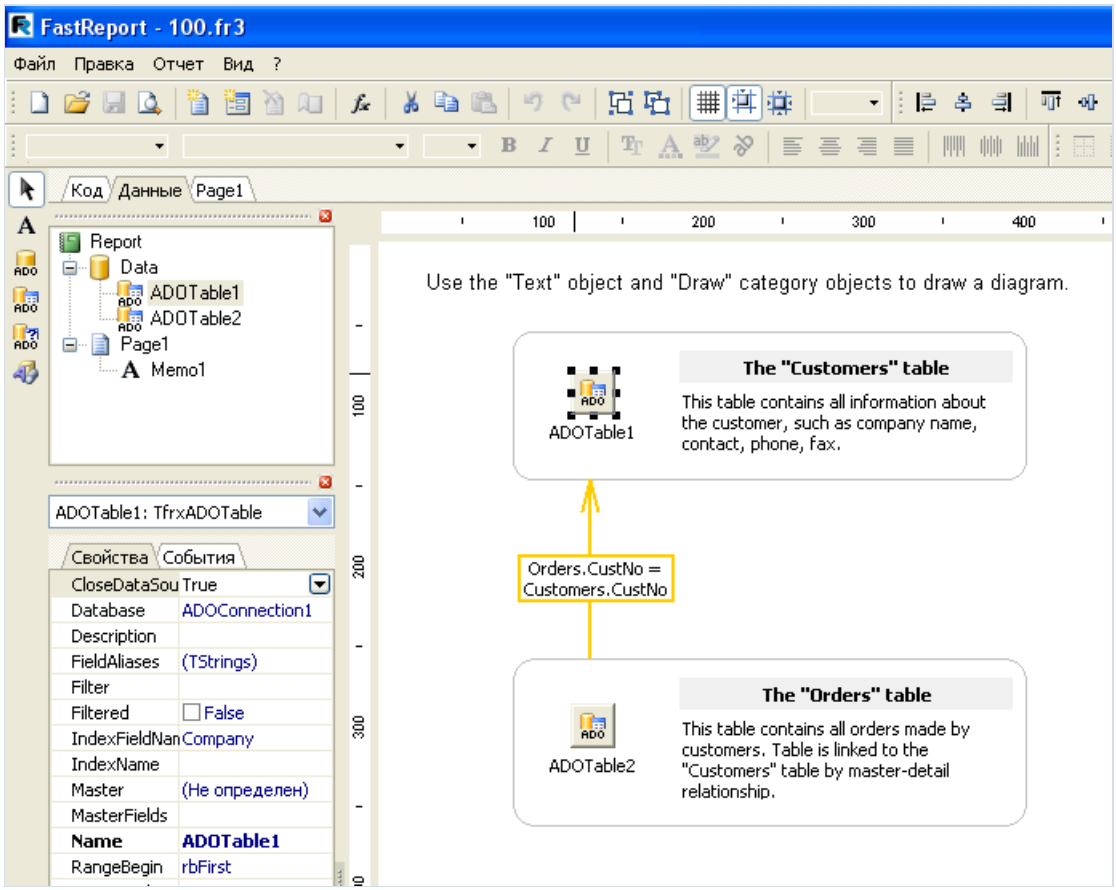
Обработчик работает следующим образом:

- Мы показываем первый диалог.
- Если он был закрыт кнопкой OK, смотрим состояние флажка CheckBox1.
- Если состояние флажка = Checked, показываем второй диалог.
- Если он был закрыт кнопкой OK, устанавливаем Result = True.
- Если обработчик возвращает Result = True, отчет строится; если Result = False, отчет останавливается.

Компоненты доступа к данным

Большинство отчетов, как правило, основано на данных из БД. Для доступа к таким данным Delphi предоставляет эффективные механизмы, которые 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 источники данных в run-time. В FastReport принципы создания источников данных 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

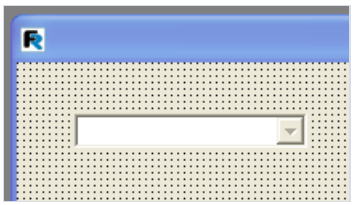
В этой главе мы рассмотрим использование компонентов для доступа к данным с помощью ADO. Также вы можете использовать компоненты BDE, IBX, DBX, FIB и множество сторонних компонентов для работы FR с движками БД.

Компоненты ADO подключаются при использовании в проекте компонента `TfrxADOComponents`  из палитры FastReport. При этом в панели объектов дизайнера появляются следующие объекты: `TfrxDBLookupComboBox`, `TfrxADOTable`, `TfrxADOQuery`, `TfrxADODataBase`. Эти объекты по своему назначению аналогичны соответствующим компонентам Delphi (`TDBLookupComboBox`, `TADOTable`, `TADOQuery`, `TADOConnection`).

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

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.
AutoOpenDataSet	Если свойство установлено в True, то присоединенный источник данных будет открыт автоматически после события OnActivate диалога

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.

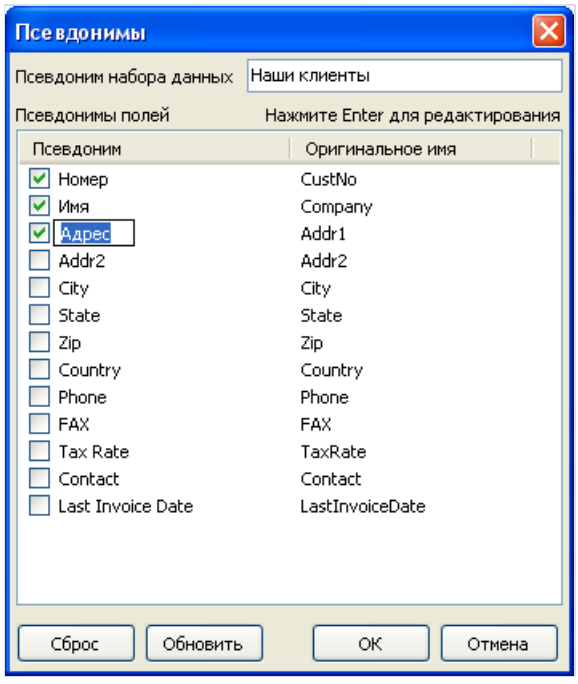
TfrxADOTable

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

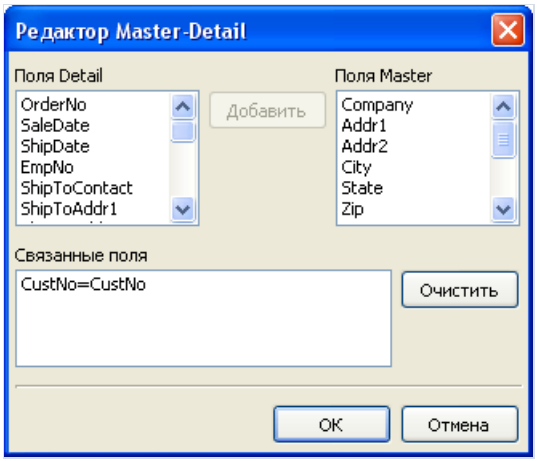
Свойство	Описание
Database	Имя подключения к БД (имя компонента `TfrxADODatabase`).
FieldAliases	Позволяет задать пользовательские имена полей.
Filter	Выражение для фильтрации записей.
Filtered	Определяет, надо ли применять фильтр.
IndexFieldNames	Имена полей, образующих индекс.
IndexName	Имя вторичного индекса.
MasterFields	Поля, связанные с master-набором данных.
Master	Мастер-набор данных.
TableName	Имя таблицы БД.
UserName	Алиас (пользовательское имя) набора данных.

Назначения свойств компонента аналогичны свойствам Delphi `TADOTable`. Для подключения компонента к таблице БД достаточно заполнить свойства `Database` 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 визуально 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.

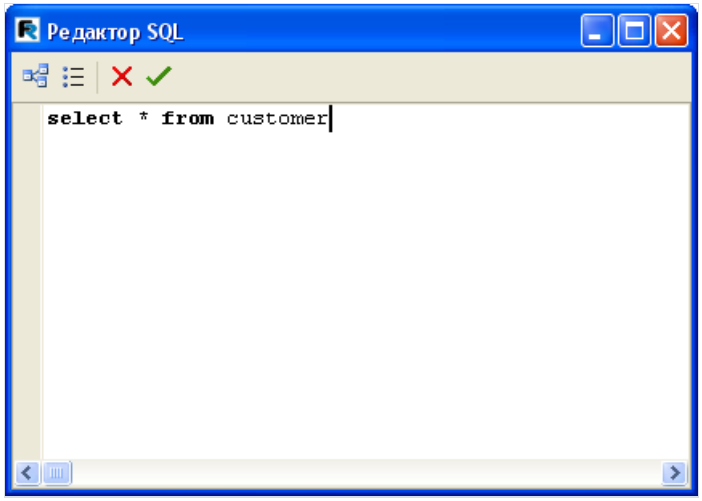
Для связи полей наборов выделите поле из списка слева (detail набор), затем поле из списка справа (master набор), и нажмите кнопку "Добавить". При этом связка полей переместится в нижний список. Чтобы очистить нижний список, воспользуйтесь кнопкой "Очистить". Связываемые поля должны иметь одинаковый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

TfrxADOQuery

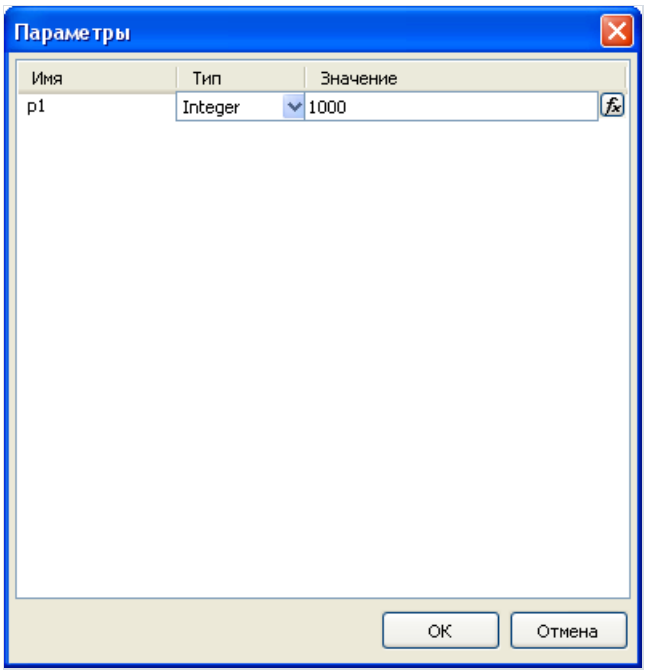
Компонент предназначен для выполнения SQL-запросов к БД. Компонент имеет следующие свойства:

Свойство	Описание
Database	Имя подключения к БД (имя компонента `TfrxADODatabase`).
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.
UserName	Алиас (пользовательское имя) набора данных.
IgnoreDupParams	Если True, то имена параметров запроса не будут дублироваться в редакторе параметров.

Свойства `Database`, `FieldAliases`, `Filter`, `Filtered`, `Master` аналогичны вышеописанным свойствам компонента `TfrxADOTable`. 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).

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

TfrxADODataBase

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

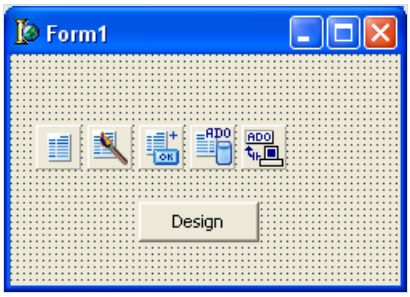
Свойство	Описание
Connected	Если True, активизирует подключение.
DatabaseName	Строка подключения к БД.
LoginPrompt	Определяет, надо ли запрашивать у пользователя пароль при подключении к БД.

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

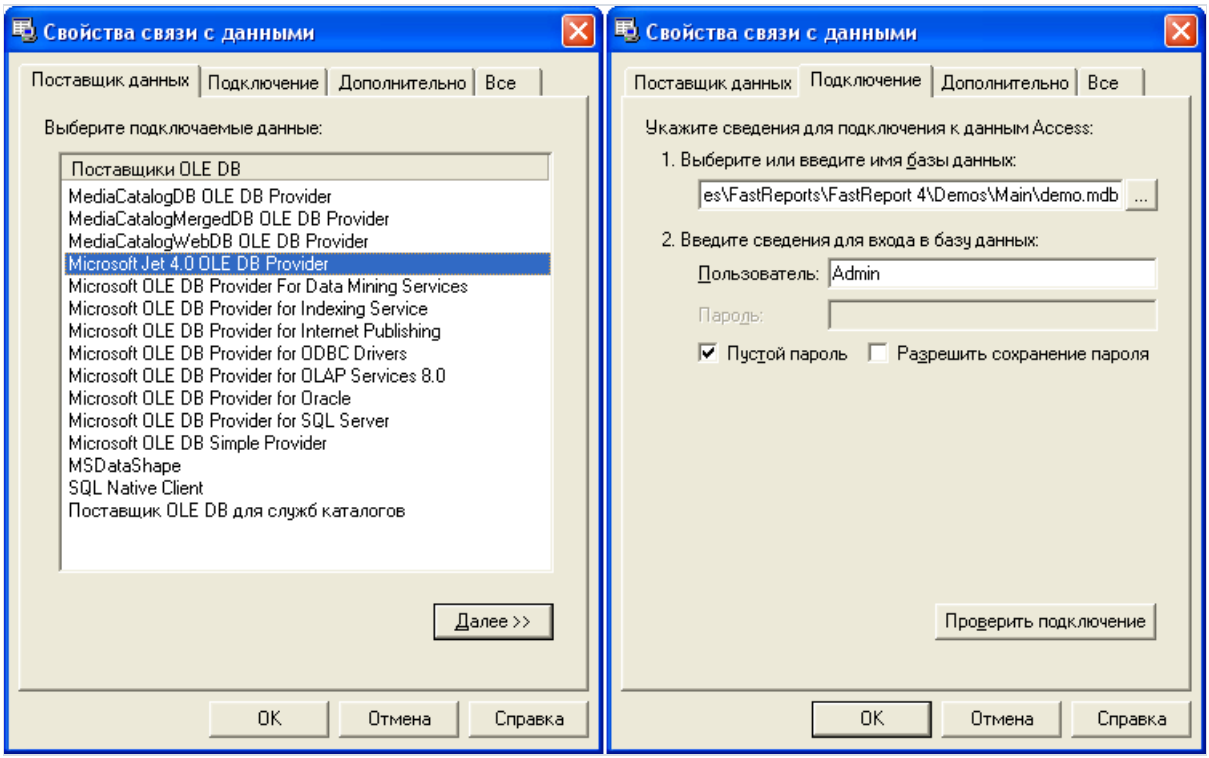
Построение отчетов

Рассмотрим построение простого отчета, содержащего компоненты доступа к данным. В качестве данных будем использовать демонстрационную базу данных из поставки FastReport - "FR\Demos\Main\demo.mdb".

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`, `TfrxADOComponents`, `TADOConnection`.



Настройте подключение к базе данных. Для этого сделайте двойной щелчок на компоненте `TADOConnection`, выберите "Build connection string" и выберите провайдера и базу данных, как показано на рисунке:



После этого закройте окно кнопкой OK и настройте свойства следующих компонентов:

```
ADOConnection1:
LoginPrompt = False

frxADOComponents1:
DefaultDatabase = ADOConnection1
```

Для кнопки "Design" определите следующий обработчик:

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

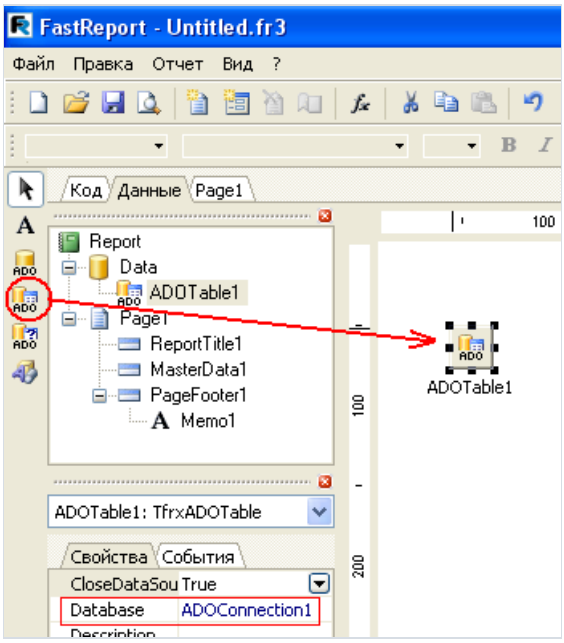
После этого скомпилируйте и запустите проект. Это все, что требуется для создания end-user дизайнера отчетов.

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 проделайте следующие шаги.

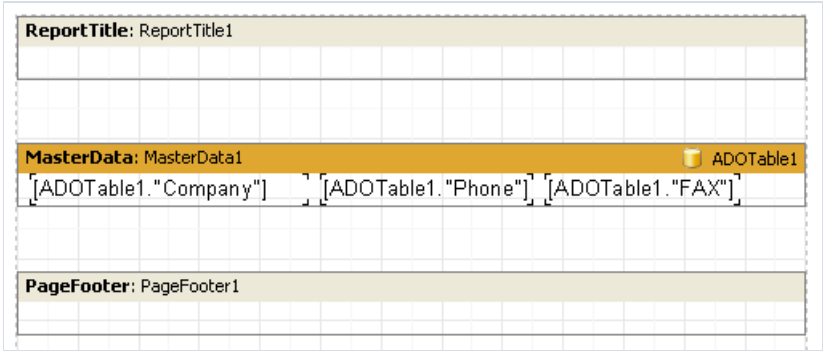
Нажмите кнопку "Новый отчет" на панели инструментов дизайнера. При этом FastReport создаст пустой отчет, содержащий страницы "Код", "Данные", "Page1". Переключитесь на страницу "Данные" и положите на страницу компонент "Таблица ADO":



Обратите внимание, что у компонента уже заполнено свойство `Database` - оно указывает на нашу базу данных. Это произошло потому, что мы указали ее в свойстве `DefaultDatabase` компонента `TfrxADOComponents`. Нам осталось только выбрать таблицу:

```
TableName = 'Customer'
```

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.

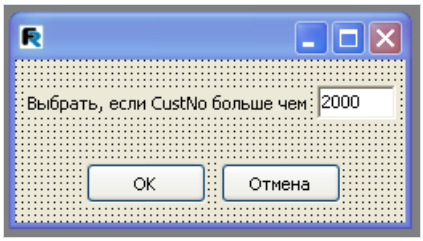
Отчет с запросом параметров

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

Создайте новый отчет. Переключитесь на страницу "Данные" и добавьте объект "Запрос ADO". Вызовите его редактор и напишите следующий текст запроса:

```
select * from Customer where CustNo > :p1
```

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



Настройте свойства компонентов:

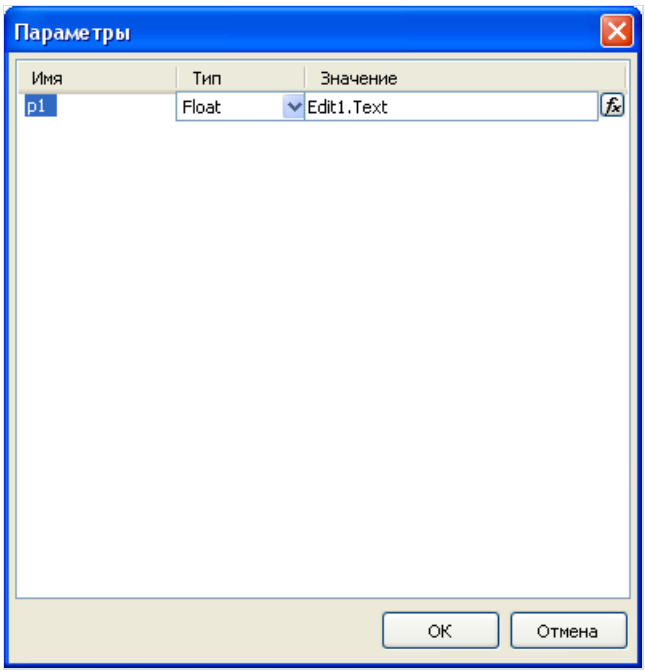
```
Label1:
Caption = 'Выбрать, если CustNo больше чем'

Edit1:
Text = '2000'

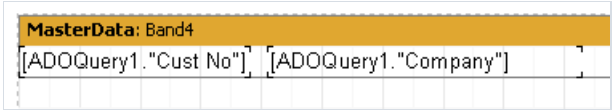
Button1:
Caption = 'OK'
ModalResult = mrOk

Button2:
Caption = 'Отмена'
ModalResult = mrCancel
```

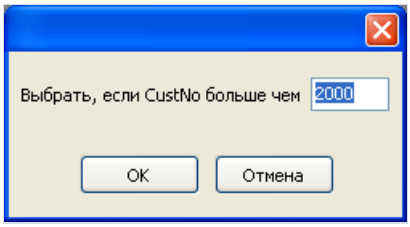
Откройте редактор свойства `Params` компонента Query и настройте параметр:



После этого перейдите на страницу с формой отчета и создайте следующий отчет:

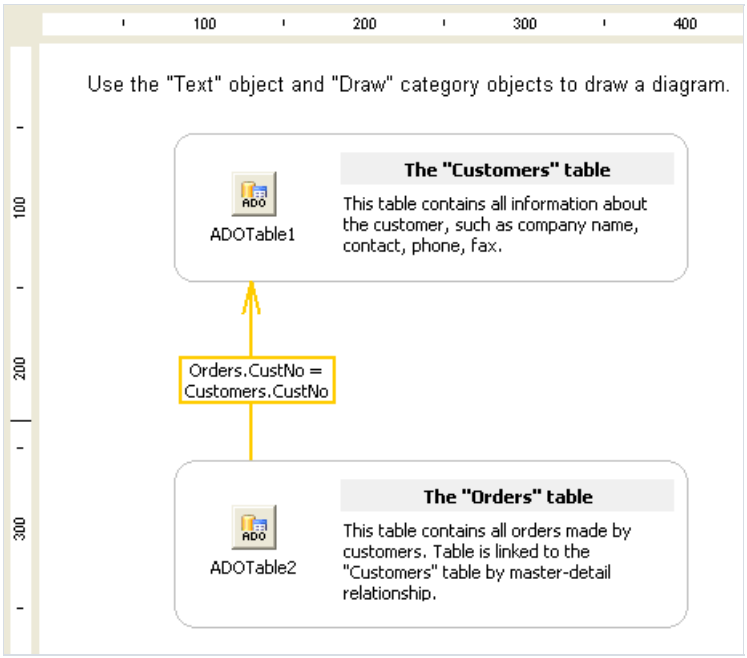


При построении отчета на экран будет выведен диалог, в котором предлагается ввести номер. После ввода нужного значения и нажатия кнопки OK построение отчета будет выполнено. На печать выведутся покупатели с номерами, большими чем введенный.



Другие полезные возможности

На закладке "Данные" можно размещать не только компоненты доступа к данным. С помощью объектов "Текст" и "Рисование" здесь можно размещать поясняющие надписи и рисовать простые диаграммы, как показано на рисунке:



Наследование отчетов

При разработке отчетов мы часто сталкиваемся с ситуацией, когда в каждом отчете встречаются одни и те же данные - зачастую это реквизиты предприятия, логотипы, одно и то же оформление бланков и т.п. Теперь представьте себе, что таких отчетов - не один десяток, и возникает необходимость что-то поправить (например, поменялись реквизиты). Придется открывать каждый отчет и вносить в него исправления.

Для избежания подобных ситуаций можно использовать наследование отчетов. Что это такое?

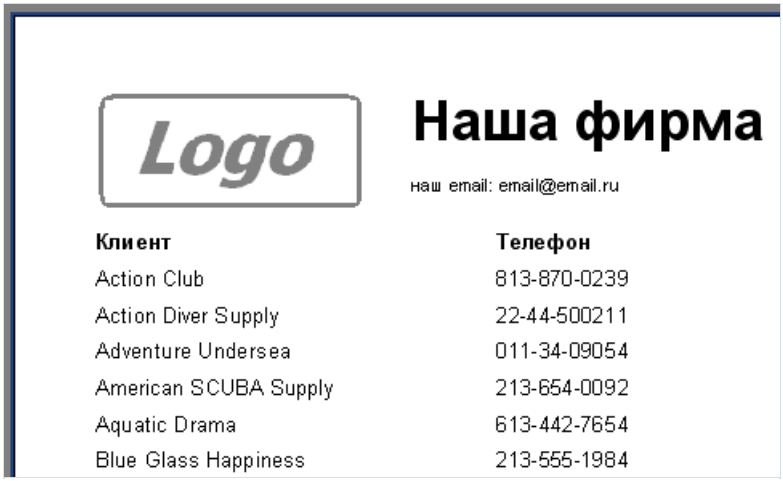
Допустим, у нас есть какие-то общие для всех отчетов элементы. Это может быть шапка отчета с реквизитами и логотипом, подвал страницы. Их оформление стандартно для вашего предприятия и не меняется от отчета к отчету. Такие общие элементы можно вынести в отдельный файл отчета (базовый отчет).

В FastReport есть средства, которые позволяют создать новый отчет на основе базового отчета. При этом новый отчет будет содержать все элементы базового плюс свои собственные.

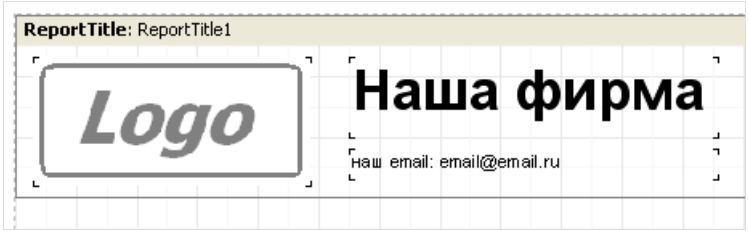
Как такой подход поможет сэкономить время при масштабных изменениях? Очень просто: менять придется только базовый отчет! Все остальные отчеты, которые наследованы от базового, автоматически "подхватят" изменения. Такое поведение обеспечивается механизмом наследования - фактически, при открытии наследованного отчета сначала загружается базовый отчет.

Создание отчета

Создадим простой отчет, который использует наследование. Наш отчет должен будет выглядеть так:

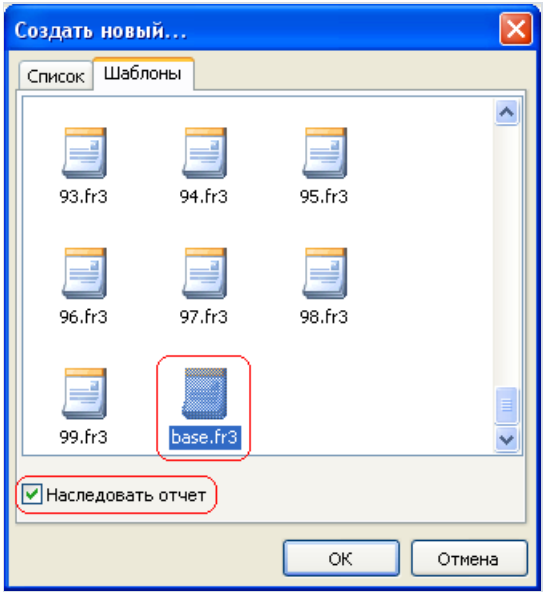


Сначала нам нужно создать базовый отчет. Какую информацию он будет содержать? Очевидно, это будет заголовок с логотипом и реквизитами. Создадим новый отчет и поместим в него объекты:



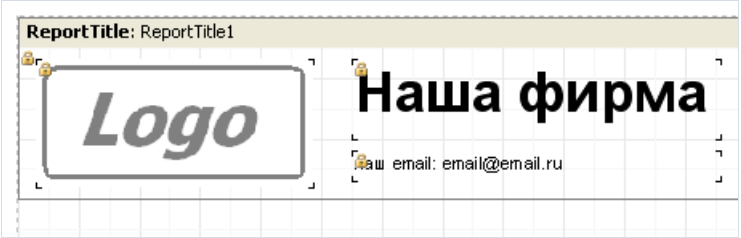
Сохраним наш отчет под именем "base.fr3". В какой папке? Это зависит от того, как настроено ваше приложение. По умолчанию шаблоны хранятся в той же папке, где и исполняемый файл. Задать папку для шаблонов можно в компоненте `TfrxDesigner`, указав свойство `TemplateDir`.

Теперь создадим наследованный отчет. Для этого выберем пункт меню "Файл/Новый..." и в открывшемся окне выберем закладку "Шаблоны". Найдем наш файл base.fr3 в списке, выберем его и включим флажок "Наследовать отчет":



FastReport создаст новый отчет, в котором уже есть объекты базового отчета. Все они помечены значком

замка:

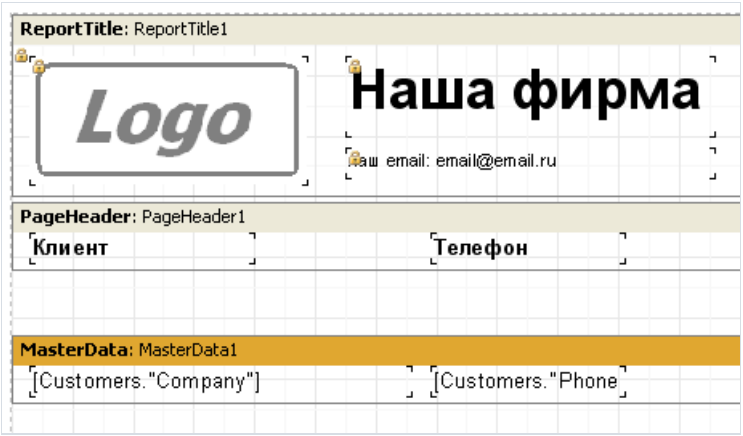


Что это значит? Такие объекты нельзя удалить или переименовать. Также их нельзя переместить на другой бэнд. Все остальное - изменять текст, цвет, размеры - можно.

> Если вы поменяли какое-то свойство (к примеру, цвет) у объекта с замком - оно сохранится в данном экземпляре отчета, и изменения в базовом отчете его уже не коснутся.

Например: в базовом отчете объект был белым, а в наследованном отчете вы поменяли его цвет на красный. Если теперь зайти в базовый отчет и поменять цвет объекта на зеленый, в наследованном отчете он останется красным. Если бы мы не меняли цвет на красный, то изменение цвета в базовом отчете отразилось бы и на наследованном. Это же касается и текста, и размеров/расположения, и любого другого свойства объекта.

Однако вернемся к нашему отчету. Все, что нам осталось сделать - добавить заголовок страницы и бэнд с данными:



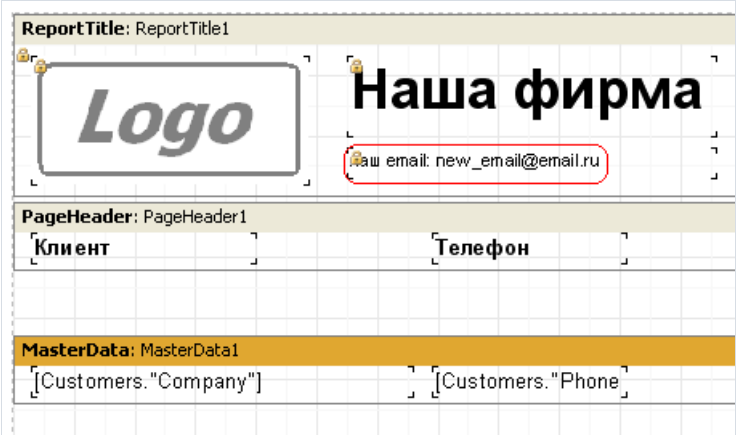
Отчет готов.

Изменение базового отчета

Рассмотрим ситуацию, когда необходимо внести изменения в базовый отчет. Для этого откройте базовый отчет (в нашем примере - base.fr3) и поменяйте нужные поля. Допустим, нужно изменить email:



Сохраним отчет. Теперь откроем наследованный отчет и убедимся, что изменения появились и в нем:



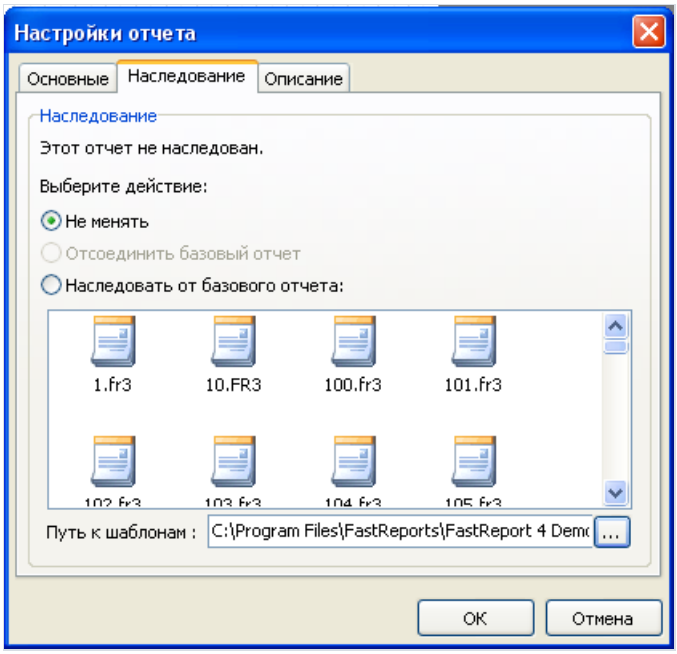
Что, если нам необходимо добавить объекты в базовый отчет? Это легко, но помните о том, что в базовом и наследованном отчете не должно быть двух объектов с одинаковыми именами. Мы в данный момент оперируем с базовым отчетом и не знаем, сколько отчетов уже наследованы от него, и какие в этих отчетах попадаются имена объектов.

> Общее правило: при добавлении объекта в базовый отчет указывайте его имя в виде "имяотчета_имяобъекта". Например, добавьте в базовый отчет объект "Текст" и задайте его имя "BaseMemo3".

На удаление или перемещение объектов таких ограничений нет.

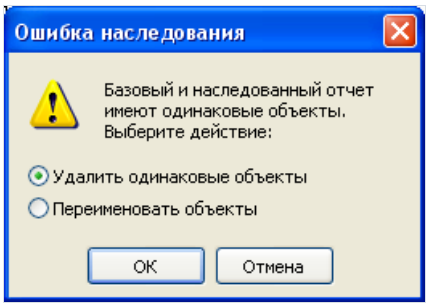
Управление наследованием

Ранее мы рассмотрели создание наследованного отчета с нуля. Что, если у нас уже есть отчет, который надо наследовать от базового отчета? Для этого загрузим наш отчет, который надо наследовать, и зайдем в меню "Отчет/Настройки...". На закладке "Наследование" сосредоточены элементы управления наследованием:



Нам нужно выбрать опцию "Наследовать от базового отчета" и выбрать сам отчет в списке. Также можно установить путь к шаблонам отчета (меняйте его осторожно, т.к. шаблоны, которые использовали другой путь, могут не загрузиться).

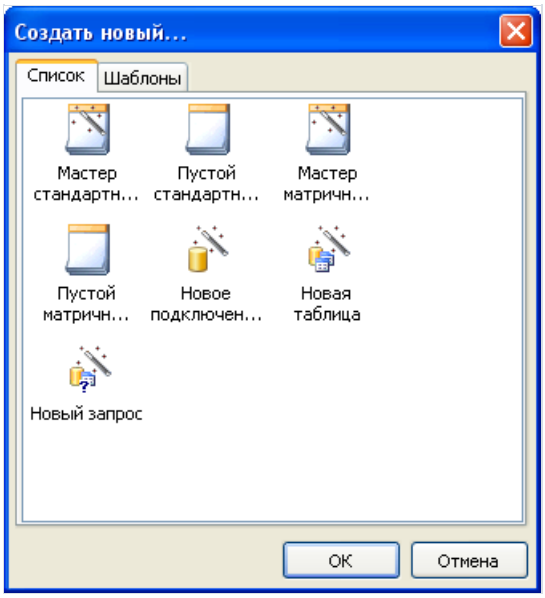
После нажатия на кнопку OK FastReport объединит два отчета. При этом может быть выдано следующее окно с предупреждением:



Это произойдет в случае, если в двух отчетах имеются объекты с одинаковыми именами. Вы можете либо удалить такие объекты из наследованного отчета, либо переименовать их.

Мастера

В комплекте FastReport поставляется несколько мастеров, предназначенных для упрощения построения отчета. Мастера доступны через пункт меню "Файл/Новый...".



Мастер нового отчета

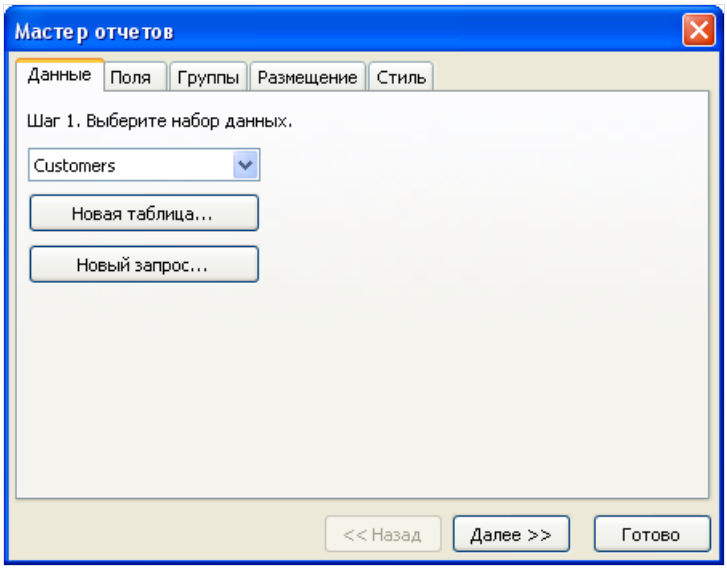
Имеется 4 мастера нового отчета:

- Мастер стандартного отчета
- Мастер матричного отчета
- Пустой стандартный отчет
- Пустой матричный отчет

Мастера "Пустой стандартный отчет" и "Пустой матричный отчет" создают пустой отчет (для обычных или матричных принтеров - о матричных отчетах вы можете почитать в следующей главе), который содержит одну страницу.

Мастера "Мастер стандартного отчета" и "Мастер матричного отчета" позволяют выбрать список полей, которые будут отображены в отчете, группировку и способ размещения полей в отчете. Рассмотрим создание отчета с помощью мастера "Мастер стандартного отчета" подробнее.

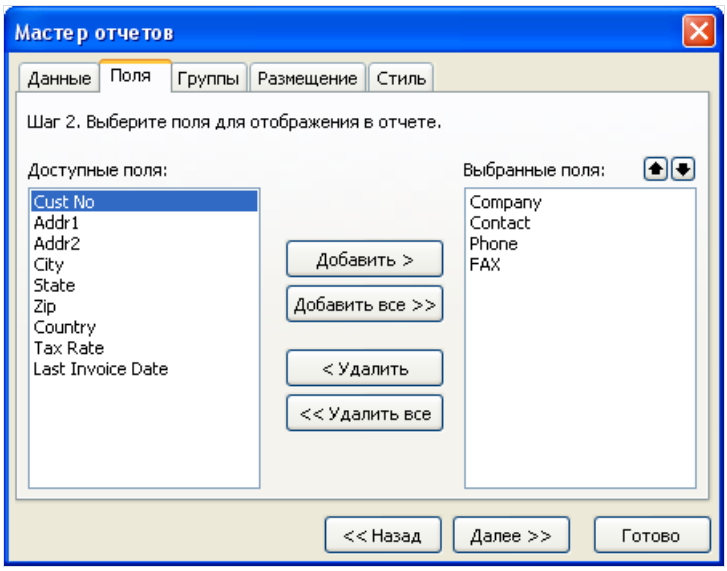
Выберем меню "Файл/Новый...", в открывшемся окне - пункт "Мастер стандартного отчета". Мы увидим окно мастера отчета:



Как видно, окно имеет несколько закладок. На первой закладке нам нужно выбрать источник данных, на основе которого будет построен отчет. Здесь отображены все источники данных, которые имеются в вашем приложении (компоненты `TfrxDBDataSet`).

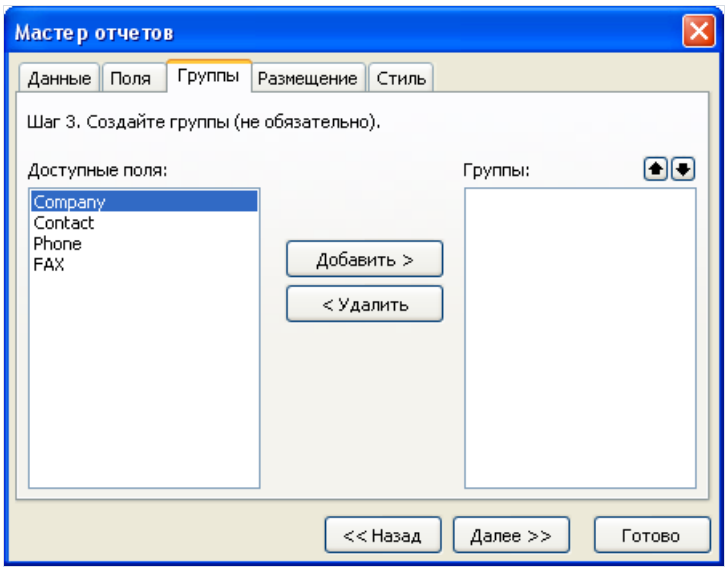
Мы также можем создать новый источник данных - таблицу или запрос, воспользовавшись кнопками "Новая таблица" и "Новый запрос" - при этом будет вызван мастер новой таблицы или нового запроса (см. далее в этой главе). Выберем источник данных Customers и нажмем кнопку "Далее >>".

На следующей закладке необходимо выбрать поля, которые вы хотите показать в отчете:



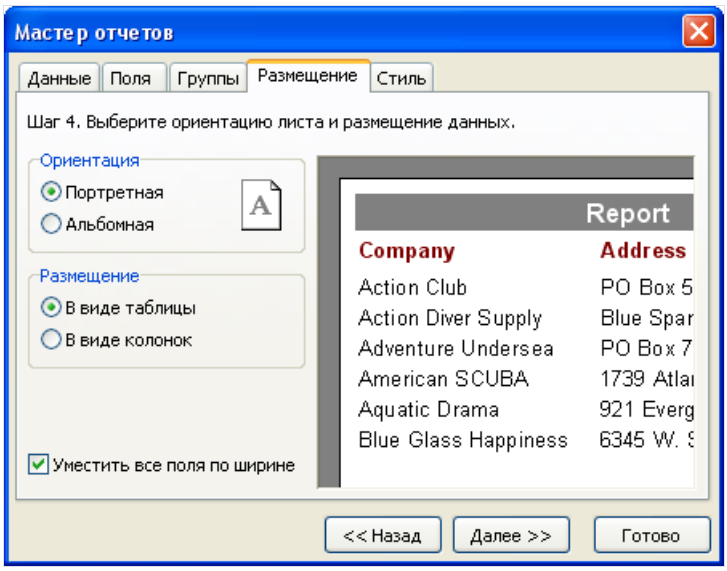
В списке слева отображаются доступные поля, в списке справа - выбранные. Перемещать поля из одного списка в другой можно с помощью кнопок "Добавить", "Добавить все", "Удалить", "Удалить все". С помощью кнопок  поля можно менять местами. Добавим в список выбранных полей поля Company, Contact, Phone, FAX и нажмем кнопку "Далее >>".

На следующей закладке можно добавить в отчет группировку по одному или нескольким выбранным полям. При этом в отчет будут добавлены бэнды Group header, Group footer.



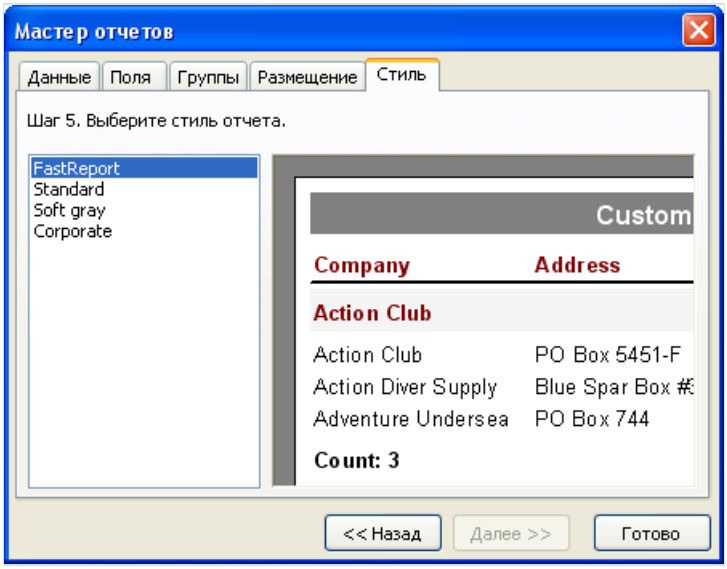
Этот шаг можно пропустить - нажмем кнопку "Далее >>".

На следующей закладке можно выбрать ориентацию страницы отчета и способ размещения полей на ней:

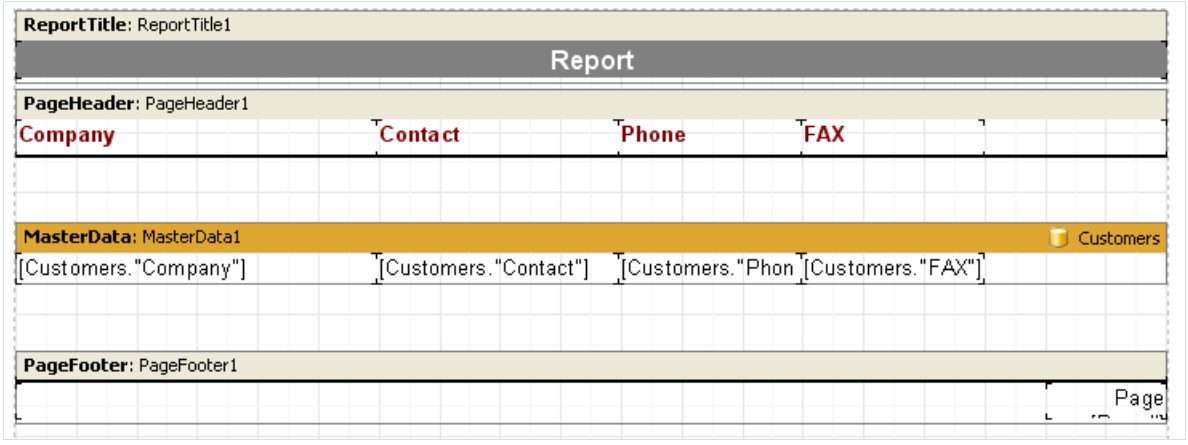


Можно выбрать стиль размещения полей - табличный, когда поля расположены слева направо, или колоночный, когда поля расположены друг под другом. При выборе размещения полей картинка отчета в правой части перерисовывается. Опция "Уместить все поля по ширине" подбирает ширину выбранных полей таким образом, чтобы все поля уместились на странице.

Наконец, на последней закладке мы можем выбрать стиль отчета - цветовую палитру различных элементов отчета.



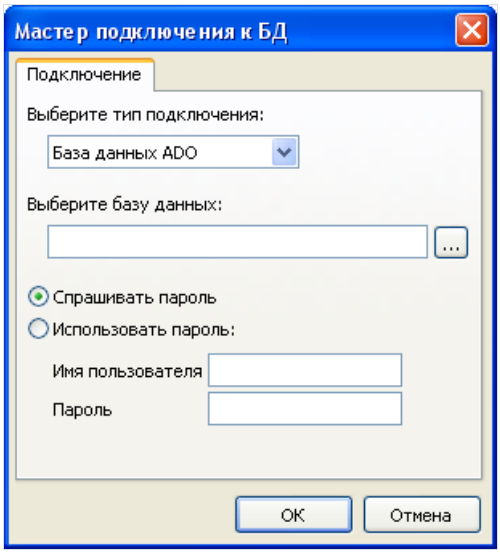
После того, как мы нажмем кнопку "Готово", мастер создаст отчет следующего вида:



Отчет можно сразу же просмотреть в окне предварительного просмотра.

Мастер нового подключения

Этот мастер позволяет добавить в уже существующий отчет новое подключение к БД. Это может быть необходимо, если вы хотите отобразить в отчете данные из двух или более баз данных. Мастер добавляет в отчет компонент типа "База данных ADO".

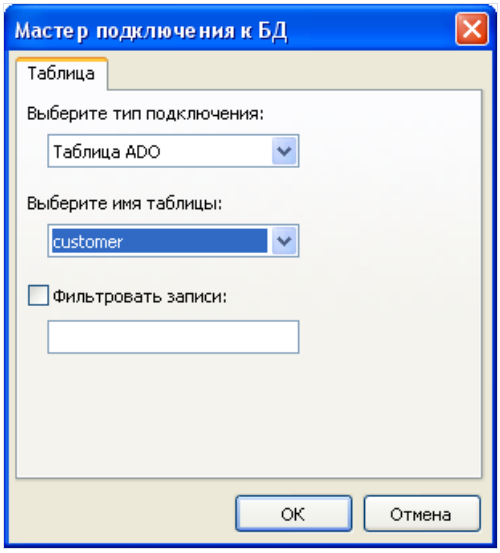


Вам необходимо создать строку подключения (для этого воспользуйтесь кнопкой ) - при этом откроется стандартное окно Windows, где можно выбрать тип подключения и его параметры. После этого задайте имя пользователя и пароль, если необходимо.

Отметим, что создать новое подключение также можно, если переключиться на закладку "Данные" и добавить в отчет компонент "База данных ADO".

Мастер новой таблицы

Этот мастер позволяет добавить в уже существующий отчет новый источник данных - таблицу.



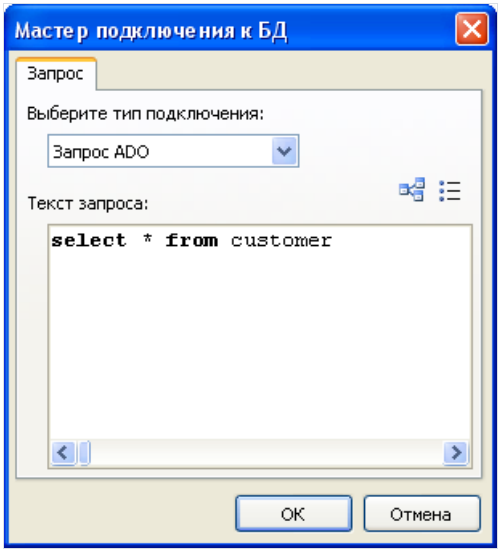
В окне мастера необходимо выбрать имя таблицы. Также вы можете указать условие для фильтрации записей таблицы, например:

```
(CustNo > 2000) and (CustNo < 3000)
```

Отметим, что создать новую таблицу также можно, если переключиться на закладку "Данные" и добавить в отчет компонент "Таблица ADO".

Мастер нового запроса

Этот мастер позволяет добавить в уже существующий отчет новый источник данных - запрос SQL.

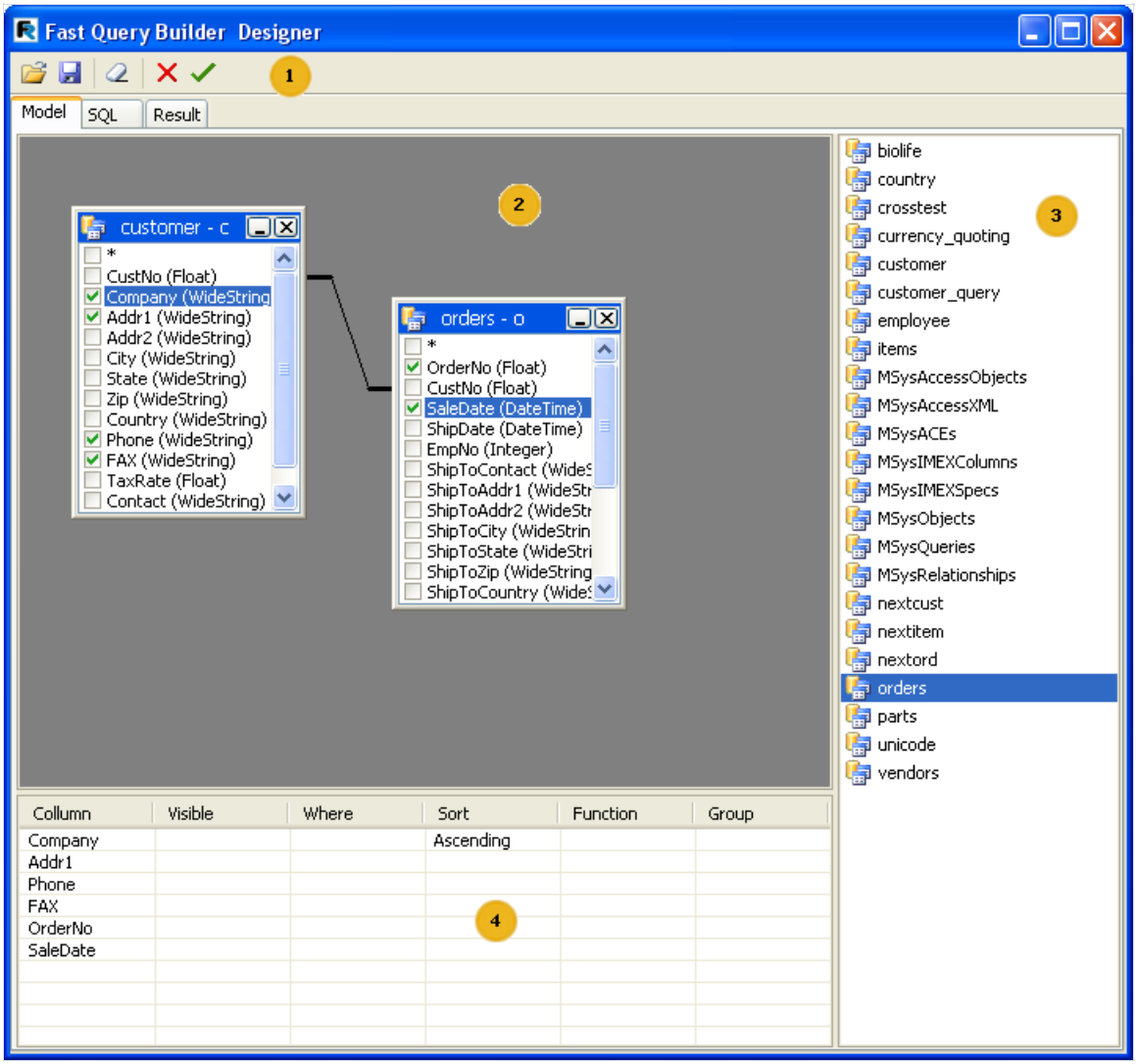


В окне мастера необходимо ввести текст SQL запроса. Вы также можете воспользоваться визуальным конструктором запроса, нажав кнопку . Визуальный конструктор описан далее в этой главе.

Отметим, что создать новый запрос также можно, если переключиться на закладку "Данные" и добавить в отчет компонент "Запрос ADO".

Конструктор запросов

В комплект FastReport (версии Professional, Enterprise) включен визуальный конструктор запросов (для этих целей используется FastQueryBuilder, доступный также как отдельный продукт для использования в ваших приложениях). Конструктор запросов предназначен для визуального построения текста запроса на языке SQL. Внешний вид конструктора следующий:



Цифрами на рисунке обозначены:

1 - панель инструментов

2 - рабочее поле дизайнера

3 - список доступных таблиц

4 - параметры выбранных полей таблиц

Панель инструментов:

- открыть SQL файл

- сохранить запрос в файл (так же в файл сохраняется схема запроса)

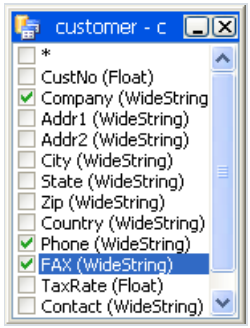
- очистка рабочего пространства дизайнера

- кнопка Ок. Выход из дизайнера с сохранением.

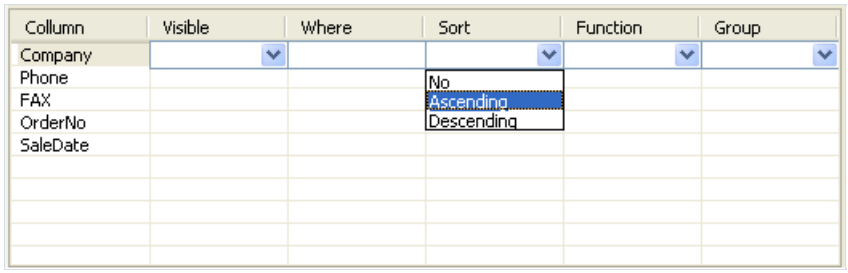
- кнопка Отмена. Выход из дизайнера без сохранения.

Рабочее поле конструктора и список доступных таблиц поддерживают технологию drag&drop, т.е. для размещения таблицы в рабочей области достаточно переместить её туда мышкой. Другой вариант: двойной щелчок на названии таблице в списке доступных таблиц.

Для включения в запрос какого-либо поля из таблицы, достаточно отметить его:



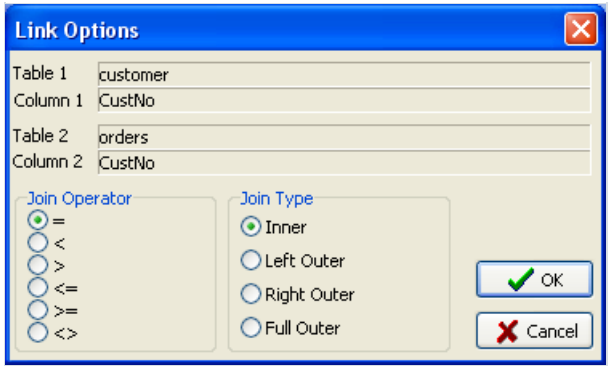
Отмеченные поля появятся в таблице параметров:



- `Visibility` - определяет, попадет ли поле в конструкцию select
- `Where` - условие отбора поля. Например '> 5'
- `Sort` - определяет сортировку по полю.
- `Function` - определяет функцию, применимую к полю
- `Group` - группировка по полю.

Путем "перетаскивания" полей между таблицами возможно образование связей (join). При создании связи проверяется совместимость типов полей. Между несовместимыми полями создать связь нельзя.

Для настройки параметров связи необходимо щелкнуть мышкой на линии связи и выбрать пункт Link options. Появится окно параметров связи:

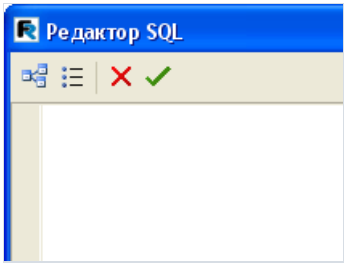


Использование конструктора запросов

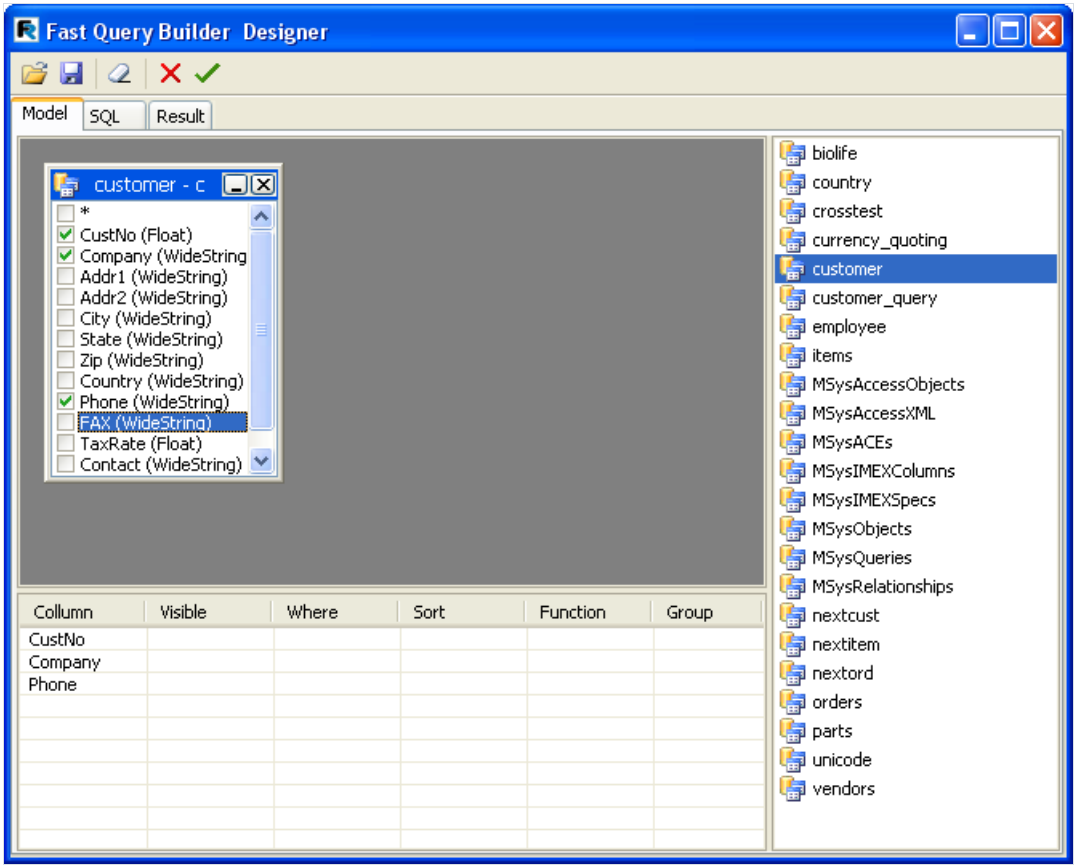
Построим простой отчет, используя конструктор запросов.

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

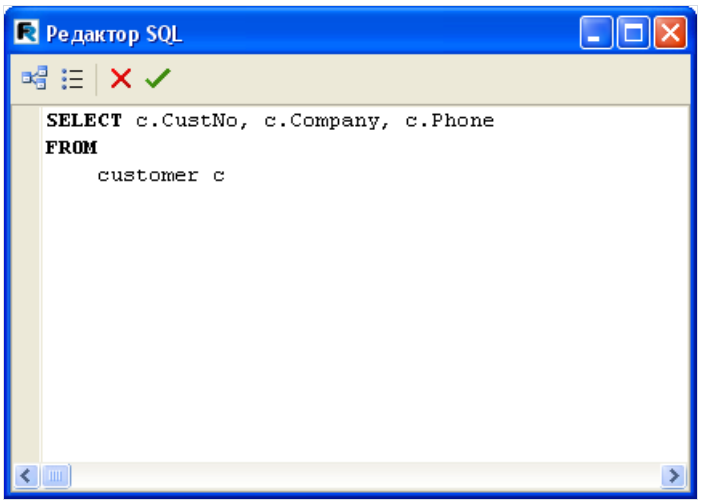
На страницу "Данные" положите компонент "Запрос ADO". Сделайте двойной щелчок на компоненте, и вы увидите окно редактора запроса.



Нажмите кнопку в редакторе, и вы увидите окно конструктора запросов. Выберите таблицу Customer в левой части окна и перенесите ее мышкой на рабочее поле (можно также сделать двойной щелчок для перемещения таблицы). Пометьте галочками поля CustNo, Company, Phone:



Это все, что необходимо для построения запроса. Вы можете посмотреть текст запроса на закладке SQL, а на закладке Result увидеть данные, которые вернул запрос. Нажмите кнопку , чтобы закрыть конструктор. При этом мы вернемся в окно редактора запроса, в котором теперь отображается сгенерированный текст запроса:



> Если вы исправите текст запроса, то потеряете схему (размещение таблиц в конструкторе запроса и связи между ними). Если текст запроса не изменять вручную, вы всегда можете зайти в конструктор запроса и поправить схему визуально.

Нажав кнопку OK в редакторе, мы вернемся в дизайнер отчета. Нам осталось подключить бэнд "Данные 1 уровня" к источнику данных и разместить поля на бэнде.

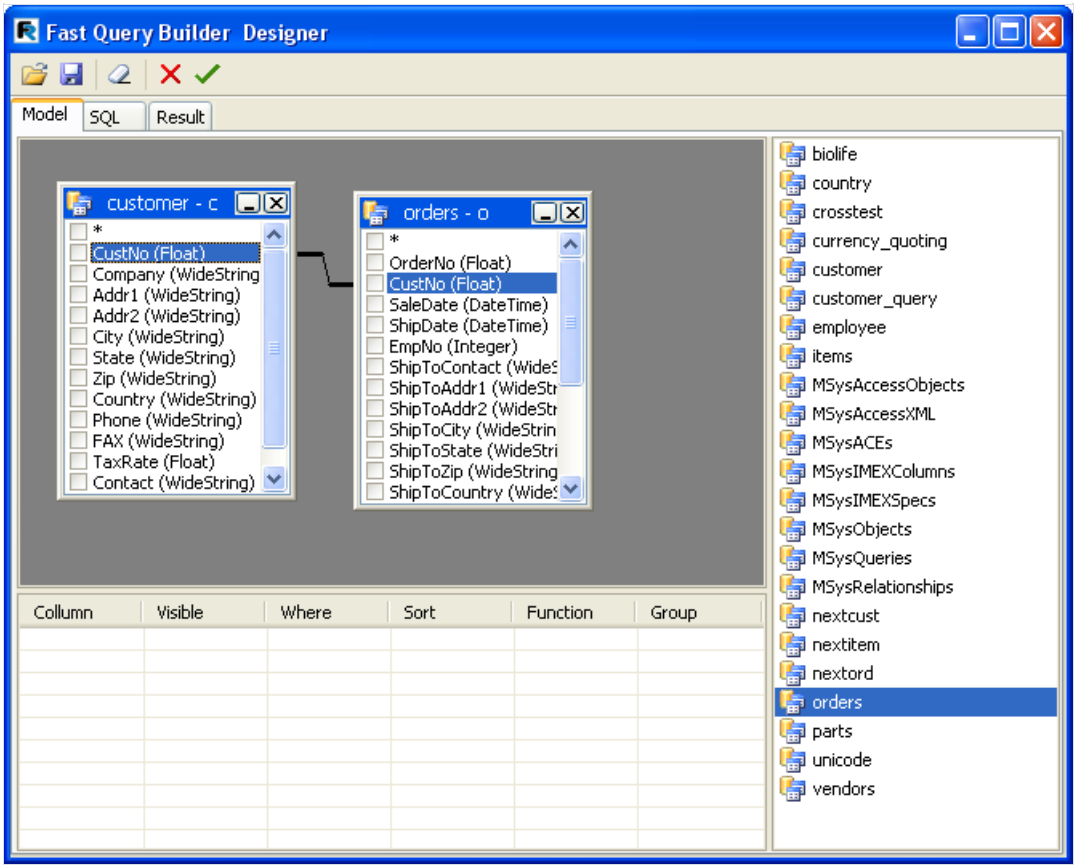
Построение сложного запроса

В предыдущем примере мы строили отчет на основе данных из одной таблицы. Рассмотрим построение запроса, который включает в себя данные из двух таблиц.

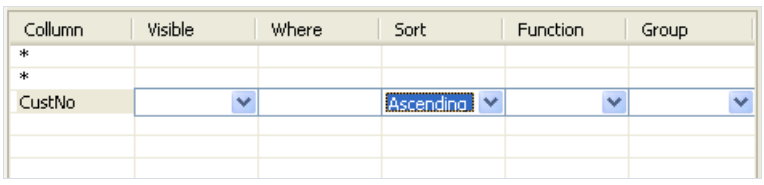
Ранее мы рассматривали работу отчета с группами (глава "Отчет с группами"). Построим запрос для этого отчета с помощью конструктора запросов. Нам необходимо составить запрос на языке SQL, который вернет данные из обеих таблиц, сгруппированные по определенному условию. В нашем случае условие – соответствие полей CustNo в обеих таблицах.

Как и в предыдущем примере, создаем новый отчет и кладем на страницу компонент "Запрос ADO". В редакторе запроса нажимаем кнопку для запуска конструктора запроса.

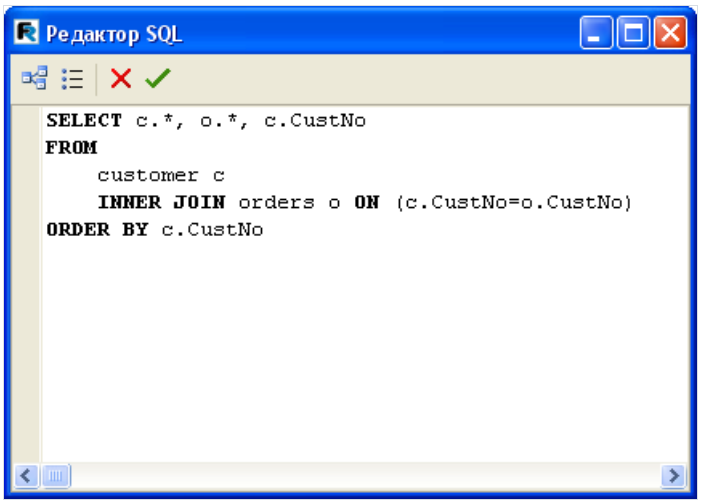
Перетаскиваем на рабочее поле две таблицы – Customers и Orders. Обе таблицы имеют поле CustNo, по которому мы должны их связать. Путем перетаскивания поля CustNo из одной таблицы в другую мы создаем связь между таблицами:



Теперь необходимо отметить поля, которые должен включать в себя запрос, и сгруппировать его по полю CustNo. Для этого отметьте галочками поля "*" в обеих таблицах, а также поле CustNo в таблице Customer. В нижней части окна появятся выбранные нами поля, после чего надо выбрать сортировку для поля CustNo:

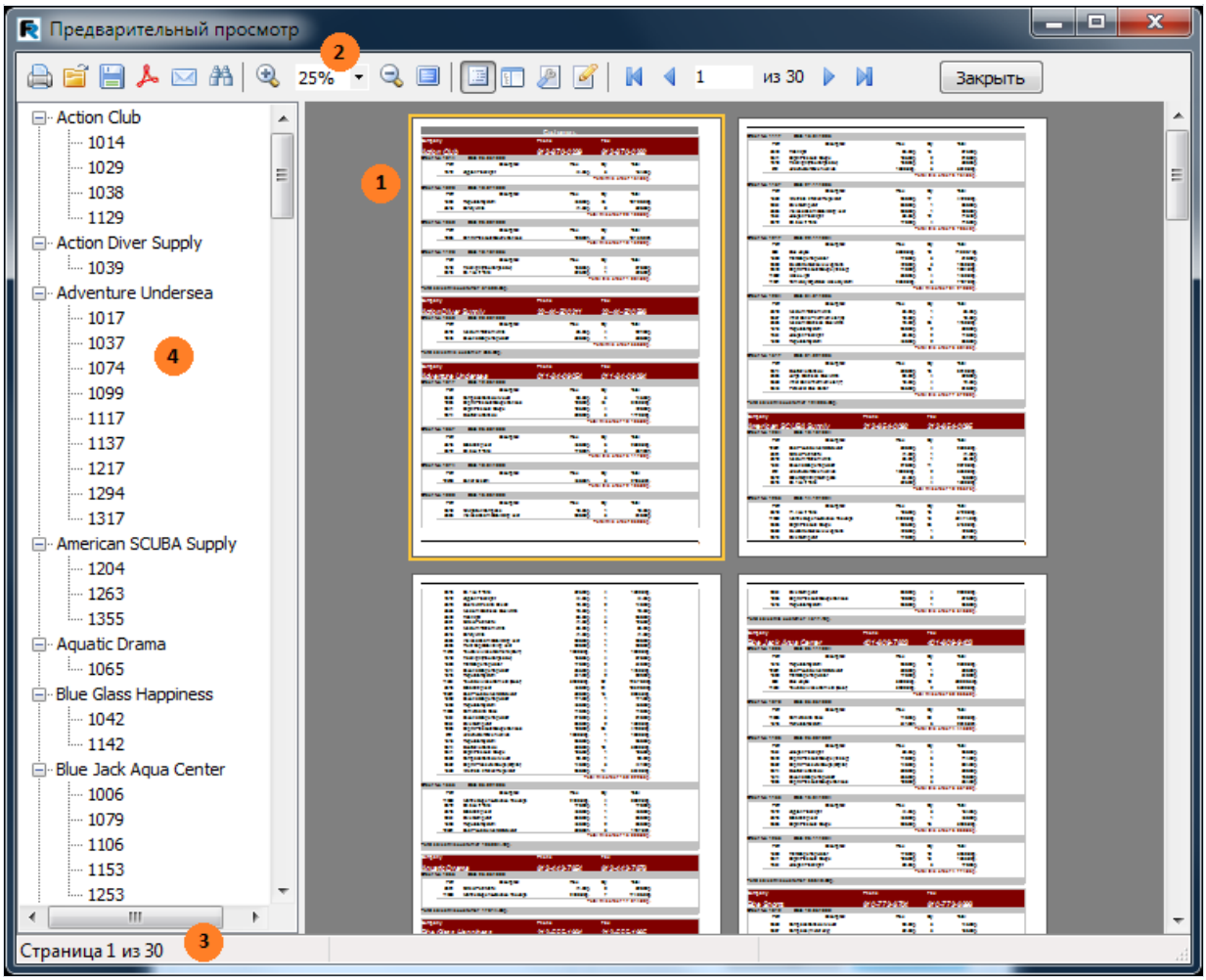


Все, запрос готов. Его текст выглядит так:


Редактор SQL
SELECT c.*, o.*, c.CustNo
FROM
customer c
INNER JOIN orders o ON (c.CustNo=o.CustNo)
ORDER BY c.CustNo

Просмотр, печать, экспорт отчета

Построенный отчет можно отобразить на экране, распечатать на принтере или экспортировать в один из поддерживаемых форматов. Все это можно сделать в окне предварительного просмотра.



Цифрами на рисунке обозначены:

1 – страницы готового отчета;

2 – панель инструментов;

3 – строка состояния;

4 – боковая панель. Здесь может отображаться либо дерево отчета (как на рисунке), либо эскизы страниц.

На панели инструментов имеются следующие кнопки:



Иконка	Название	Описание
	Печать отчета	Печатает отчет. Клавиатурный аналог – Ctrl+P.
	Открыть отчет	Открывает файл с готовым отчетом (*.fp3).

Иконка	Название	Описание
	Сохранить отчет	Сохраняет отчет в файл (*.fp3), или экспортирует его в один из поддерживаемых форматов.
	Экспорт в PDF	Экспортирует отчет в файл Adobe Acrobat (*.pdf). Эта кнопка отображается, если установлен соответствующий фильтр экспорта.
	Отправить по почте	Экспортирует отчет в один из поддерживаемых форматов и отправляет его по электронной почте как вложение. Эта кнопка отображается, если установлен соответствующий фильтр экспорта.
	Поиск текста	Поиск текста в отчете. Клавиатурный аналог – Ctrl+F.
	Увеличить	Увеличивает масштаб.
	Масштаб	Выбор произвольного масштаба.
	Уменьшить	Уменьшает масштаб.
	На весь экран	Отображает отчет на весь экран. Для возвращения к нормальному режиму сделайте двойной щелчок мышью на отчете.
	Дерево отчета	Показывает или скрывает дерево отчета.
	Эскизы	Показывает или скрывает эскизы страниц
	Свойства страницы	Вызывает диалог со свойствами страницы.
	Редактировать страницу	Редактирует текущую страницу.
	В начало	Переход на первую страницу отчета.
	Предыдущая страница	Переход на предыдущую страницу отчета.
	Номер страницы	Переход на страницу отчета с указанным номером. Введите номер и нажмите Enter.
	Следующая страница	Переход на следующую страницу отчета.
	В конец	Переход на последнюю страницу отчета.
Закрыть	Закрыть окно	Закрыть окно просмотра.

Клавиши управления

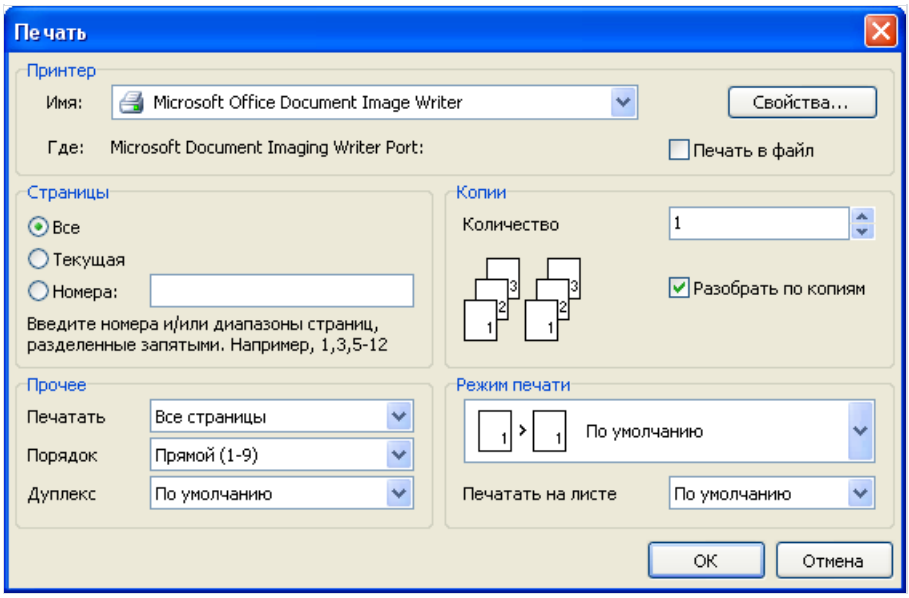
Клавиши	Описание
Ctrl+S	Сохранить отчет в файл *.fp3.
Ctrl+P	Печатать отчет.
Ctrl+F	Поиск текста.
F3	Продолжение поиска.
Стрелки	Плавный скроллинг документа.
PageUp, PageDown	Прокрутка вверх/вниз.
Ctrl+PageUp, PageDown	Прокрутка на следующую/предыдущую страницы.
Home	В начало документа.
End	В конец документа.

Управление мышью

Действие	Описание
Левая кнопка	Щелчок на выбранном объекте (в интерактивном отчете); скроллинг отчета в режиме «рука» (при нажатой кнопке двигайте мышью); в режиме «лупа» - увеличение масштаба.
Правая кнопка	Контекстное меню; в режиме «лупа» - уменьшение масштаба.
Двойной щелчок	В режиме отображения на весь экран – возврат к нормальному режиму.
Колесо мыши	Прокрутка листа отчета.

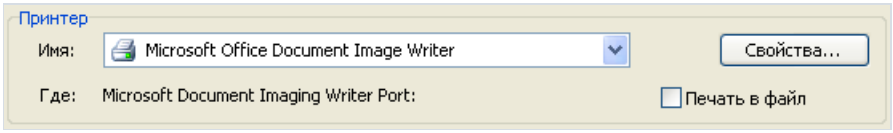
Печать отчета

Для того, чтобы распечатать отчет на принтере, нажмите кнопку (или комбинацию клавиш Ctrl+P). Появится окно – диалог печати:

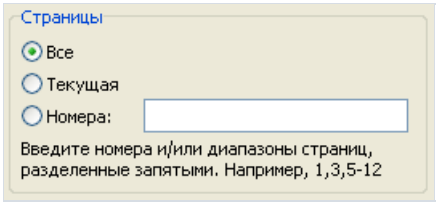


Рассмотрим настройки, доступные в этом диалоге.

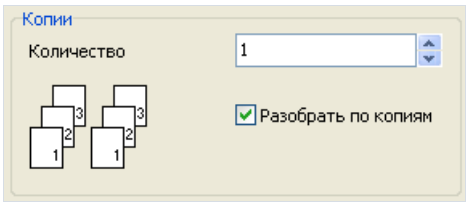
Группа "Принтер": здесь можно выбрать принтер, задать его настройки (кнопка "Свойства") и выбрать печать в файл.



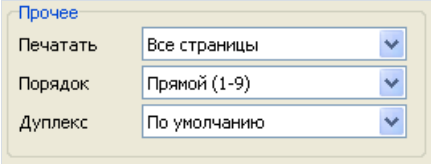
Группа "Страницы": здесь можно выбрать, какие страницы печатать (все, текущую или заданные номера страниц).



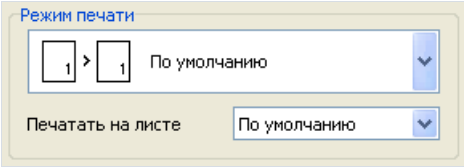
Группа "Копии": здесь можно задать количество копий и выбрать порядок страниц в копиях ("Разобрать по копиям"):

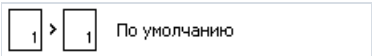


Группа "Прочее": здесь можно выбрать, какие страницы из выбранного диапазона печатать (все, четные, нечетные), выбрать порядок печати (прямой, обратный) и задать настройки для двусторонней печати ("Дуплекс" - если ваш принтер его поддерживает).

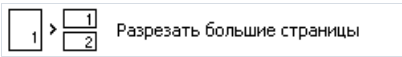


Наконец, группа "Режим печати" позволяет выбрать один из режимов печати.





- Печать по умолчанию. Принтер печатает на формате бумаги, указанной в отчете. Одной странице отчета соответствует один лист распечатки.



- Разрезать большие страницы. Этот режим используется, если необходимо распечатать отчет формата А3 на бумаге А4. При этом из одной страницы отчета получается два печатных листа. При выборе этого режима надо выбрать формат бумаги, на котором вы хотите печатать, из списка "Печатать на листе".



- Объединять маленькие страницы. Этот режим используется, если необходимо напечатать отчет А4 на формате А3. На одном печатном листе печатается две страницы отчета. При выборе этого режима надо выбрать формат бумаги, на котором вы хотите печатать, из списка "Печатать на листе".

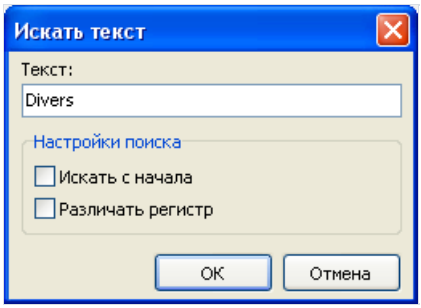


- Масштабировать. Выполняется печать отчета на заданном формате. При этом изображение масштабируется (уменьшается или увеличивается) в зависимости от соотношения формата отчета и формата листа. Одной странице отчета соответствует один лист распечатки. При выборе этого режима надо выбрать формат бумаги, на котором вы хотите печатать, из списка "Печатать на листе".

После нажатия на кнопку ОК начинается печать отчета. Если выбран флажок «Печать в файл», то будет запрошено имя файла и отчет будет сохранен в этот файл (файл с расширением *.prn, содержит копию информации, отправляемой на принтер).

Поиск текста в отчете

FastReport позволяет искать заданную строку текста в окне предварительного просмотра. Для этого служит кнопка на панели инструментов (или ее клавиатурный аналог Ctrl+F). При этом на экране появляется диалог поиска:



Здесь можно задать строку поиска, а также опции:

- Искать с начала – искать текст с начала документа. Иначе поиск будет выполняться с текущей страницы;
- Различать регистр – различать регистр букв (строчные и прописные) при поиске.

При нажатии кнопки OK будет произведен поиск текста и высвечен первый найденный элемент:



Чтобы продолжить поиск, нажмите клавишу F3. Будет подсвечен следующий элемент.

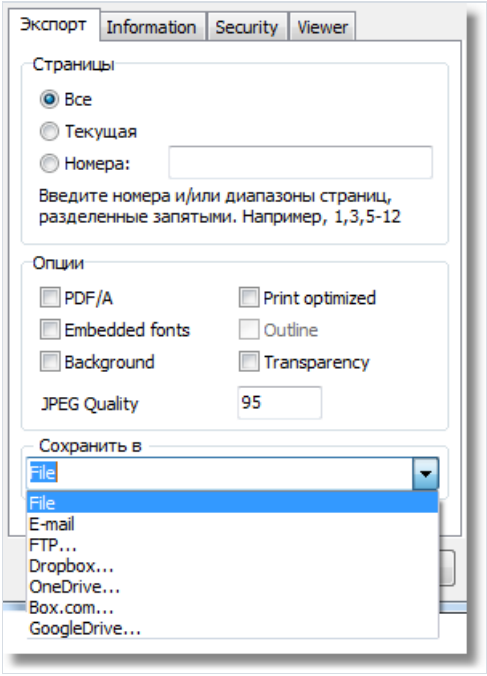
Сохранение отчетов

В FastReport используется система транспортов, которая позволяет сохранить готовый отчет или результат экспорта в одно из следующих мест:

- файл на диске;
- отправка по электронной почте;
- FTP;
- Dropbox;
- OneDrive;
- Google Drive;
- Box.com.

Чтобы сохранить готовый отчет (файл .fp3), в окне просмотра нажмите кнопку "Сохранить" и выберите пункт меню "Готовый отчет".

Для того чтобы сохранить результат экспорта, выберите нужный тип экспорта (например, "Документ PDF") и в окне настроек откройте список "Сохранить в":

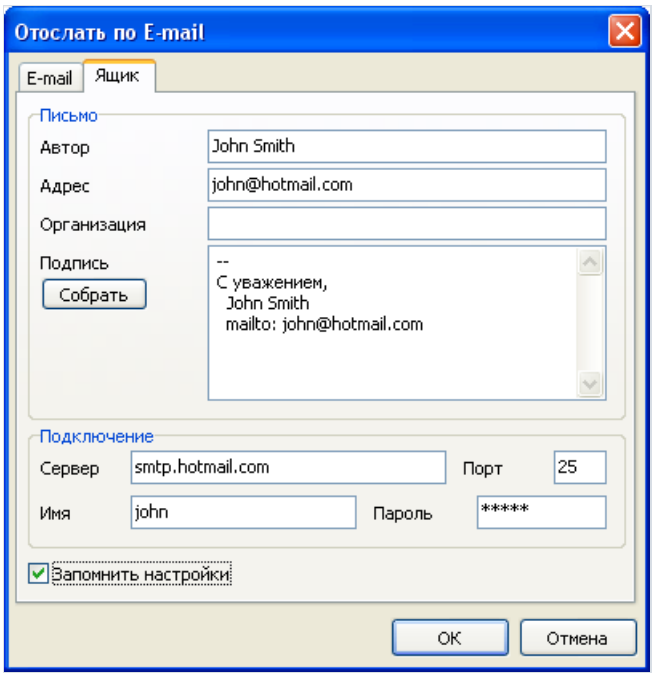


Отправка отчета по электронной почте

FastReport позволяет отправить готовый отчет по электронной почте в нужном вам формате. Для отправки письма не используются никакие вспомогательные программы.

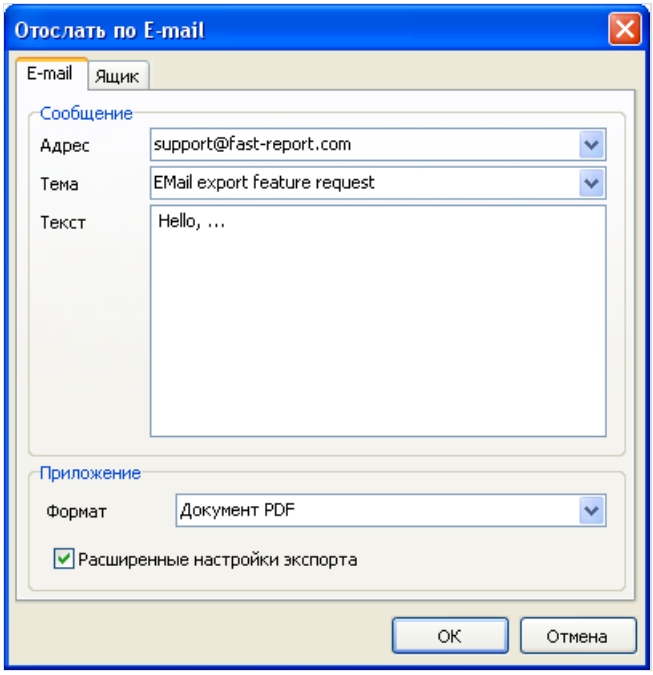
При выборе отправки по e-mail будет предложено диалоговое окно для настройки параметров сообщения и выбора формата.

Перед началом работы необходимо настроить параметры владельца почтового ящика. Все эти параметры находятся на закладке "Ящик":



- Автор – имя отправителя;
- Адрес – электронный адрес отправителя;
- Организация – организация отправителя;
- Подпись – подпись для письма, может быть автоматически сформирована при нажатии на кнопку "Собрать" при условии, если заполнены ранее рассмотренные поля;
- Сервер – адрес SMTP сервера;
- Порт – порт SMTP сервера;
- Имя – имя доступа для авторизации на SMTP сервере, если ее использование необходимо для отправки письма через заданный SMTP сервер;
- Пароль – пароль для авторизации;
- Запомнить настройки – запомнить все параметры для дальнейшего использования.

После заполнения параметров, необходимых для отправки письма (это нужно сделать только один раз), необходимо заполнить параметры самого письма на закладке «E-mail»:

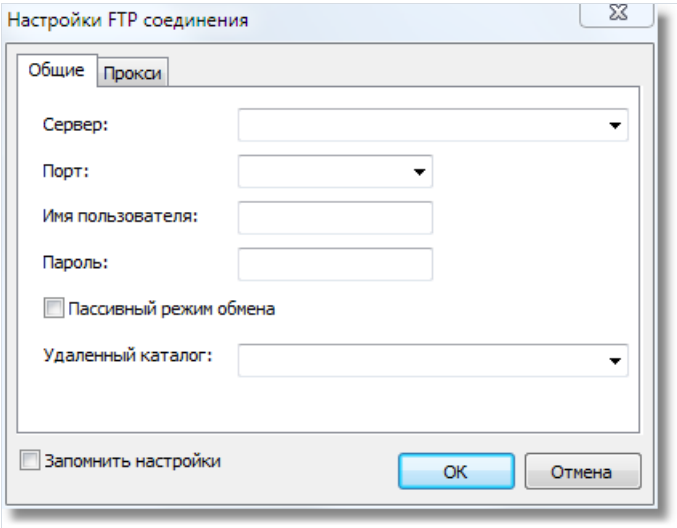


- Адрес – электронный адрес получателя письма, ранее использовавшиеся адреса можно выбрать из выпадающего меню;
- Тема – тема сообщения, ранее использовавшиеся темы можно выбрать из выпадающего меню;
- Текст – Текст сообщения;
- Формат – формат прилагаемого к письму отчета, может быть выбран один из ранее рассмотренных форматов, а также собственный формат подготовленного отчета FastReport (.FP3);
- Расширенные настройки экспорта – при включении этой опции, после нажатия на кнопку «OK» будет выдано диалоговое окно с настройками выбранного формата экспорта, иначе будут использованы параметры экспорта по умолчанию.

Особенности экспорта по e-mail: поддерживается только plain аутентификация на SMTP серверах, если аутентификация не требуется – заполнять поля «Имя» и «Пароль» в настройках не нужно.

Сохранение на FTP

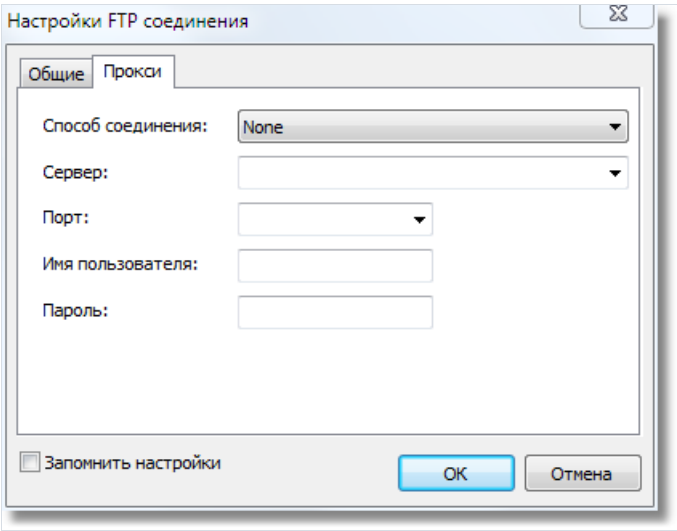
При сохранении готового отчета или результата экспорта на FTP сервер появится следующее окно:



На вкладке "Общие" есть следующие поля:

- Сервер. В это поле нужно ввести URL-адрес FTP сервера, на который нужно сохранить файл.
- Порт. Введите номер порта.
- Имя пользователя и Пароль. Нужно ввести ваши логин и пароль на данном FTP-сервере.
- Пассивный режим обмена. Выберите режим обмена данными с FTP-сервером.
- Удаленный каталог. Введите название папки на сервере (не обязательно).

Если вы используете прокси-сервер, то на вкладке прокси вы можете ввести URL-адрес прокси сервера, порт, логин и пароль:



Введенные настройки можно сохранить для дальнейшего использования, выбрав флажок "Запомнить настройки". После нажатия кнопки OK файл будет сохранен на сервере FTP.

Сохранение в Dropbox

Перед тем как воспользоваться этой возможностью, необходимо создать приложение в Dropbox аккаунте. Для этого нужно войти в свой аккаунт на Dropbox и выполнить следующие шаги:

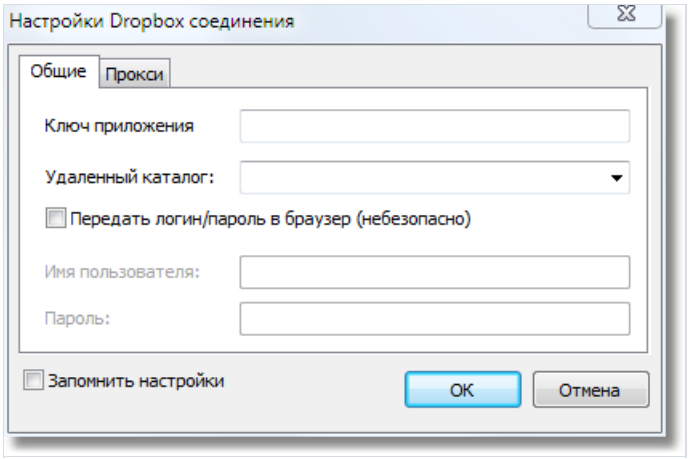
- Щелкнуть кнопку "Еще (More)". Она находится внизу страницы Dropbox.
- В выпадающем списке выбрать "Разработчикам (Developers)". Вы попадете на страницу для разработчиков.
- Перейти по ссылке "App Console". В результате вы попадете к списку приложений.
- Щелкнуть кнопку "Create App". Dropbox захочет проверить ваш E-mail. Нажмите кнопку "Send Email".

На вашу почту будет отправлено письмо, в котором нужно нажать "Подтвердить адрес электронной почты".

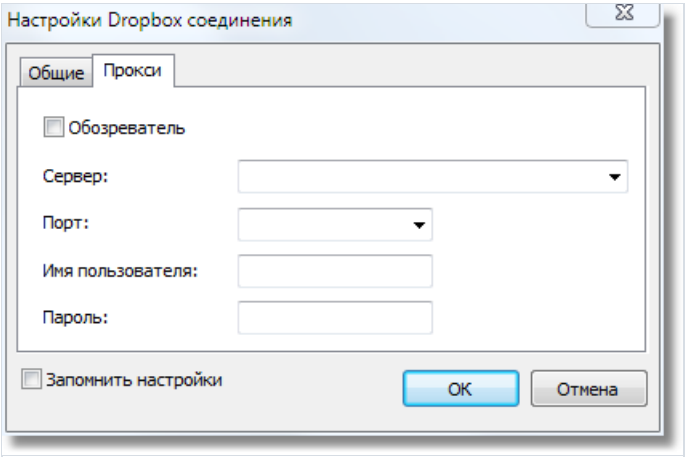
В итоге вы попадете на страницу "Create a new Dropbox Platform app". Здесь нужно выбрать "Dropbox API app" и на вопрос "What type of data does your app need to store on Dropbox?" выбрать ответ "Files and datastores". А на вопрос "Can your app be limited to its own, private folder?" можно выбрать любой из двух предложенных ответов. Последним на этой странице нужно ввести имя приложения (оно может быть любым). После нажатия на кнопку "Create app" система проверит, не занято ли уже введенное вами имя приложения, и создаст приложение.

В итоге мы попадем на страницу настройки приложения. Здесь нам интересны "App key" и "App secret", они понадобятся при экспорте в Dropbox.

При сохранении готового отчета или результата экспорта в Dropbox появится следующее окно:



Если вы используете прокси-сервер, то на вкладке "Прокси" вы можете ввести URL-адрес прокси сервера, порт, логин и пароль:

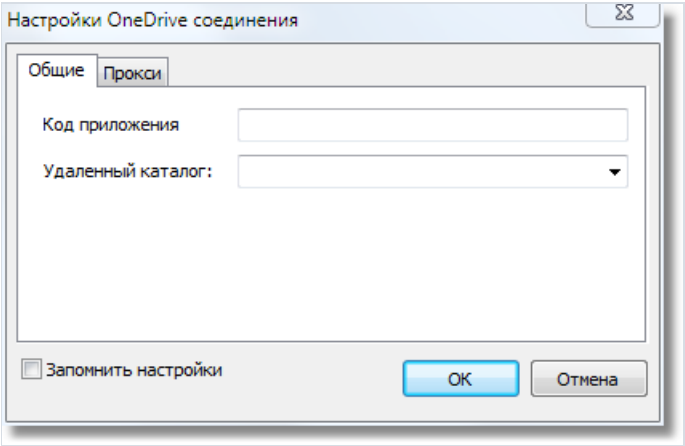


Введенные настройки можно сохранить для дальнейшего использования, выбрав флажок "Запомнить настройки". После нажатия кнопки OK будет открыто окно браузера, где нужно ввести свой логин и пароль для авторизации в Dropbox. После этого файл будет сохранен в Dropbox.

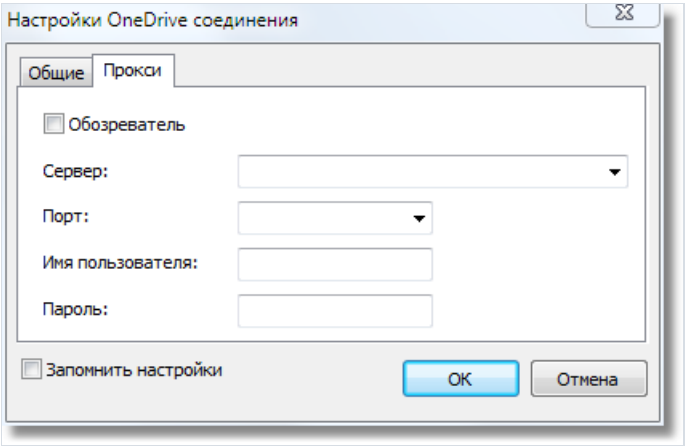
Сохранение в OneDrive

Перед тем как воспользоваться этой возможностью, необходимо создать приложение в OneDrive. Для этого на странице OneDrive переходим по ссылке "Разработчикам". На странице центра разработки жмем "Мои приложения". Далее жмем "Создать приложение". Вводим имя приложения и выбираем язык. Читаем "условия использования" и "заявление о конфиденциальности" и жмем кнопку "Я принимаю". В итоге попадаем на страницу настроек приложения. Здесь мы видим "ИД клиента" и "Секрет клиента", которые нам и нужны. Остается только указать домен перенаправления и нажать кнопку "Сохранить".

При сохранении готового отчета или результата экспорта в OneDrive появится следующее окно:



Если вы используете прокси-сервер, то на вкладке "Прокси" вы можете ввести URL-адрес прокси сервера, порт, логин и пароль:



Введенные настройки можно сохранить для дальнейшего использования, выбрав флажок "Запомнить настройки". После нажатия кнопки OK будет открыто окно браузера, где нужно ввести свой логин и пароль для авторизации в OneDrive. После этого файл будет сохранен в OneDrive.

Сохранение в Google Drive

Перед тем как воспользоваться этой возможностью, необходимо создать приложение в GoogleDrive. Для этого нужно зайти по адресу https://code.google.com/apis/console

На этой странице принимаем условия лицензионного соглашения.

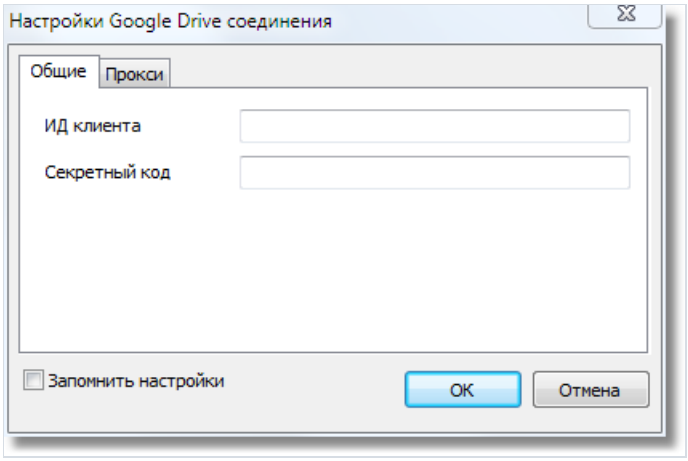
После этого мы попадаем на страницу управления проектами. Здесь переходим на вкладку "Services" и активируем Drive API. Переходим на вкладку "API Access" и жмем "Create an OAuth 2.0 client ID". В секции "Branding Information" вводим имя приложения и жмем Next. В секции "Client ID Settings" выбираем следующее:

- "Installed application" для Application type.
- "Other" для Installed application type.

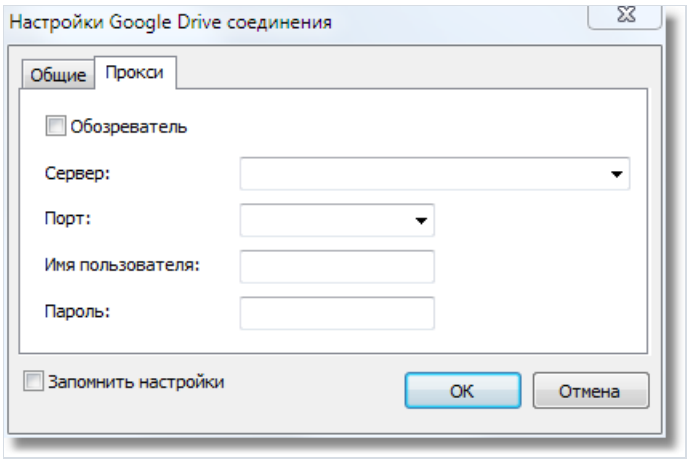
Жмем "Create Client ID".

В результате на странице "API Access page" появится секция "Client ID for installed applications", в которой мы увидим Client ID и Client secret. Оба этих значения нам понадобятся в дальнейшем.

При сохранении готового отчета или результата экспорта в Google Drive появится следующее окно:



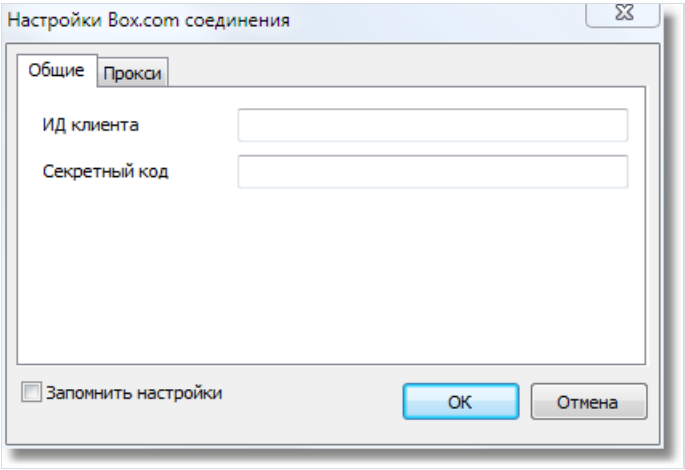
Если вы используете прокси-сервер, то на вкладке "Прокси" вы можете ввести URL-адрес прокси сервера, порт, логин и пароль:



Введенные настройки можно сохранить для дальнейшего использования, выбрав флажок "Запомнить настройки". После нажатия кнопки OK файл будет сохранен в Google Drive.

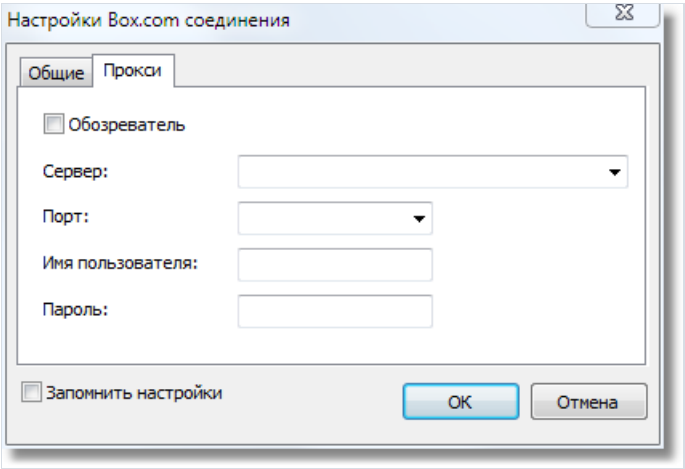
Сохранение в Box.com

При сохранении готового отчета или результата экспорта в Box.com появится следующее окно:



Здесь нужно ввести полученные ранее ИД клиента (Client ID) и Секретный код (Client secret).

Если вы используете прокси-сервер, то на вкладке "Прокси" вы можете ввести URL-адрес прокси сервера, порт, логин и пароль:

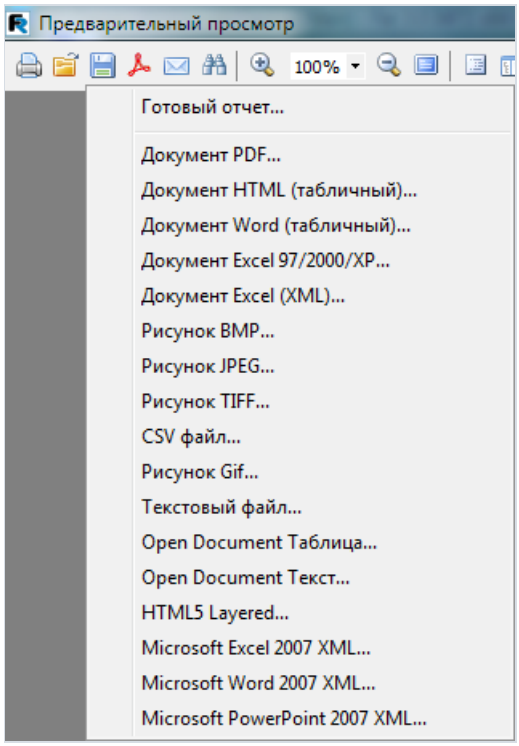


Введенные настройки можно сохранить для дальнейшего использования, выбрав флажок "Запомнить настройки". После нажатия кнопки OK файл будет сохранен в Box.com.

Экспорт отчетов

FastReport позволяет осуществлять экспорт построенного отчета в различные форматы для последующего редактирования, архивирования, пересылки по электронной почте и др.

На данный момент поддерживается экспорт в следующие форматы: PDF, HTML, Excel, Excel 2007, Excel XML, RTF, Word 2007, PowerPoint 2007, TXT, ODS, ODT, CSV, BMP, JPG, TIFF, GIF. Для выбора экспорта нажмите кнопку "Сохранить" на панели инструментов:



Экспорты в FastReport используют один из 3 методов:

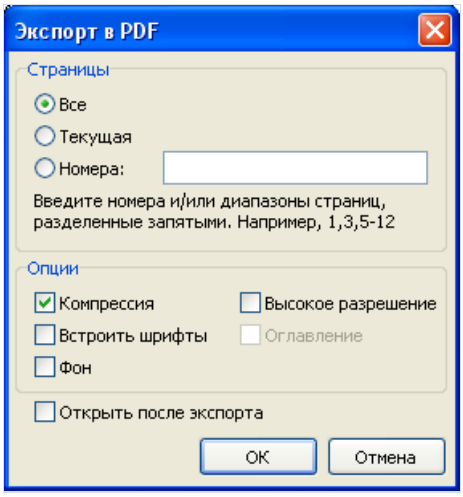
- `послойный` – осуществляется поочередный перенос объектов отчета в результирующий файл, соответствие экспорта приближено к оригиналу;
- `табличный` – при переносе объектов используется промежуточная матрица размещения объектов, высокое соответствие оригиналу при условии соблюдения правил создания корректного шаблона отчета (раздел Рекомендации по разработке отчетов);
- `отрисовка` – осуществляется отрисовка всех объектов отчета на изображении страницы, полное соответствие оригиналу, применяется при экспорте в графические форматы.

Экспорт в формат PDF

PDF (Portable Document Format) – платформо-независимый формат электронных документов, созданный фирмой Adobe Systems. Для просмотра используется бесплатный пакет Acrobat Reader. Данный формат достаточно гибкий - позволяет внедрять необходимые шрифты, векторные и растровые изображения, очень хорошо подходит для передачи и хранения документов, предназначенных для просмотра и последующей печати.

Метод экспорта – послойный.

При экспорте в формат PDF будет предложено диалоговое окно для настройки параметров выходного файла.



Параметры экспорта:

- Компрессия – сжатие выходного файла, уменьшает размер файла, но увеличивает время экспорта;
- Встроить шрифты – все шрифты, использованные в отчете, будут также помещены в выходной файл PDF для корректного отображения файла на компьютере, где этих шрифтов может не быть, размер выходного файла значительно увеличивается;
- Фон – экспорт графического изображения, присвоенного странице в файл PDF, значительно увеличивает размер выходного файла;
- Высокое разрешение – вывод графических изображений в высоком разрешении для последующего корректного отображения при печати на принтере результирующего файла, включение этой опции нужно только в том случае, если документ содержит графику и будет необходима его печать, значительно увеличивает размер выходного файла;
- Оглавление – опция активна, когда в отчете используется дерево отчета, включает возможность экспорта дерева в результирующий документ;
- Открыть после экспорта – результирующий файл будет открыт сразу же после экспорта программой просмотра PDF файлов, назначенной в операционной системе по умолчанию (к примеру, Adobe Acrobat Reader).

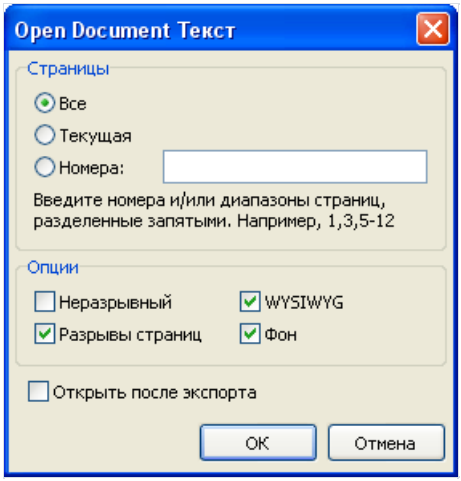
Экспорт в формат Open Document

OpenDocument Format (ODF, сокращённое от OASIS Open Document Format for Office Application — открытый формат документов для офисных приложений) — открытый формат файлов документов для хранения и обмена редактируемыми офисными документами, в том числе текстовыми документами (такими как заметки, отчёты и книги), электронными таблицами, рисунками, базами данных, презентациями. Этот стандарт был разработан индустриальным сообществом OASIS и основан на XML-формате, изначально созданном OpenOffice.org. 1 мая 2006 года принят как международный стандарт ISO/IEC 26300.

FastReport поддерживает экспорт в таблицу (расширение .ods) и текст (расширение .odt) OpenDocumet. Эти файлы могут быть открыты с помощью бесплатного офисного пакета OpenOffice.

Метод экспорта – табличный.

При экспорте будет предложено диалоговое окно для настройки параметров выходного документа.



Параметры экспорта:

- Неразрывный – непрерывный экспорт без разрывов страниц и таблиц документа с пропуском колонтитулов (колонтитул выводится только в начале первой страницы и в конце последней), полезен при выводе длинных документов, предназначенных для дальнейшей обработки;
- WYSIWYG – полное соответствие внешнему виду отчета, при отключении этой опции будет производиться оптимизация по уменьшению количества строк и столбцов в результирующей таблице;
- Фон – экспорт цвета заполнения, присвоенного странице отчета;
- Разрывы страниц – включает разрыв страниц в результирующем документе;
- Открыть после экспорта – результирующий файл будет открыт сразу же после экспорта.

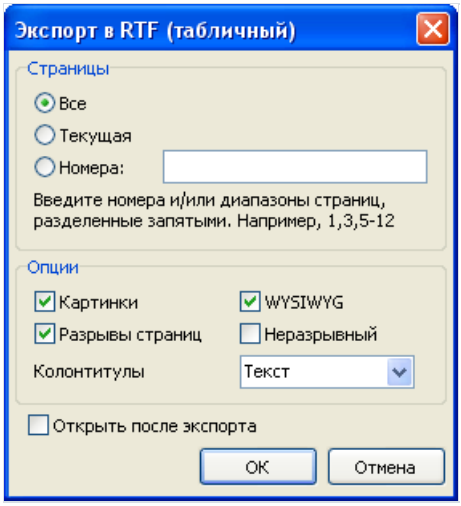
Особенности экспорта: RichText объекты передаются как простой текст, поддерживается передача графических изображений.

Экспорт в формат RTF

RTF (Rich Text Format) был разработан фирмой Microsoft как стандартный формат для обмена текстовыми документами. На данный момент RTF-документы совместимы с большинством современных текстовых редакторов и операционных систем.

Метод экспорта – табличный.

При экспорте в формат RTF будет предложено диалоговое окно для настройки параметров выходного файла.



Параметры экспорта:

- Картинки – включает возможность экспорта графических изображений в результирующий файл;
- Разрывы страниц – включает разрыв страниц в RTF файле;
- WYSIWYG – полное соответствие внешнему виду отчета, при отключении этой опции будет производиться оптимизация по уменьшению количества строк и столбцов в результирующей таблице;
- Неразрывный - непрерывный экспорт без разрывов страниц и таблиц документа с пропуском колонтитулов (колонтитул выводится только в начале первой страницы и в конце последней), полезен при выводе длинных документов, предназначенных для дальнейшей обработки;
- Колонтитулы - режим вывода колонтитулов страниц: Текст – выводятся как обычный текст, Колонтитулы – в итоговом документе создаются колонтитулы (внимание: такие переменные как номера страниц не поддерживаются), Нет – колонтитулы игнорируются;
- Открыть после экспорта – результирующий файл будет открыт сразу же после экспорта программой просмотра RTF файлов, назначенной в операционной системе по умолчанию (к примеру, Microsoft WordPad).

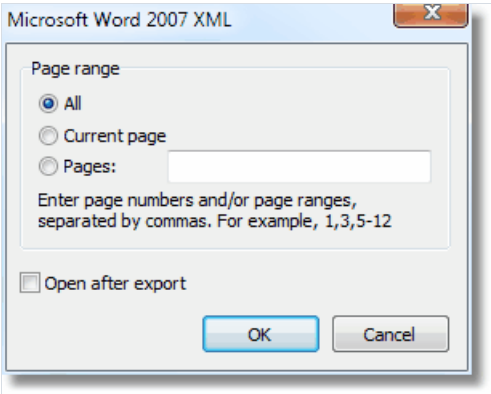
Особенности экспорта: RichText объекты полностью интегрируются в формат RTF, внешний вид и объем файла сильно зависят от шаблона отчета (раздел «Рекомендации по разработке отчетов»);

Экспорт в Word 2007

Word 2007 – приложение для работы с текстовыми документами, включенное в систему Microsoft Office 2007.

Метод экспорта – табличный.

При экспорте в Word 2007 будет предложено диалоговое окно для настройки параметров выходного документа:



Параметры экспорта:

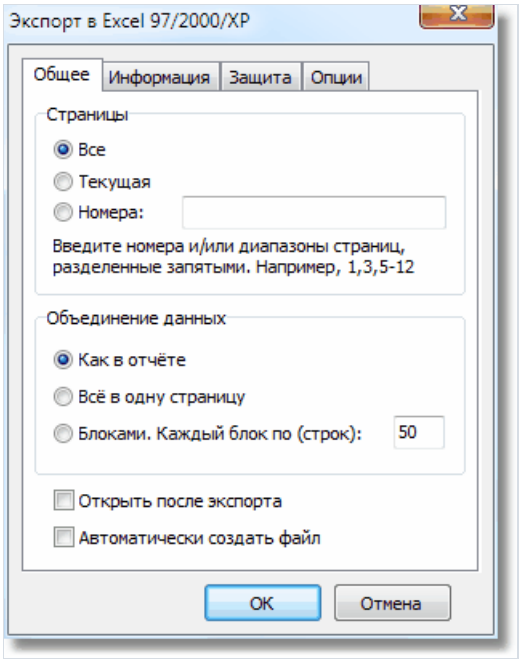
- Открыть после экспорта – результирующий файл будет открыт в Word сразу же после экспорта.

Экспорт в Excel 97/2000/XP

Excel – приложение для работы с электронными таблицами, включенное в систему Microsoft Office System.

Метод экспорта – табличный.

При экспорте в Excel будет предложено диалоговое окно для настройки параметров выходного документа.



Объединение данных:

- Как в отчете - каждая страница готового отчета экспортируется на отдельный лист Excel;
- Все в одну страницу - весь отчет экспортируется на один лист Excel, без разрывов страниц и колонтитулов (колонтитул выводится только в начале первой страницы и в конце последней);
- Блоками. Каждый блок по (строк) - на каждый лист Excel экспортируется указанное количество строк.

Параметры экспорта:

- WYSIWYG – полное соответствие внешнему виду отчета, при отключении этой опции будет производиться оптимизация по уменьшению количества строк и столбцов в результирующей таблице;
- Картинки – включает возможность экспорта графических изображений в результирующую таблицу;
- Линии сетки - включает отображение сетки в файле Excel;
- Подобрать размер - оптимизирует размеры ячеек, чтобы вместить содержащийся в них текст;
- Удалять пустые строки - удаляет из результирующей таблицы пустые строки;
- Экспортировать формулы - если текст в ячейке начинается с символа "=", то он экспортируется как формула Excel;
- Открыть Excel после экспорта – результирующий файл будет открыт сразу же после экспорта в Excel;
- Автоматически создать файл - создает файл со случайным именем и расширением .xls.

Особенности экспорта: RichText объекты передаются как простой текст, поддерживается передача

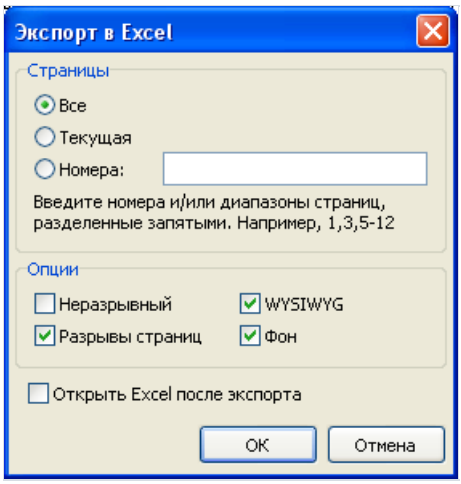
графических изображений.

Экспорт в Excel XML

XML (Extensible Markup Language) - расширяемый язык разметки. XML предназначен для хранения структурированных данных, а также для обмена информацией между различными программами. FastReport использует формат XML для передачи данных в табличный редактор Excel версии 2003 и более поздней.

Метод экспорта – табличный.

При экспорте в Excel XML будет предложено диалоговое окно для настройки параметров выходного документа.



Параметры экспорта:

- Неразрывный - непрерывный экспорт без разрывов страниц и таблиц документа с пропуском колонтитулов (колонтитул выводится только в начале первой страницы и в конце последней), полезен при выводе длинных документов, предназначенных для дальнейшей обработки;
- Разрывы страниц – включает разрыв страниц в результирующем документе;
- WYSIWYG – полное соответствие внешнему виду отчета, при отключении этой опции будет производиться оптимизация по уменьшению количества строк и столбцов в результирующей таблице;
- Фон – экспорт цвета заполнения, присвоенного странице отчета;
- Открыть Excel после экспорта – результирующий файл будет открыт сразу же после экспорта в Excel.

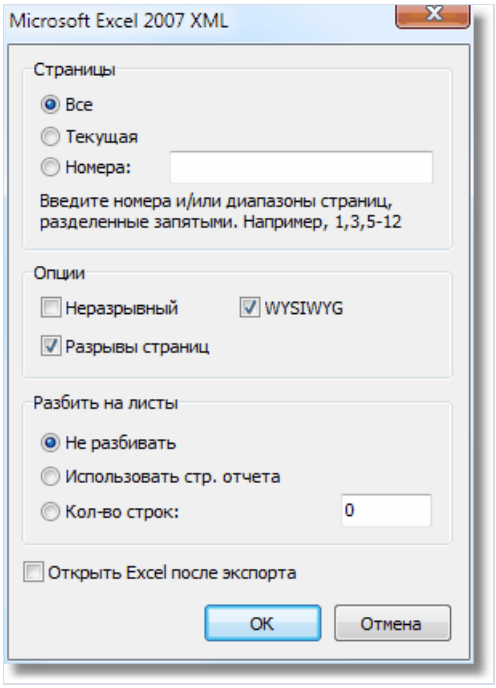
Особенности экспорта: RichText объекты передаются как простой текст, не поддерживается передача графических изображений.

Экспорт в Excel 2007

Excel 2007 – приложение для работы с электронными таблицами, включенное в систему Microsoft Office 2007.

Метод экспорта – табличный.

При экспорте в Excel будет предложено диалоговое окно для настройки параметров выходного документа:



Параметры экспорта:

- Неразрывный - непрерывный экспорт без разрывов страниц и таблиц документа с пропуском колонтитулов (колонтитул выводится только в начале первой страницы и в конце последней), полезен при выводе длинных документов, предназначенных для дальнейшей обработки;
- Разрывы страниц – включает разрыв страниц в результирующем документе;
- WYSIWYG – полное соответствие внешнему виду отчета, при отключении этой опции будет производиться оптимизация по уменьшению количества строк и столбцов в результирующей таблице;
- Открыть Excel после экспорта – результирующий файл будет открыт в Excel сразу же после экспорта.

Разбить на листы:

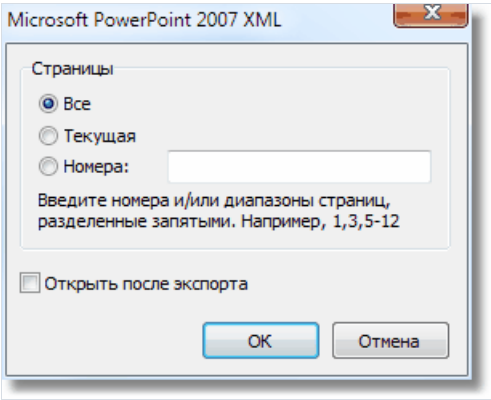
- Не разбивать - весь отчет экспортируется на один лист Excel;
- Использовать стр.отчета - каждая страница готового отчета экспортируется на отдельный лист Excel;
- Кол-во строк - на каждый лист Excel экспортируется указанное количество строк.

Экспорт в PowerPoint 2007

PowerPoint 2007 – приложение для работы с презентациями, включенное в систему Microsoft Office 2007.

Метод экспорта – послойный.

При экспорте в PowerPoint 2007 будет предложено диалоговое окно для настройки параметров выходного документа:



Параметры экспорта:

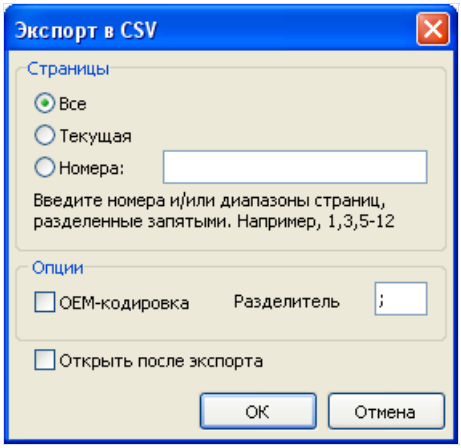
- Открыть после экспорта – результирующий файл будет открыт в PowerPoint сразу же после экспорта.

Экспорт в формат CSV

CSV-файл содержит значения, отформатированные в виде таблицы и упорядоченные таким образом, что каждое значение в столбце отделено от значения в следующем столбце разделителем, а каждый новый ряд начинается с новой строки. Данный формат может быть импортирован в различные табличные редакторы.

Метод экспорта – табличный.

При экспорте в CSV будет предложено диалоговое окно для настройки параметров выходного документа.



Параметры экспорта:

- OEM кодировка – выбор OEM кодировки результирующего файла;
- Разделитель – разделитель значений в файле;
- Открыть после экспорта – результирующий файл будет открыт сразу же после экспорта программой просмотра CSV файлов, назначенной в операционной системе по умолчанию.

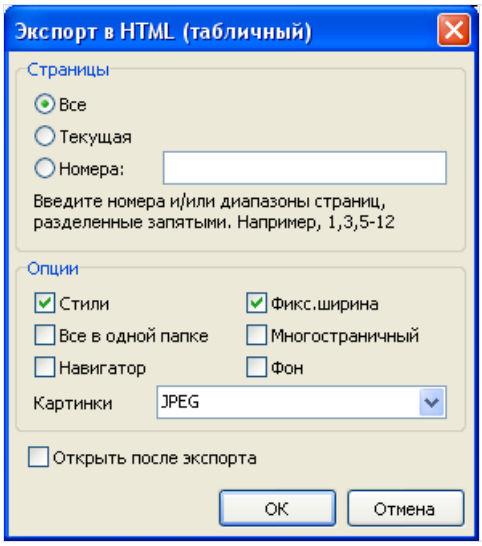
Особенности экспорта: оформление отчета при передаче в этот формат не сохраняется, графические изображения не поддерживаются.

Экспорт в формат HTML

HTML (Hypertext Markup Language) – считается стандартным языком для разметки документов в Internet. HTML создавался как язык для обмена научной и технической документацией, пригодный для использования людьми, не являющимися специалистами в области верстки. Служит для создания относительно простых, но красиво оформленных документов. Помимо упрощения структуры документа, в HTML внесена поддержка гипертекста.

Метод экспорта – табличный.

При экспорте в HTML будет предложено диалоговое окно для настройки параметров выходного документа.



Параметры экспорта:

- Стили – передача стилей оформления в текстовых объектов, отключение ускоряет процесс экспорта, но ухудшает внешний вид документа;
- Все в одной папке – все дополнительные файлы сохраняются в той же папке, где и главный файл;
- Навигатор – создается специальный навигатор для быстрого перемещения по страницам;
- Фикс.ширина – блокировка автоматического изменения ширины таблицы при изменении размера окна просмотра;
- Многостраничный – каждая страница будет записана в отдельный файл;
- Фон – экспорт графических атрибутов, присвоенных странице отчета;
- Картинки – включает возможность экспорта графических изображений;
- Открыть после экспорта – результирующий файл будет открыт сразу же после экспорта программой просмотра HTML файлов, назначенной в операционной системе по умолчанию.

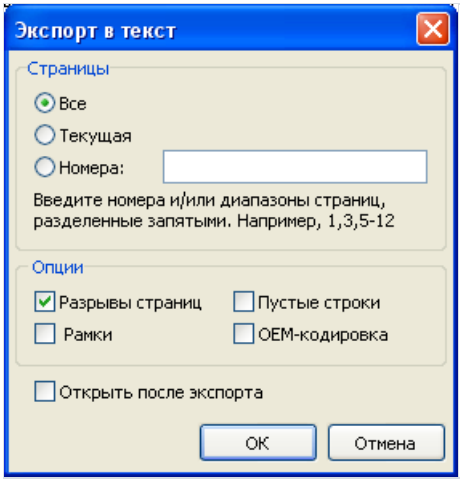
Особенности экспорта: экспорт может состоять из нескольких файлов, графические изображения поддерживаются и сохраняются каждое в своем файле, RichText объекты передаются как простой текст, внешний вид и объем файла сильно зависят от шаблона отчета (раздел «Рекомендации по разработке отчетов»).

Экспорт в текстовый формат

Обычный текстовый файл – содержит информацию из отчета, максимально оптимизированную и преобразованную в связи со спецификой данного формата.

Метод экспорта – табличный.

При экспорте в текст будет предложено диалоговое окно для настройки параметров выходного документа.



Параметры экспорта:

- Разрывы страниц – экспорт разделителей страниц в выходной файл;
- Пустые строки – экспорт пустых строк;
- Рамки – экспорт рамок текстовых объектов;
- OEM-кодировка – выбор OEM кодировки результирующего файла;
- Открыть после экспорта – результирующий файл будет открыт сразу же после экспорта программой просмотра текстовых файлов, назначенной в операционной системе по умолчанию.

Особенности экспорта: оформление отчета при передаче в этот формат не сохраняется, графические изображения не поддерживаются, ширина экспортируемой страницы автоматически рассчитывается в зависимости от вида текстовых объектов на странице отчета.

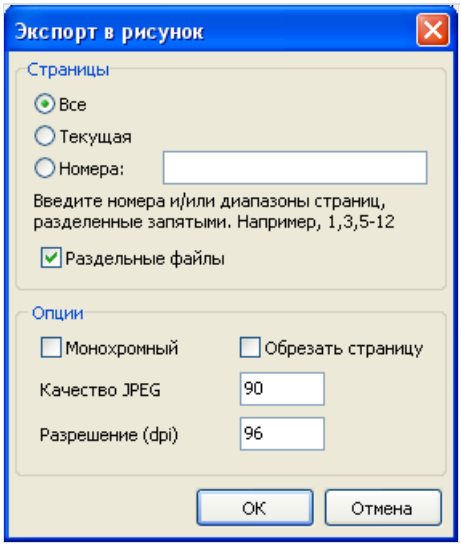
Экспорт в графические форматы Jpeg, BMP, Gif, Tiff

FastReport позволяет экспортировать информацию в графические форматы.

- JPEG (Joint Photographic Experts Group) – формат базирующийся на алгоритме сжатия, основанном не на поиске одинаковых элементов, а на разнице между пикселями. Отличается высоким уровнем компрессии за счет потери части графической информации.
- BMP (Windows Device Independent Bitmap) – применяется для хранения растровых изображений, предназначенных для использования в Windows. Стандартный формат файлов для компьютеров под управлением Windows.
- GIF (Graphics Interchange Format) – не зависящий от аппаратного обеспечения формат GIF был разработан для передачи растровых изображений по сетям. Позволяет неплохо сжимать файлы, в которых много однородных заливок (логотипы, надписи, схемы).
- TIFF, TIF (Target Image File Format) – аппаратно независимый формат, один из самых распространенных и надежных на сегодняшний день в полиграфии и передаче факсимильной информации.

Принцип экспорта – отрисовка.

При экспорте в один из вышеназванных графических форматов будет предложено диалоговое окно для настройки параметров изображения.



Параметры экспорта:

- Раздельные файлы – если опция включена, то каждая страница отчета будет экспортирована в отдельный файл, имя файла будет сформировано на основе выбранного с добавлением подчеркивания и номера страницы;
- Монохромный – создание черно-белого изображения;
- Обрезать страницу – после экспорта, будет произведено отсечение пустого места по краям;
- Качество JPEG – степень сжатия JPEG файла, опция активна, только при экспорте в Jpeg формат;
- Разрешение (dpi) – разрешение выходного графического изображения.

Особенности экспорта: При экспорте нескольких страниц в один файл (при отключенной опции "Раздельные файлы") нужно помнить о большой ресурсоемкости экспорта.

Рекомендации по разработке отчетов

Необходимо отметить, что качество экспорта в тот или иной формат сильно зависит от грамотной разработки шаблона отчета. FastReport предлагает избыточное количество возможностей для манипуляций объектам при создании шаблона отчета, что дает заметное преимущество при быстрой разработке любых отчетов и последующей их печати на принтере. Отпечатанный документ будет выглядеть точно так же, как и на экране, что и является основной целью применения генератора отчетов FastReport.

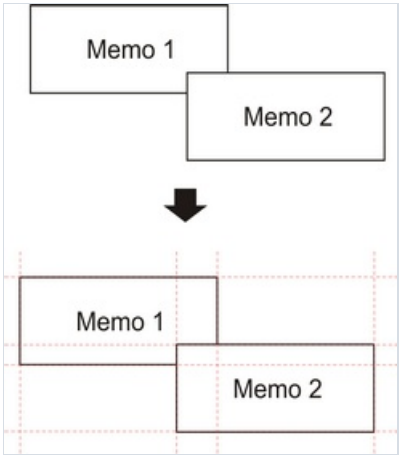
Обратная сторона такой свободы разработки – сложность экспорта полученного документа в различные форматы данных, имеющие свои, иногда довольно большие, ограничения в представлении информации. В этом разделе будут даны специальные рекомендации по разработке отчетов, предназначенных для экспорта в другие форматы данных.

Очень многие форматы используют табличное представление данных. В первую очередь речь идет о таких форматах, как HTML, XLS, XML, RTF, CSV. Никакие пересечения или наслоения ячеек в подобных форматах недопустимы (если брать в рассмотрение именно табличную разметку, это касается HTML и RTF), в отличии от свободы в процессе разработки шаблона отчета в дизайнере FastReport.

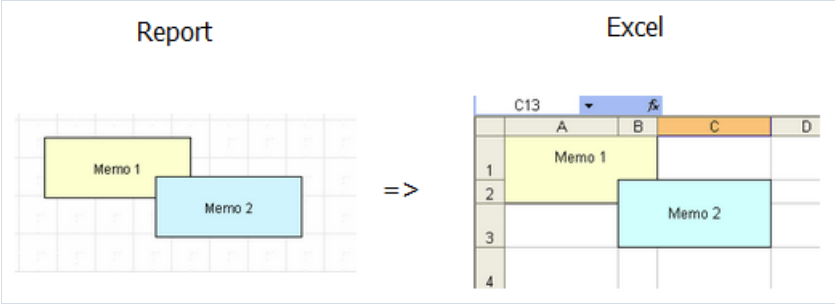
Фильтры экспорта, как правило, максимально учитывают эти требования при переносе объектов из отчета FastReport в необходимый формат. Это реализуется при помощи специальных алгоритмов учета пересечений объектов и оптимального их расположения.

В местах пересечений объектов возникают новые столбцы и строки в результирующей таблице. Это необходимо для сохранения точного позиционирования переносимых объектов FastReport и для получения максимального сходства результата и оригинального отчета.

Большое количество пересекающихся объектов в отчете приводит к росту числа столбцов и строк в таблице, что усложняет дальнейшее использование результирующего файла и замедляет процесс экспорта.

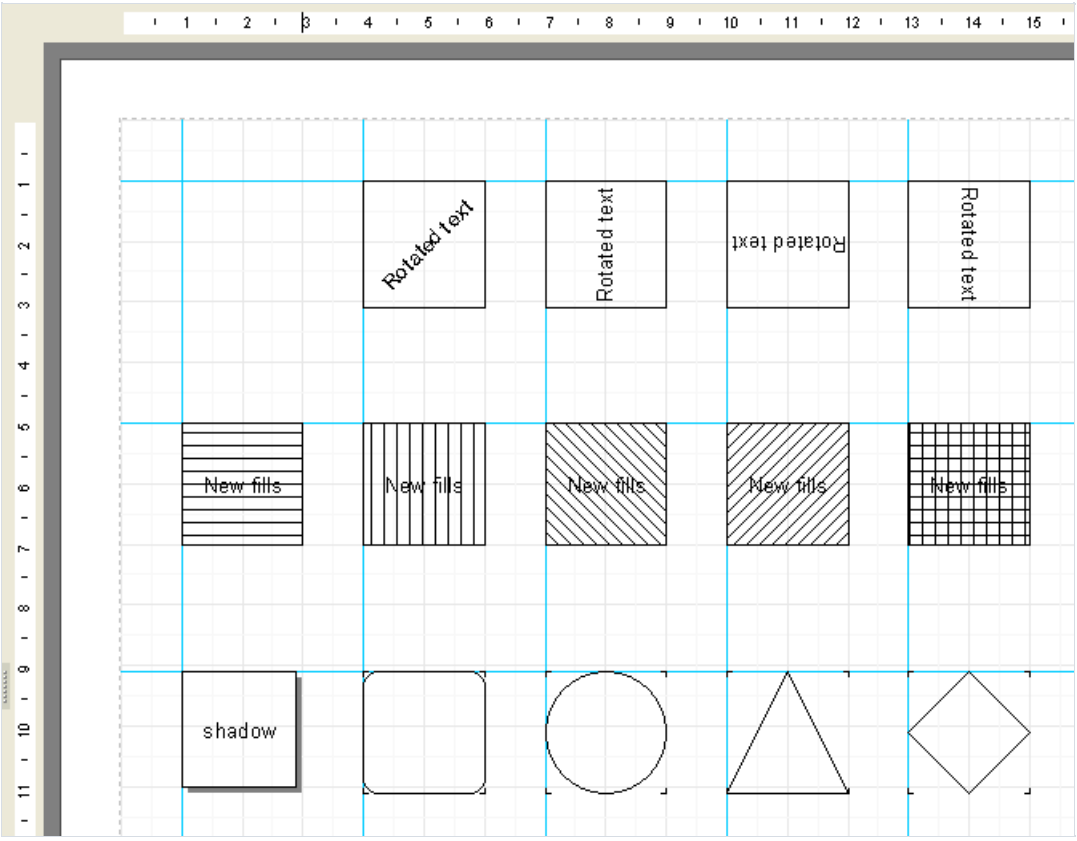


К примеру, при разработке отчета было допущено незначительное пересечение двух объектов расположенных один под другим находящихся на одном бэнде. Число записей при формировании отчета составило 150. При экспорте в формат RTF будет создано 450 строк в результирующей таблице (150 строк на каждый объект и 150 строк на пересечение). Если пересечение устранить, количество строк будет 300. На больших отчетах и при большем количестве объектов разница будет просто огромной, что, соответственно, скажется и на размере выходного файла.



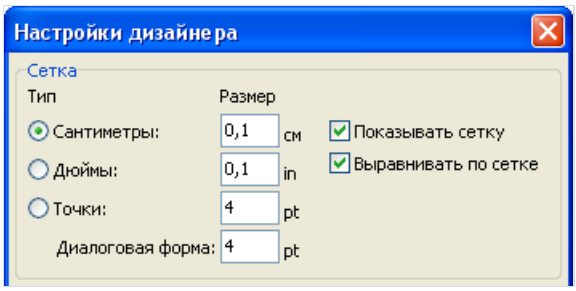
При создании таблиц в отчетах проследите, чтобы границы соседних ячеек соприкасались друг с другом. Важно, чтобы ячейки не пересекались и не наслаивались. Алгоритм фильтра экспорта сделает отсечение ячеек, но результат экспорта может быть далек от желаемого (вы увидите не совсем то, что хотели).

Располагайте объекты так, чтобы они находились на одной линии, как по вертикали, так и по горизонтали. В этом могут помочь выносные линии:



Для использования выносных линий в дизайнере FastReport просто кликните мышью на горизонтальную или вертикальную линейку, ограничивающую страницу отчета слева и сверху, и, удерживая кнопку мыши нажатой, перетащите выносную линию в нужную позицию на странице. В дальнейшем вы сможете располагать объекты непосредственно вдоль выносных линий по горизонтали и вертикали.

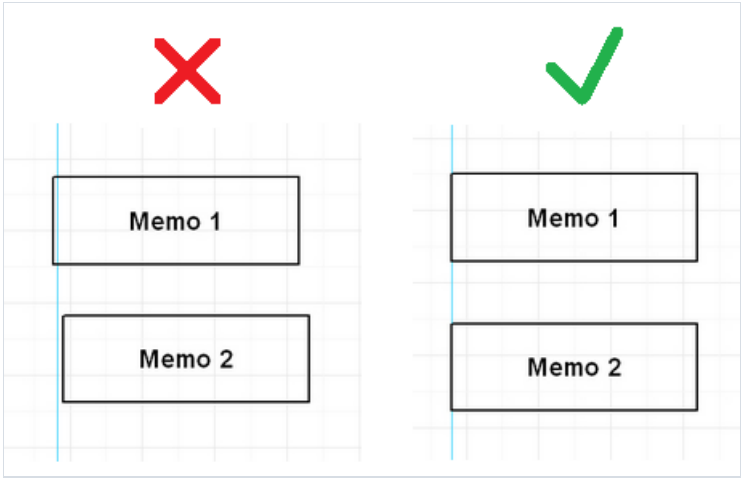
Избежать перекрытия ячеек также может помочь выравнивание текстовых объектов по сетке. Проследите за тем, чтобы было включено выравнивание по сетке в опциях дизайнера. Для упрощения выравнивания можно увеличить шаг сетки. Настройки шага сетки и выравнивания можно найти в меню дизайнера «Вид/Настройки...»:



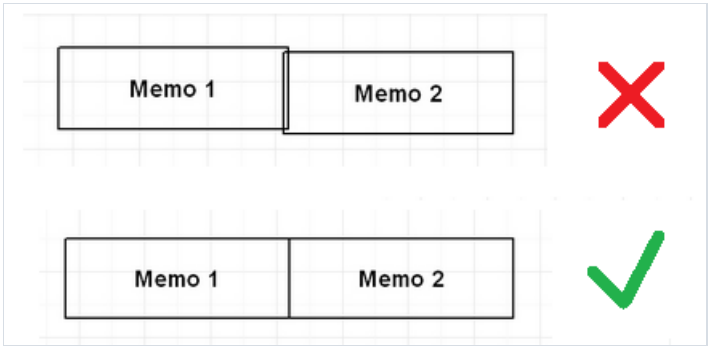
Для обрамления текста рамкой лучше использовать встроенные средства текстовых объектов, а не отдельные графические объекты – линии, прямоугольники и другое. Старайтесь не использовать фоновых объектов под прозрачными текстовыми объектами.

Применение этих простых правил на практике поможет вам создать отчет, который будет прекрасно выглядеть после экспорта в любой из форматов, которые используют табличную (или основанную на табличной) разметку для представления данных.

Ниже приведены примеры правильного и нежелательного расположения объектов при создании шаблона отчета:



Объекты смещены по горизонтали. Необходимо по мере возможности использовать выравнивание по выносным линиям, чтобы объекты имели одинаковую горизонтальную координату.



Объекты имеют перекрытие. В таком случае при экспорте в табличный формат будут созданы дополнительные бесполезные строки и столбцы, а также 3 дополнительных ячейки в зоне пересечения.

Рекомендуется также внимательно ознакомиться с демонстрационными отчетами идущими в комплекте поставки FastReport для освоения основных приемов оптимальной разработки отчетов.