

FastReport 4.0 User's Manual

Table of contents

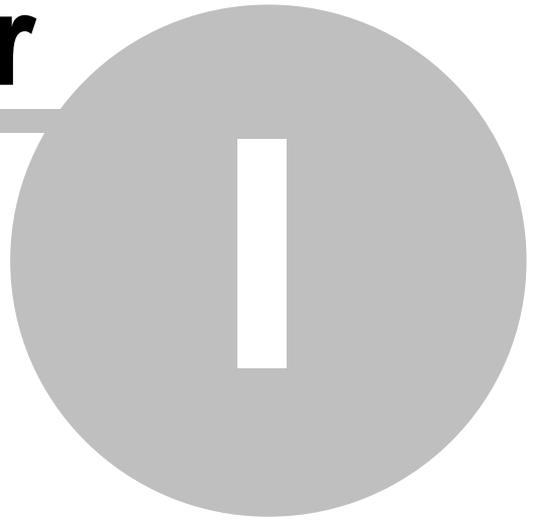
	1
Chapter I Designer	2
1 Control keys	3
2 Mouse control	3
3 Toolbars	4
Designer mode bar	4
"Standard" toolbar	5
"Text" toolbar	6
"Frame" toolbar	7
"Align" toolbar	7
4 Designer options	8
5 Report settings	9
6 Page options	12
Chapter II Creating reports	16
1 Report objects	16
2 "Hello, World!" report example	16
3 The "Text" object	17
4 HTML-tags in the "Text" object	19
5 Displaying expressions with the help of the "Text" object	20
6 Bands in FastReport	21
7 Databands	23
8 TfrxDBDataSet component	24
9 "Customer List" report	25
10 Displaying DB fields with the help of the "Text" object	27
11 Aliases	28
12 Variables	30
13 "Picture" object	32
14 Report with pictures	33
15 Multi-lined text displaying	35
16 Data splitting	37
17 Text wrap of objects	39
18 Displaying data in the form of a table	41
19 Printing labels	43
20 Child-bands	45
21 Shifting objects	47
22 Report with two data levels (master-detail)	48

23 Headers and footers of a data band	52
24 Multipage reports	54
Chapter III Groups, aggregates	57
1 Report with groups	57
2 Other group features	60
3 Reset page numbers	62
4 Drill-down groups	62
5 Lines numbering	63
6 Aggregate functions	64
7 Page and report totals	67
8 Inserting aggregate function	68
Chapter IV Formatting, highlight	71
1 Values formatting	71
2 Inline formatting	72
3 Conditional highlighting	74
4 Alternate color every other data row	75
Chapter V Nested reports (subreports)	78
1 Nested reports (subreports)	78
2 Side-by-side subreports	78
3 Limitations on using subreports	79
4 PrintOnParent option	80
Chapter VI Script	82
1 Taste of script	82
2 Structure of a script	85
3 "Hello, World!" script	87
4 Using objects in the script	88
5 Calling the variables from the report's variables list	88
6 Calling the DB fields	89
7 Using aggregate functions in the script	90
8 Displaying the variable's value in a report	90
9 Events	91
10 Example of using the "OnBeforePrint" event	92
11 Printing the group's sum total in the group's header	94
12 "OnAfterData" event	99
13 Service objects	99
"Report" object	100
"Engine" object	100

"Outline" object	102
14 Using the "Engine" object	103
15 Anchors	106
16 Using the "Outline" object	107
17 "OnManualBuild" page's event	111
18 Creation of objects in the script	116
Chapter VII Cross-tab reports	119
1 Construct a cross-report	120
2 Changing appearance	123
3 Using functions	125
4 Sorting values	126
5 Table with composite headers	126
6 Adjusting cell width	128
7 Font colors and highlighting	130
8 Managing a cross-table from the script	132
9 Adjusting rows/columns size	137
10 Filling a table manually	138
11 Add external objects to the table	140
12 Some useful settings	142
Chapter VIII Charts	147
1 Limitation of number of chart values	151
2 Some useful settings	152
3 Chart with specified values	152
4 Chart completion from Script	153
5 Printing of a chart built in Delphi	154
Chapter IX Dot-Matrix Reports	156
1 Cross-tab in dot-matrix	159
2 Dot-matrix reports printing	161
3 "Command" object	162
Chapter X Dialogue forms	164
1 Controls	164
2 "Hello, World!" report	166
3 Entering parameters and transferring them into a report	167
4 Interaction of controls	168
5 Several dialogue forms	168
6 Dialogue forms managing	169

Chapter XI	Data access components	173
1	Components' description	174
	TfrxDBLookupComboBox	174
	TfrxADOTable	175
	TfrxADOQuery	177
	TfrxADODataBase	179
2	Report constructing	179
3	Simple report of the "List" type	180
4	Report with parameters' query	182
5	Other useful settings	184
Chapter XII	Report inheritance	186
1	Creating a report	186
2	Changing a base report	188
3	Inheritance control	189
Chapter XIII	Wizards	192
1	New report wizard	192
2	New connection wizard	197
3	New table wizard	198
4	New query wizard	199
5	Query construction	199
	Query constructor usage	202
	Complicated query building	204
Chapter XIV	Report viewing, printing and export	208
1	Control keys	209
2	Mouse control	210
3	Report printing	210
4	Text search in report	213
5	Report Export	214
	Export to PDF Format	214
	Export to Open Document	216
	Export to RTF Format	217
	Export to Excel	218
	Export to XML Format	219
	Export to CSV Format	220
	Export into HTML Format	221
	Export to Text Format	222
	Export to Jpeg, BMP, Gif, Tiff Graphic Formats	223
6	Sending a Report via E-mail	224
7	Report Design References	226

Chapter

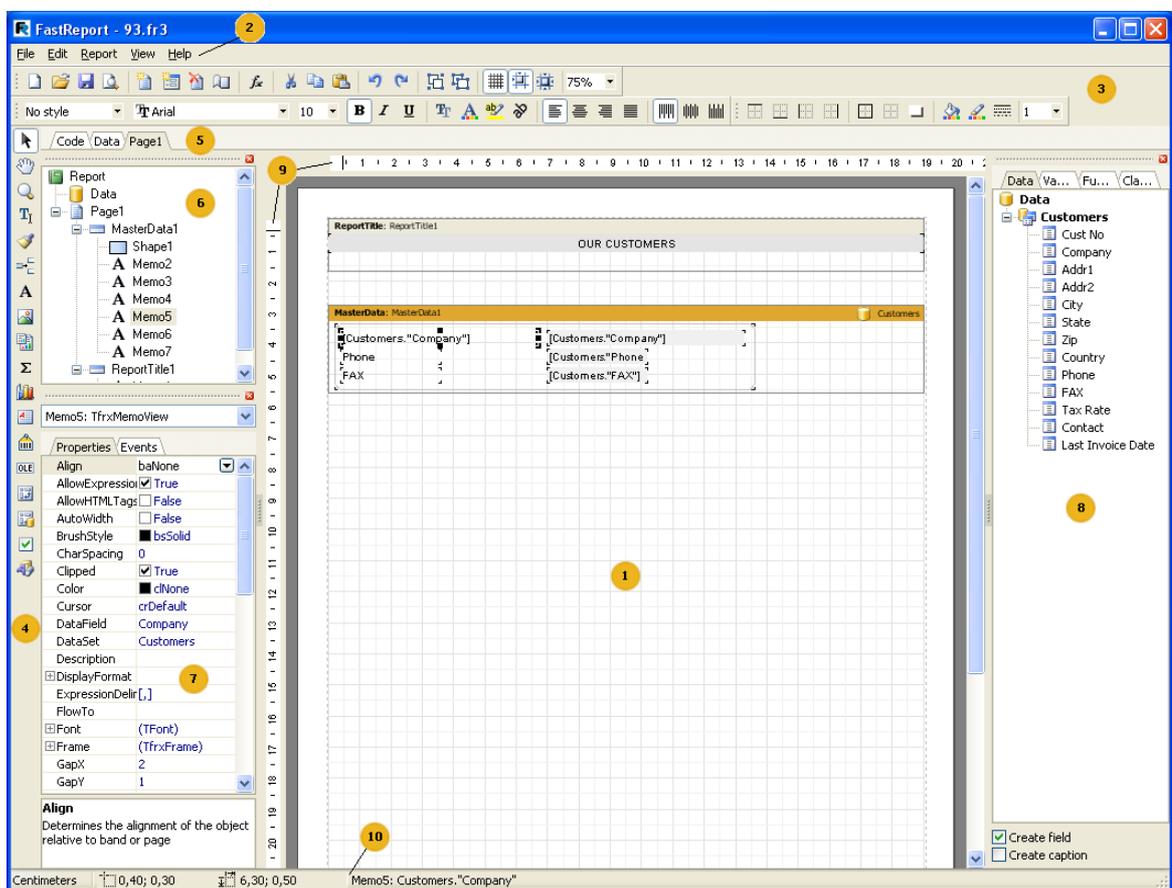


Designer

The report component is supplied with an embedded designer, which can be called in design-time by double-clicking on the TfrxReport component. The designer provides the user with all the tools for designing a report's appearance, along with the ability of simultaneous previewing. Designer's interface meets up-to-date requirements. It contains several toolbars, which can be docked wherever you want. The information about bar's location, along with all other designer settings is stored in a ini-file, if one is assigned, or in the registry, and is restored each time you launch the designer.

To give the end user of your project the ability to design reports, you should either add the "TfrxDesigner" component from the FastReport component palette, or add the "frxDesgn" unit into the uses list. Using the designer in run-time allows a user to set the report's appearance, as well as the ability to edit a finished report.

Note: you should also add any other additional Tfrx components you will use to the Delphi form as well.



In the picture, denoted with numbers:

- 1 – report design work space;
- 2 – menu bar;
- 3 – toolbars;

- 4 – object's toolbar;
- 5 – report pages' tabs;
- 6 – “Report tree” window;
- 7 – “Object inspector” window;
- 8 – “Data tree” window. You can drag elements to a report page from this window;
- 9 – rulers. When dragging a ruler to a report page, the guide line (which objects can be adhered to) appears;
- 10 – status line.

1.1 Control keys

Keys	Description
Ctrl+O	“File Open...” menu command
Ctrl+S	“File Save” menu command
Ctrl+P	“File Preview” menu command
Ctrl+Z	“Edit Undo” menu command
Ctrl+C	“Edit Copy” menu command
Ctrl+V	“Edit Paste” menu command
Ctrl+X	“Edit Cut” menu command
Ctrl+A	“Edit Select all” menu command
Arrow, Tab	Move between objects
Del	Delete of the selected objects
Enter	Call the editor of the selected object
Shift+arrows	Modify sizes of the selected objects
Ctrl+arrows	Move the selected objects
Alt+arrows	The selected object is adhered to the nearest one in the specified direction.

1.2 Mouse control

Operation	Description
-----------	-------------

Left button	Selecting an object; pasting a new object; moving and resizing objects. For the selected objects, you can perform zooming in and out by dragging the red square in the left bottom corner of the selected objects' group.
Right button	Selected object's contextual menu.
Double-click	Object editor call. Double-clicking on the white space of a page calls the "Page Settings" dialogue box.
Mouse wheel	Scrolling a report page.
Shift + left button	Toggle object's selection.
Ctrl + right button	If you hold the left mouse button during moving a mouse, a frame appears. As soon as you release the mouse button, all the objects captured in the frame would be selected. This operation can also be performed by clicking on the blank space on the page, and moving the mouse cursor to the required position.
Alt + left button	If the "Text" object is selected, it edits its contents in place.

1.3 Toolbars

1.3.1 Designer mode bar

The bar is integrated with the object bar and has the following buttons:

Icon	Name	Description
	Object selecting	A standard mode of operation, in which a cursor allows to select objects, modify their sizes, etc.
	Hand	Clicking on this icon allows dragging a report page.
	Zoom	Clicking on the left button doubles the zoom (adds 100%), while clicking the right one zooms out by 100%. When holding the left mouse button while dragging, the selected area would be zoomed.
	Text editor	Clicking on the "Text" object allows editing its contents right on the report page. If you hold the left mouse button when moving the cursor, the "Text" object appears in the selected place, and then its editor launches.

	Format copying	The button becomes enabled when the “Text” object is selected. When clicking on the “Text” object with the left button, it copies formatting, which has the previously selected “Text” object, into the object.
---	----------------	---

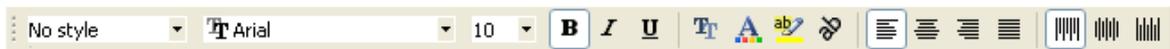
1.3.2 “Standard” toolbar



Icon	Name	Description
	New report	Creates a new blank report.
	Open report	Opens an existing report from the file. Hotkeys combination – “Ctrl+O.”
	Save report	Saves a report into the file. Hotkeys combination – “Ctrl+S.”
	Preview	Performs report constructing and its previewing. Hotkeys combination – “Ctrl+P.”
	New page	Adds a new page into the report.
	New dialogue form	Adds a new dialogue form into the report.
	Delete page	Deletes the current page.
	Page properties	Calls a dialogue with page properties.
	Variables	Calls the report variable’s editor.
	Cut	Cuts the selected objects into the clipboard. Hotkeys combination – “Ctrl+X.”
	Copy	Copies the selected objects into the clipboard. Hotkeys combination – “Ctrl+C.”
	Paste	Pastes objects from the clipboard. Hotkeys combination – “Ctrl+V.”
	Cancel	Undo the last operation. Hotkeys combination – “Ctrl+Z.”
	Repeat	Redo the last cancelled operation. Hotkeys combination – “Ctrl+Y.”
	Show grid	Shows the grid on the page. Grid pitch can be set in designer options.

	Grid alignment	During dragging or during modifying object sizes, the coordinates/sizes are modified step-wise, according to grid pitch.
	Locate in grid crosspoints	Modifies sizes/location of the selected objects so that they would be located at grid crosspoints.
	Zoom	Sets the zoom.

1.3.3 “Text” toolbar



Icon	Name	Description
	Style	Allows to select a style. To define the style list, call the “Report Styles...” menu item.
	Font	Allows to select font name from the drop-down list. Stores last five fonts previously used.
	Font size	Allows to select font size from the drop-down list. Size can also be entered manually.
	Bold	Enables/disables font bolding.
	Italic	Enables/disables italics.
	Underline	Enables/disables font underlining.
	Font	Displays Font settings dialog.
	Font color	Selects font color from the drop-down list.
	Highlighting	Shows the dialogue with highlighting attributes for the selected “Text” object.
	Text rotation	Allows to select text rotation.
	Left alignment	Enables text left alignment.
	Center alignment	Enables text center alignment.
	Right alignment	Enables text right alignment..
	Justify by width	Enables text justifying by width.
	Top alignment	Enables text top alignment..
	Height alignment	Enables text height alignment..
	Bottom alignment	Enables text bottom alignment.

1.3.4 “Frame” toolbar



Icon	Description	Description
	Top line	Enables/disables top frame line.
	Bottom line	Enables/disables bottom frame line.
	Left line	Enables/disables left frame line.
	Right line	Enables/disables right frame line.
	All lines	Enables all frame lines.
	No lines	Disables all frame lines.
	Shadow	Enables/disables shadow.
	Background color	Selects background color from the drop-down list.
	Line color	Selects line color from the drop-down list.
	Line style	Selects line style from the drop-down list.
1	Line width	Selects line width from the drop-down list.

1.3.5 “Align” toolbar

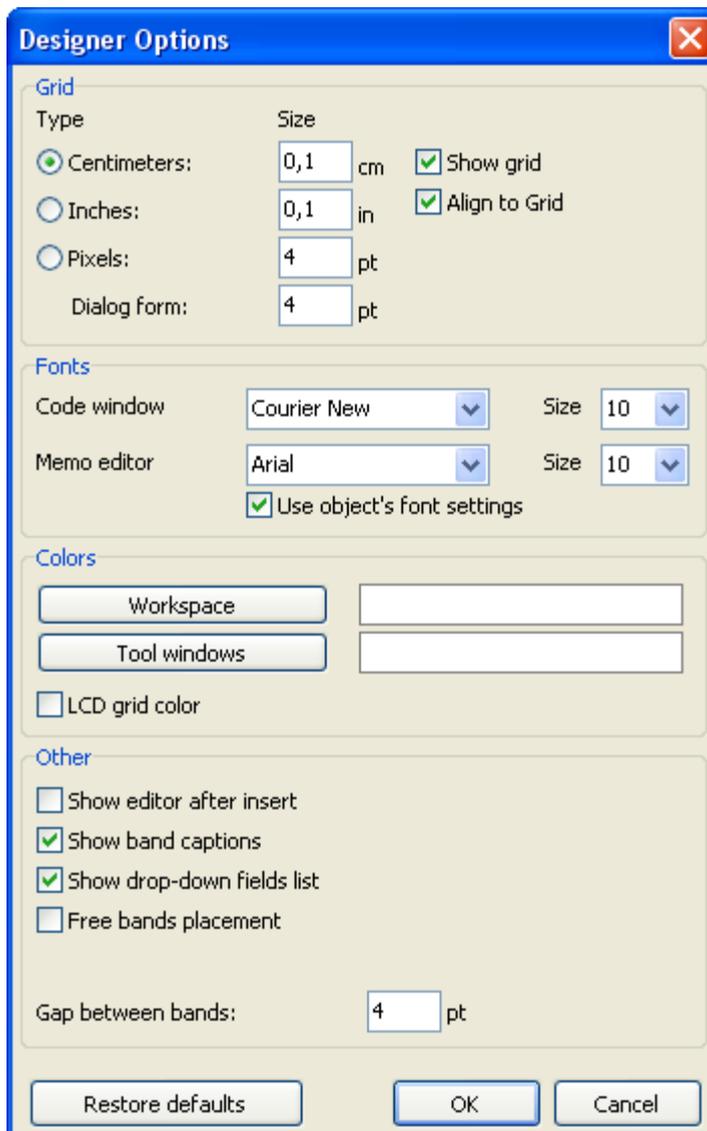


Icon	Description
	Align left edges.
	Center across.
	Align by right edges.
	Align top edges.
	Justify vertically.
	Align bottom edges.
	Justify by width.
	Justify by height.

	Center across in window.
	Center vertically in window.
	Set the same width as in the first selected object
	Set the same height as in the first selected object

1.4 Designer options

Set the designer options via the “View|Options...” menu command.



Designer Options

Grid

Type Size

Centimeters: 0,1 cm Show grid

Inches: 0,1 in Align to Grid

Pixels: 4 pt

Dialog form: 4 pt

Fonts

Code window Courier New Size 10

Memo editor Arial Size 10

Use object's font settings

Colors

Workspace []

Tool windows []

LCD grid color

Other

Show editor after insert

Show band captions

Show drop-down fields list

Free bands placement

Gap between bands: 4 pt

Restore defaults OK Cancel

Here you can set the desired units (centimeters, inches, pixels), and specify grid

step for each unit. Tip: You can also switch units in the designer by double-clicking on the left part of the status line where the current units are displayed.

You can also specify whether grid should be displayed, and align to grid. This can be done by buttons in the “Standard” toolbar of the designer as well.

You can set a font for the code editor window and for the “Text” object editor. If the “Use object's font settings” option is enabled, the font in the text editor window would correspond with the font of the object being edited.

The default white background of the designer and service windows can be modified by the “Workspace” and the “Tool windows” buttons.

The “LCD grid color” option increases contrast of the grid lines a little, and it improves their visibility on LCD monitors.

The “Show editor after insert” option controls what happens when new objects are inserted. If the option is enabled, its editor will be displayed each time an object is inserted. When creating a large number of blank objects, it is recommended to disable the option.

Disabling the “Show band captions” option, you can disable bands’ captions in order to save some place in a page. At that, the band’s captions would be written inside of it.

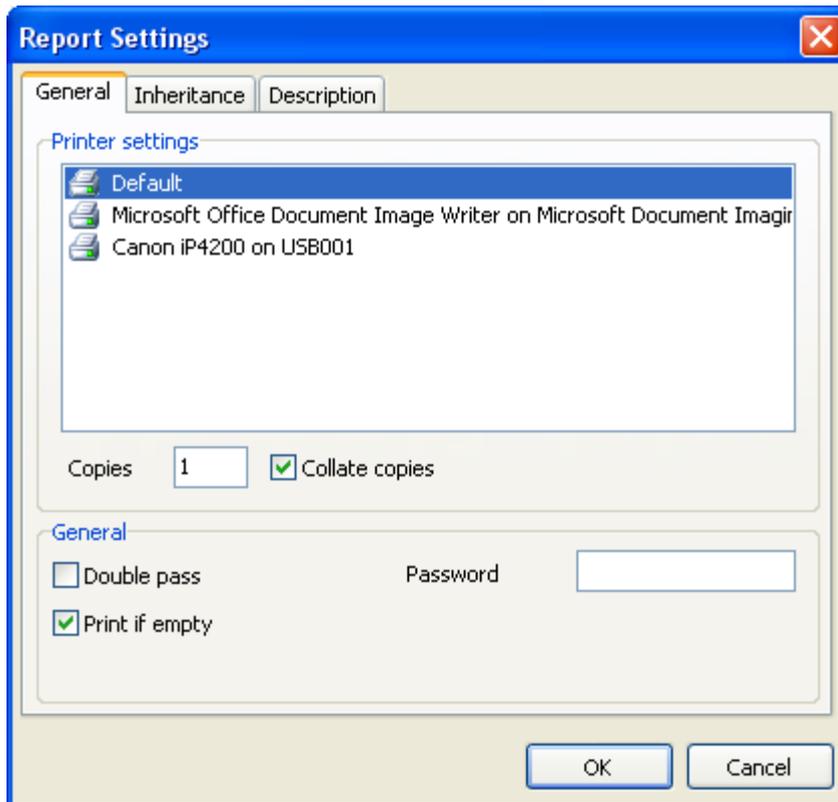
The “Show drop-down list of fields” option stops the drop-down list from displaying when pointing with the mouse to a “Text” object which is connected to data fields. This may be necessary if there are many narrow text objects in a band

The “Free band placement” option disables snapping of bands to the page. This option is disabled by default and bands are automatically grouped in page according to their function. A gap between bands can be set in the “Gap between bands” field.

1.5 Report settings

A window with report parameters is available from the “Report|Options...” menu. A dialogue has three pages.

In the first page you can see the general settings of the report:



You can tie a report to one of the printers installed in the system. This means that report printing will be performed by the selected printer by default. This might be useful in case when there are several different printers in the system; e.g. text documents can be tied with monochrome printer, while documents with graphics - to the color one. There is the "Default printer" item in the list of printers. When this item is selected, the report will not be tied with any printer, and therefore printing will be performed by a printer, which is set as the default one.

You can also set number of report copies to be printed and specify, whether it is necessary to perform collation. The values, which a user sets in this dialogue, would be displayed in the "Print" window.

If the "Double pass" flag is selected, report's formation will be performed in two steps. During the first pass, a report is formed, and is divided into pages, but the result is not saved anywhere. In the second pass a standard report formation with saving a result in the stream is performed.

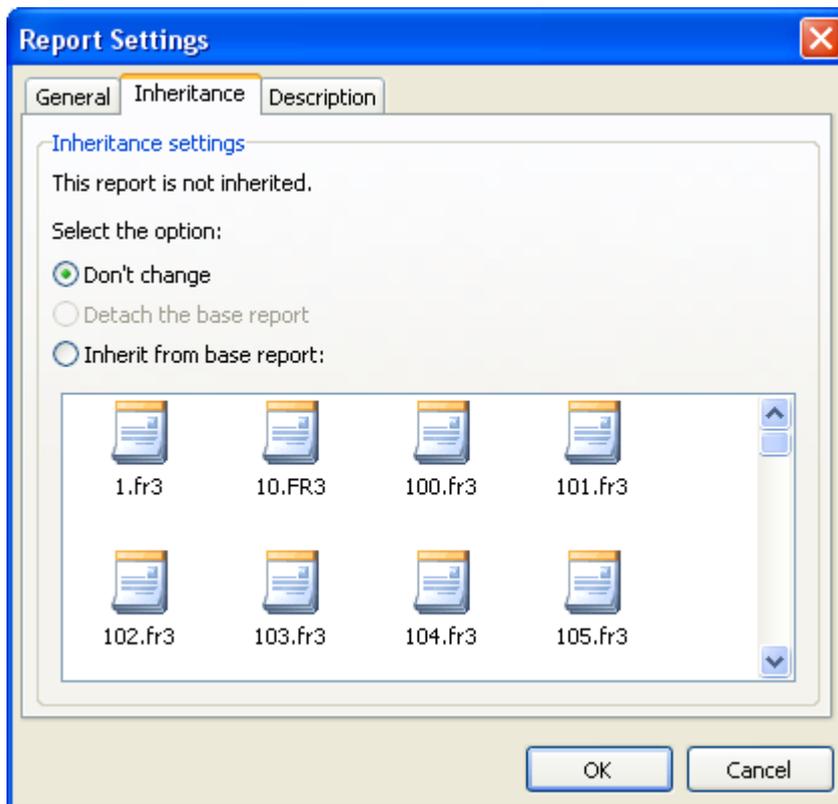
Why two passes are necessary? Most often, this option is used in cases when in a report there is a need for the total number of pages in it, i.e. the information of the "Page 1 of 15" type. The total number of pages is calculated during the first pass and is available via the "TOTALPAGES" system variable. The most frequent mistake is an attempt to use this variable in a single-pass report; in this case it returns "0."

Another use is to perform any calculations in the first pass and display the results in the second pass. For example, when it is required to display a sum in the group header, which usually is calculated and displayed in the group footer. Calculations of this type are accomplished by writing report script code in the OnBeforePrint event of an object.

The “Print if empty” flag allows construction of a report, which contains no data lines. If this option is disabled, blank reports would not be constructed.

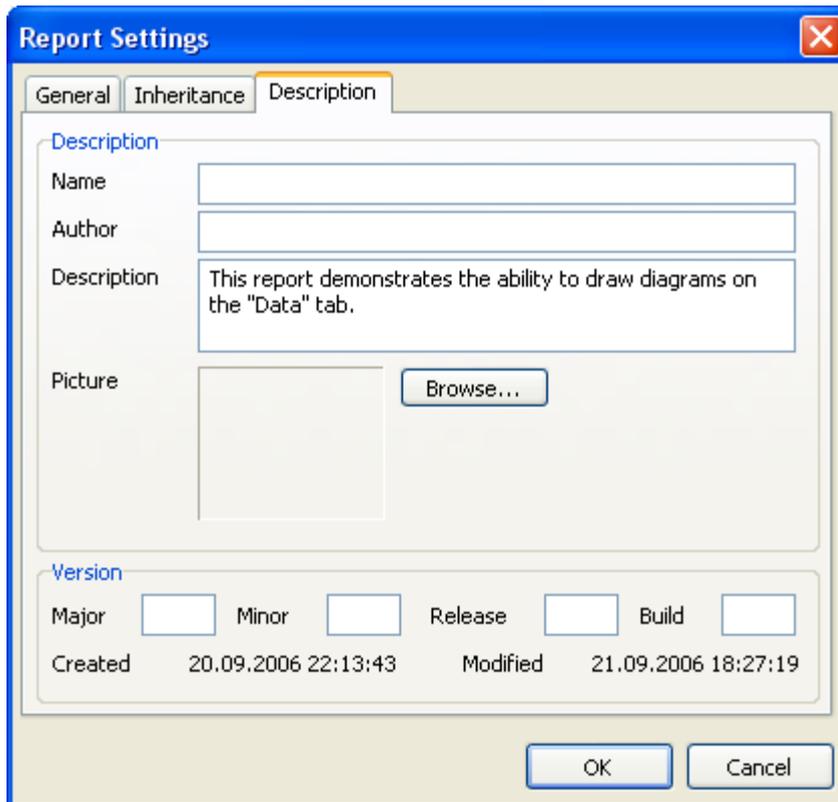
The “Password” field enables setting of a password which must be entered before opening a report.

On the second page you can setup the report inheritance options:



You can learn about inheritance later in the "Report inheritance" chapter. In this dialog you can see the base report's name (if report is inherited), detach the base report (in this case your report will be standalone, non-inherited) and inherit the report from one of base reports.

Controls in the third page of the dialogue allows you to set report's description properties..



All fields in this dialog are for information purposes only.

1.6 Page options

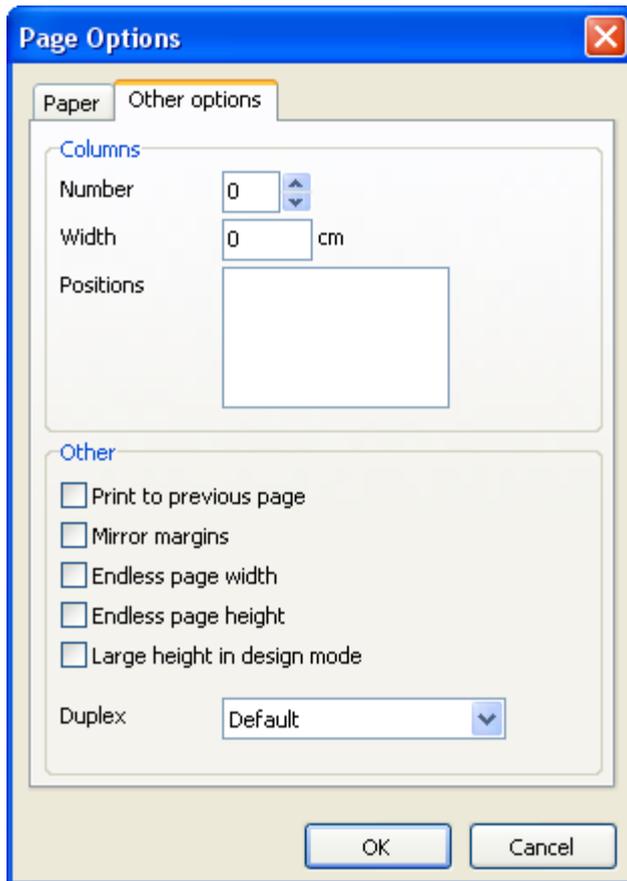
Page's parameters are available via either the "File|Page settings..." menu, or by double-clicking on page blank space. The dialogue has two pages:

The image shows a 'Page Options' dialog box with a blue title bar and a close button (X) in the top right corner. The dialog has two tabs: 'Paper' (selected) and 'Other options'. The 'Paper' tab contains several sections:

- Size:** A dropdown menu is set to 'A4'. Below it are input fields for 'Width' (21 cm) and 'Height' (29,70 cm).
- Orientation:** Two radio buttons are present: 'Portrait' (selected) and 'Landscape'. To the right is a small icon of a document with the letter 'A' on it.
- Margins:** Four input fields are arranged in a 2x2 grid: 'Left' (1 cm), 'Right' (1 cm), 'Top' (1 cm), and 'Bottom' (1 cm).
- Paper Source:** Two dropdown menus are shown: 'First page' (Default) and 'Other pages' (Default).

At the bottom of the dialog are two buttons: 'OK' and 'Cancel'.

On the first dialogue page, you can select size and alignment of paper, as well as set margins. In “Paper source” drop-down lists you can select a printer tray for the first page and the rest of the report pages.



On the second page, you can set the number of columns for multi-column reports' printing. The current settings are displayed in the designer.

The "Print to previous page" flag allows you to print pages, beginning from blank space of the previous page. This option can be used in case when a report template consists of several pages or when printing batch (composite) reports.

The "Mirror margins" option switches right and left margins of page for even pages during previewing or printing a report.

The "Endless page width & height" option increases page's sizes depending on number of data records on the page (when running a report). In this case you will see one big page in the preview window instead of several standard size pages.

The "Large height in design mode" option increases page's height several times more. This feature can be useful if many bands are located in the page, and must be used when working with the overlay band. This only effects the page height in design mode..

Chapter



Creating reports

2.1 Report objects

A blank report in FastReport is presented as a paper page. At any place on the page, a user is able to add objects, which can display different information (such as text and/or graphics), as well as to define report's appearance. Let us describe briefly the assignment of the FastReport objects, which are included in the standard package:

 - "Band" object allows creation on a design page, of an area which has definite behaviour; according to it's type.

 - "Text" object displays one or several text lines within the rectangular area;

 - "Picture" object displays a graphic file in "BMP," "JPEG," "ICO," "WMF," or "EMF" format;

 - "Line" object displays a horizontal or a vertical line;

 - "System text" object displays service information (date, time, page number, etc), as well as aggregate values;

 - "Subreport" object allows inserting an additional report design page inside the basic one;

 - the objects of "Draw" category represent different geometrical figures (diagonal line, rectangle, rounded rectangle, ellipse, triangle, and diamond);

 - "Chart" object displays data in the form of charts of different kinds (circle chart, histogram, and so on);

 - "RichText" object displays a formatted text in Rich Text Format (RTF);

 - "CheckBox" object displays a checkbox with either a check or a cross;

 - "Barcode" object displays data in the form of barcode (many different types of barcodes are available);

 - "OLE" object is able to display any object using OLE technology.

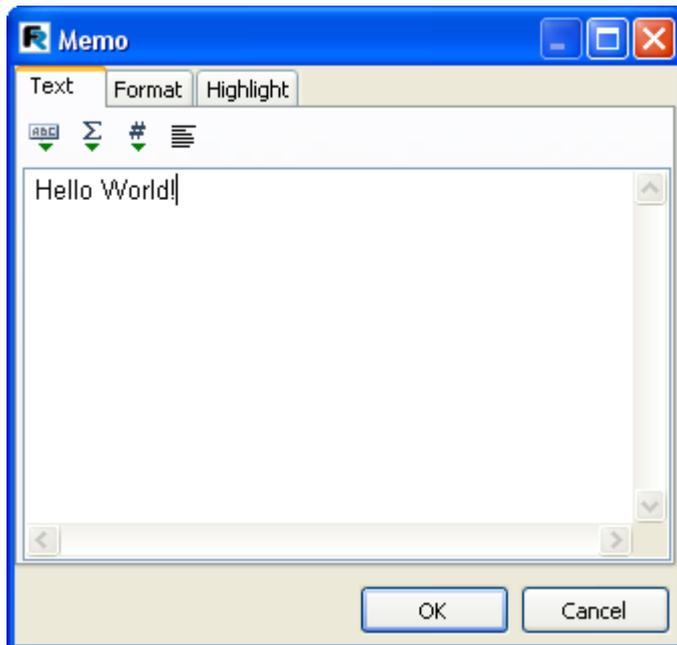
The basic objects you most likely need to work with are the "Band" and "Text" objects. You will learn about their capabilities in detail further in this chapter.

2.2 "Hello, World!" report example

The report will contain one inscription only ("Hello, World!"). Open the report designer. After that, click the "Text" button in the "Objects" designer panel. Move the mouse cursor to the desired place on the page, and click again. The object has been inserted.



The text editor window will be displayed right away; if it does not appear (this can be set in the designer settings), then doubleclick the object. Enter the “Hello, World!” text, and then click the “ ” button.



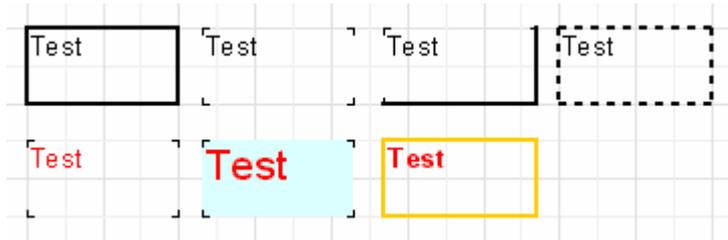
The report is created. To preview it, either select the "File|Preview" menu item, or click the corresponding button in the toolbar. The preview window containing a report page with the “Hello, World!” text will appear. This report can be printed out, saved to a file (*.fp3), or exported to one of the supported export formats.

2.3 The “Text” object

The “Text” object has many features. Now we already know that it allows you to display text, a frame, and fill color. The text can be displayed using any font of any size and style. All the parameters can be set visually, with the help of the toolbars:



Here are some examples of text design:

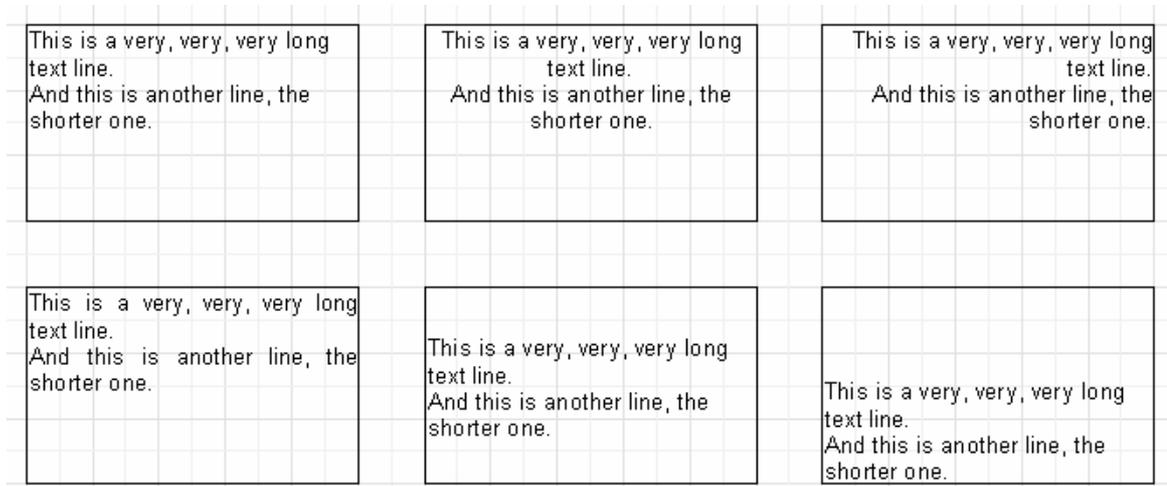


Now let's look at other features of this basic object. As an example, let us create a new text object and put two lines into it:

*This is a very, very, very long text line.
And this is another line, the shorter one.*

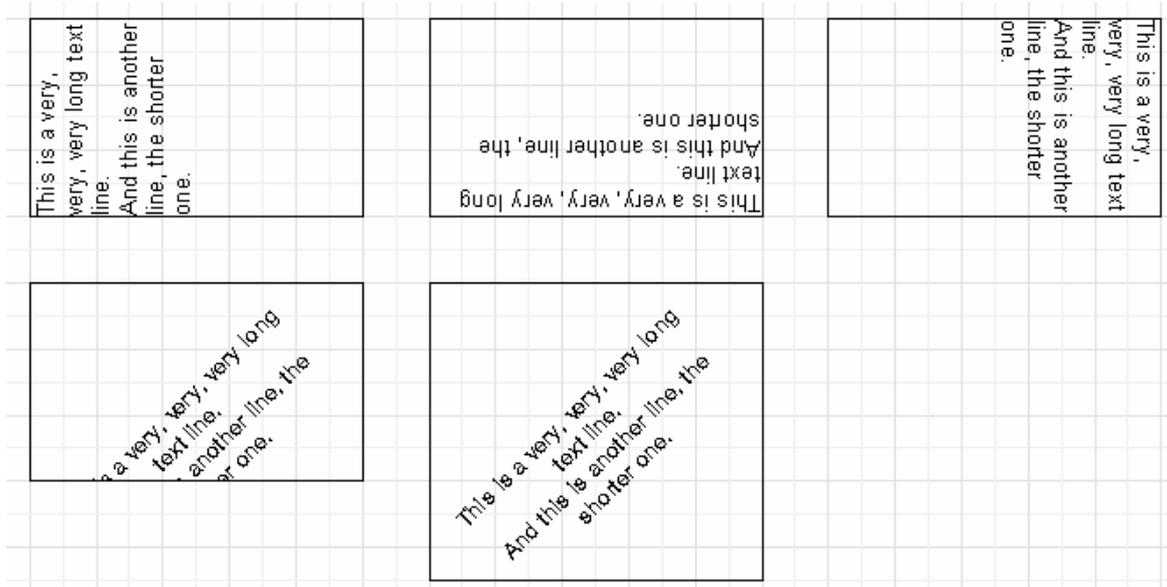
Let us enable the object frame, and then resize the object up to 9x3 cm with the help of the mouse. We see that the object can display not only a single line of text, but several lines as well. Now let us reduce the object width up to 5cm. It is obvious that long lines did not find room in the object and therefore were wrapped. This happens due to the "WordWrap" object property. If it is disabled (either in inspector or in the object context menu), the long lines will be simply cut off.

Now let's check how the text alignment inside the object works. Alignment buttons are located in the "Text" toolbar and allows one to set horizontal or vertical text alignment. Pay attention to the "Block Align" button; this button let's you align the paragraph on both object edges. To do this, the "WordWrap" property option must be enabled.



All the text in the memo can be rotated at any angle within the limit of 0.. 360 degrees. The  button in the "Text" toolbar allows you to quickly rotate the text at 45, 90, 180 and 270 degrees. If you wish to rotate the text at any other value, use the object inspector. The "Rotation" property sets the required angle. When rotating a text, setting values other than 90, 180, 270 the text can exceed bounds of the object, as in our case (see

the picture below). Let us increase object height a little, so that the text would fit the object.



Let us briefly examine some other “Text” object properties, which influence its appearance. Most of these properties are available in the object inspector only:

- BrushStyle – type of object filling;
- CharSpacing – space between symbols in pixels;
- GapX, GapY – text indents from object’s left and top boundaries (in pixels);
- LineSpacing – space between lines (in pixels);
- ParagraphGap – the first paragraph line indent (in pixels).

2.4 HTML-tags in the “Text” object

Yes, this object does understand some simple HTML tags. Tags can be located inside the text of an object. Tags are disabled by default; to enable them, either select the “Allow HTML tags” item in the object context menu, or enable the “AllowHTMLTags” property in the object inspector. Here is the list of supported tags:

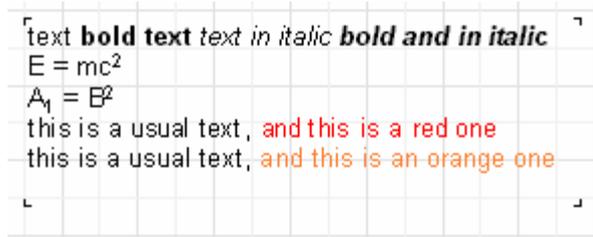
- - bold text
- <i> - text in italic
- <u> - underlined text
- <sub> - subscript
- <sup> - superscript
- - font color

Notice that not too many tags are supported, but it is rather enough for the majority of applications. It is impossible to modify font size and name; otherwise the

text-rendering unit in FastReport would become very complex.

The following examples demonstrate how these tags can be used.

```
text <b>bold text</b> <i>text in italic</i> <b><i>bold and in italic</i></b></i>
E = mc<sup>2</sup>
A<sub>1</sub> = B<sup>2</sup>
this is a usual text, <font color=red>and this is a red one</font>
this is a usual text, <font color="#FF8030">and this is an orange one</font>
```



2.5 Displaying expressions with the help of the “Text” object

One of the most important features of this universal object is its ability to display not only a static text, but expressions as well. Expressions can be located in the object together with text. Let us examine a simple example of how it can be performed. Put the following line into the object:

Hello, World! Today is [DATE].

Thus, when running the report, we can get something like follows:

Hello, World! Today is 01.01.2004.

What lead to such result? During FastReport report building, if an expression enclosed in square brackets is encountered, the engine calculates its value and inserts the value into the text (in place of the expression). The “Text” object can contain any number of expressions, together with a usual text. Both single variables and expressions can be enclosed in brackets (for example, [1+2*(3+4)]). Any constants, variables, functions, and DB fields can be used in expressions. We will learn more about these features later, in the chapter.

FastReport automatically recognizes expressions enclosed in square brackets in the text. But what should be done if our object contains square brackets, and we do not want them to be considered as expressions? For example, if we need to display such text as following:

a[1] := 10

FastReport considers [1] as an expression, and displays the following:

```
a1 := 10
```

that is not what we want, of course. One of the ways to avoid such a situation is to disable the expression. Just disable the “AllowExpressions” property (“Allow Expressions” in the context menu), therefore all the expressions in the text will be ignored. In our example, FastReport would then display exactly what we need:

```
a[1] := 10
```

Sometimes text is required to contain both an expression and a text in square brackets, for example:

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

Disabling of an expression allows us to display square brackets in the required place, but it also disables handling of expression. In this case, FastReport allows you to use another set of symbols to designate the expression. The “ExpressionDelimiters” property, which is equal to “[,]” by default, is responsible for it. In this case, the user can use angular brackets for the expressions, instead of square ones:

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

The “<, >” value must be set in the “ExpressionDelimiters” property. As you can see, the comma divides opening and closing symbols. There is one limitation however: the opening and closing symbols cannot be similar, so “%, %” will not work. One can set several symbols, for example “<%, %>” Thus, our example will look as follows:

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

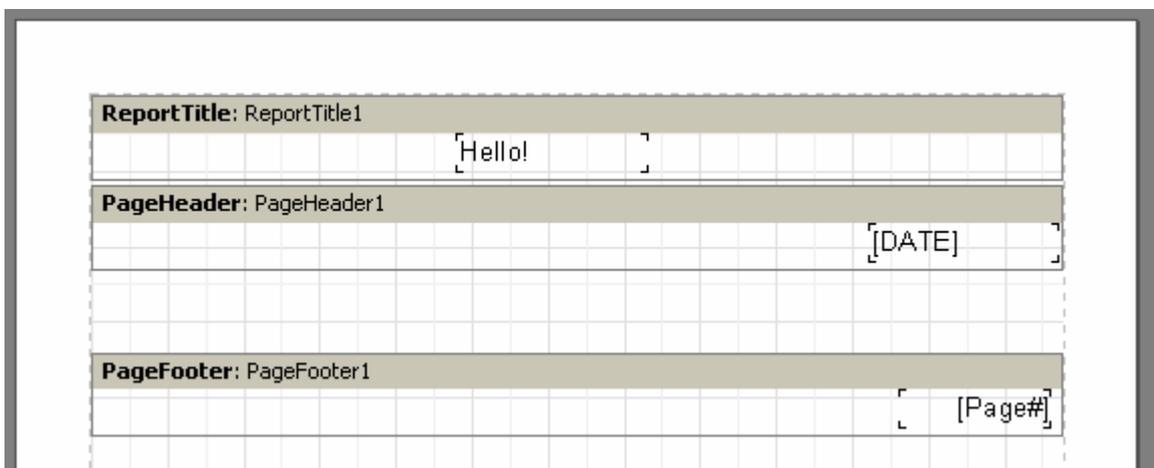
2.6 Bands in FastReport

Bands are used for logically placing the objects it contains at a location on the output page. When placing an object in a band, such as “Page Header,” we tell the report engine that the given object must be displayed at the top of each page of a finished report. Similarly, the “Page Footer” band is displayed at the bottom of each page together with all the objects it contains. Let us demonstrate it with an example. We’ll create a report, which contains the “Hello!” text at the top of the page, a current date to the right of it, and a page number at the foot of the page (to the right).

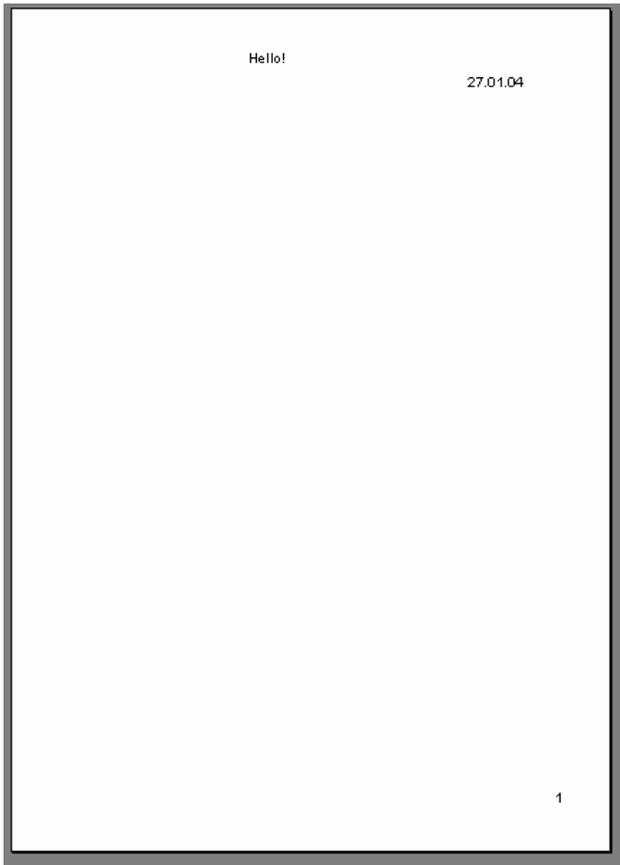
Open the FastReport designer and click the “New report” button in the toolbar. You will see a report template, which already contains three bands: “Report title,”

“Master data,” and “Page footer.” Let us remove the “Master data” band for a while (to do that, click either on any free space inside the band, or on its header, and then remove it by pressing the “delete” key or using the corresponding section in the context menu). Now let us add a new band (“Page header”). To perform this, click the “Add band” button and select “Page header” from the drop down list. We see that a new band is added to the page. At the same time, the existing bands were moved down. FastReport designer automatically positions bands on the page, and, as a result, header-bands are positioned on the top, data-bands are in the middle, and footer-bands are at the bottom.

Now let's add some objects. Add a “System text” object in the “Page header” band and select “System variable” in its editor “[DATE]” (you should remember that the date can be displayed in a “Text” object by typing “[DATE]” in its editor). We add a “Text” object, which will contain the “Hello!” text in the “Report title” band. Note you can see, the text object, which displays page number, is already added to the “Page footer” band.



When running the report, you will see that the objects in the finished report are allocated on the page in the appropriate position.



Thus, bands are responsible for object positioning in required places. Depending on band type, we can add objects either at the top or at the bottom of the page, on the first page, or on the last one. The basic bands, which we would need in most reports, work in the following way:

- “Page header” band is displayed at the very top of each page;
- “Page footer” band is displayed at the very bottom of each page;
- “Report title” band is displayed at the top of the first page, but below the “Page header” band (depending upon the “TitleBeforeHeader” page property assigned in the object inspector);
- “Report summary” band is displayed at the very end of a report, at white space.

2.7 Databands

Now, we are about to learn how to print the data from DB tables or queries. What is considered a table in such case? It is a required number of lines (records/rows), each of which has a certain number of columns (fields). To print information of this kind, FastReport uses a special type of band (databands). These are bands with names of “xxx data level” type. To print a whole table or some of its fields, you must add these band(s) to the report, connect it to the table, and add to it the objects with the fields you want to be

printed out. When FastReport builds these bands, they will be printed on the output page as many times, as there are records in the table. If there is no free space left on the output page, new output pages will be formed as needed by the report engine

2.8 TfrxDBDataSet component

The “TfrxDBDataSet” connector component  from the FastReport component palette is used in order to connect a table (or any other data source). This component plays a role of a messenger between the data source and the FastReport core. The component is responsible for record navigation and field reference. This allows the FastReport core to be independent from any data access library. FastReport can simultaneously work both with “BDE,” “IB_Objects” (with their non-standard implementation, incompatible with TDataSet), and other libraries, as well as to receive data from a source, not connected with DB, for example, from an array or a file. TfrxDBDataSet component is intended for working with data sources, compatible with TDataSet (such as BDE, ADO, IBX and a great majority of other libraries). The “TfrIBODataSet” component is intended for working with IB_Objects. The “TfrUserDataSet” component  works with other data sources (arrays, files, etc.).

It is very easy to use the “TfrxDBDataSet” component. To connect it with the data source, you should set the “DataSet” property (which connects directly to a table or a query) or the “DataSource” property (which connects to a “TDataSource” component). Both ways of connection are equivalent, though the first one allows managing without the “TDataSource” component.

To make the component (and the data connected to it) available to the report, data sources used in the report must be clearly specified. To do that, select the “Report|Data...” menu item in the “FastReport” designer, and then select the required sources in the opened window.



2.9 “Customer List” report

Our second report will be much more complicated than the first one (it will contain DB table data, a list of clients of a firm). To perform this, let us use the demonstration database DBDEMOS, which is included in the Delphi distribution kit. Let us create a new project in Delphi. Put the “TTable” component on the form and set its properties:

```
DatabaseName = 'DBDEMOS'
```

```
TableName = 'Customer.db'
```

To make the table’s data available to FastReport, We add a “TfrxDBDataSet” component, and then set its property:

```
DataSet = Table1
```

Finally, add a “TfrxReport” (the basic component of FastReport) on the form, open the designer, and click the “New report” button, so that FastReport will automatically create a basic design with three bands (“Report title,” “Master data,” and “Page footer”). To make our table useable in FastReport, we must allow it to be used. To do this, select the “Report|Data...” menu item and select frxDBDataset1 (it is the only dataset in the list at the moment), then click ok. After the dialog window closes, the Dataset and the fields of the table to which it is connected become visible in the “Data” service pane window.

Now let’s create the report. First, add a “Text” object with the “List of clients” text to the “Report title” band. Next, we connect the “Master data” band to our dataset. This can done in one of three ways:

- double-click on the band;
- select the “Edit...” item in the band contextual menu;
- click on the “DataSet” property in the object inspector.

Now we will place four text objects (which will display a client's number, a customer name, phone and fax fields of the dataset) on the band. Let us do it in several different ways in order to demonstrate the features of the FastReport designer. Put the first "Text" object on the band and enter "[frxDBDataSet1."CustNo"]", into it. This is the most inconvenient way, since the link has to be entered manually, and there is a possibility of entering the text incorrectly. To insert the field links into the text object easier, we can use the expression designer (its button is located in the toolbar of the "Text" object's editor window click it and the data dialog window will appear.). To insert our field, double-click on the required field in the dialogue. Then click the "OK" button, to close the dialogue and see the field inserted into the text.

The second way of adding the DB field into the text object is by setting properties with the object inspector. Put a second "Text" object to the band, without writing anything in the editor. We'll set it's properties using the inspector:

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

Since both of the properties are presented as a list, we only need to select the desired value using the mouse.

The third way is to "drag and drop" the required field from the "Data" service window into the report. It is the simplest and easiest way. Before doing this we should disable the "Create header" flag at the bottom of the "Data" window (otherwise we will create a second "Text" object, containing the field title, in addition to the desired field). Using the mouse select the "Phone" field, and then drag it to the band.

The fourth way. Requires the designer option ("View|Options...", "Show drop-down list of fields" flag be set and the band connected to dataset.). Place a blank "Text" object on the band, and then move the cursor to the object. In the right part of the object you will see the image of a button with the down arrow (as in opening lists). This is the DB fields' opening list. Click the button and select the "FAX" field in the list.



Our report design is finished:

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

Click on the “Preview” button to see the result.

Customer list			
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

2.10 Displaying DB fields with the help of the “Text” object

As you can see, the “Text” object is able to display data from DB, in addition to displaying static text and expressions. Moreover, we can do it in two ways: by either placing a link to the DB field into the object text, or connecting an object to the required field with the help of the “DataSet” and “DataField” properties. The first way is used when we want to display both field contents and any explanatory statement in the same object. For example:

Contact person: [frxDBDataSet1."Contact_Person"]

As you can see, special syntax is used for links to the DB field: datasetname."fieldname". The field name (as well as the set name) can contain spaces. Space between the "point" and "quote" symbols is not permitted.

Not only can a link to a field can be placed in the text of the object. We can apply different computing operations to a field as well:

*Length (cm): [<frxDBDataSet1."Length_in"> * 2.54]*

Note how square and angle brackets have been used. Remember that square brackets are used by default for delimiting expressions, which are included in the object's text. In case of need, square brackets can be replaced by a pair of any other opening/closing sequences (see the "Displaying expression with the help of the "Text" object" section). Angular brackets are used inside expressions for marking out the FastReport variables or DB fields. To be logical, we should write

Contact person: [<frxDBDataSet1."Contact_Person">]

instead of

Contact person: [frxDBDataSet1."Contact_Person"]

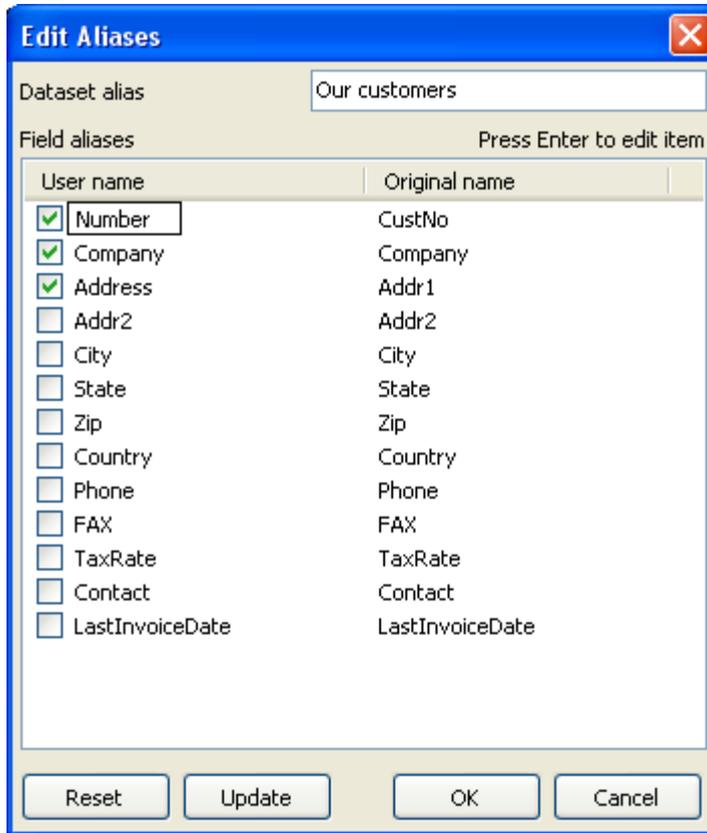
Nevertheless, both these notations are correct, since FastReport allows absence of angular brackets, in case when an expression contains only one variable or only one DB field. However, if an expression contains several members, the brackets are obligatory:

*Length (cm): [<frxDBDataSet1."Length_in"> * 2.54]*

2.11 Aliases

In the previous report, we used the data source with the frxDBDataSet1 name and the following fields: "CustNo," "Company," "Phone," and "FAX." Accordingly, we had to insert something like "[frxDBDataSet1."CustNo"]" into the report. Does it seem to be quite clear? Not really. One may want to rename the data source, and the field, naming it "Our clients" and "Number" respectively. However, "frxDBDataSet1" is a name of the component, which does not support spaces. And "CustNo" is a name of the field; it cannot be renamed directly (without database restructuring). There is however a way out. The user can use so-called pseudonyms or aliases in such situations. Both the data source and the field possess second names, i.e. aliases, which can easily be modified (the original names, of course, are not modifiable). If a name has an alias, this alias is what is used in FastReport. Otherwise, the original name is used.

It is very easy to rename a data source and its fields in FastReport. This is performed from Delphi environment. To open the alias editor, double-click on the `frxDBDataSet1` component. You can modify the data source name, names of its fields, and select the fields you would need in the report. Let us rename the source and fields (see picture):



Note, that an alias of the source can be modified without using the alias editor. To perform this, modify the "UserName" property of the `frxDBDataSet1` component.

Now we need to modify the report, as the names of the fields have been changed. To modify the names of fields in objects, it is easier to use the fourth way, which was described in the "List of clients Report" chapter. Move the mouse cursor to the "Text" object so that the button in the right part of the object would appear, click on the button, and select the desired field in the list. As you can see, now the data source name and its fields' names are more than understandable.

Note: It is better to assigning an alias in the very beginning, before designing a report. This will avoid subsequent need to rename fields in the report.

2.12 Variables

In addition to usage of aliases, there is one more way, which allows the user to set more understandable names for DB fields (and not only for them). One can associate a DB field name, as well as any expression, to a variable. To create and work with variables in FastReport, select the “Report|Variables...” menu item, and then click “Variables” in the toolbar.

The list of variables in FastReport has a two-level structure. The first level contains categories, and the second contains the variables themselves. Categorization of the variables is designed for convenience when a list of variables is too long. A list must contain at least one category. That means, that the variables cannot be located at the upper level. Furthermore, categories are needed for logical variables classification only, therefore, they are not included in reports. That is why, when setting a name for a variable, do not forget that it must be unique; it is impossible to create two identical variables in different categories.

Let us illustrate the use of variables by the following example. Assume we have two data sources: the first is “frxDBDataSet1” with the “CustNo” and “Name” fields and the second is “frxDBDataSet2” with the “OrderNo” and “Date” fields. We can associate the following list of variables to the fields:

Clients

Client number

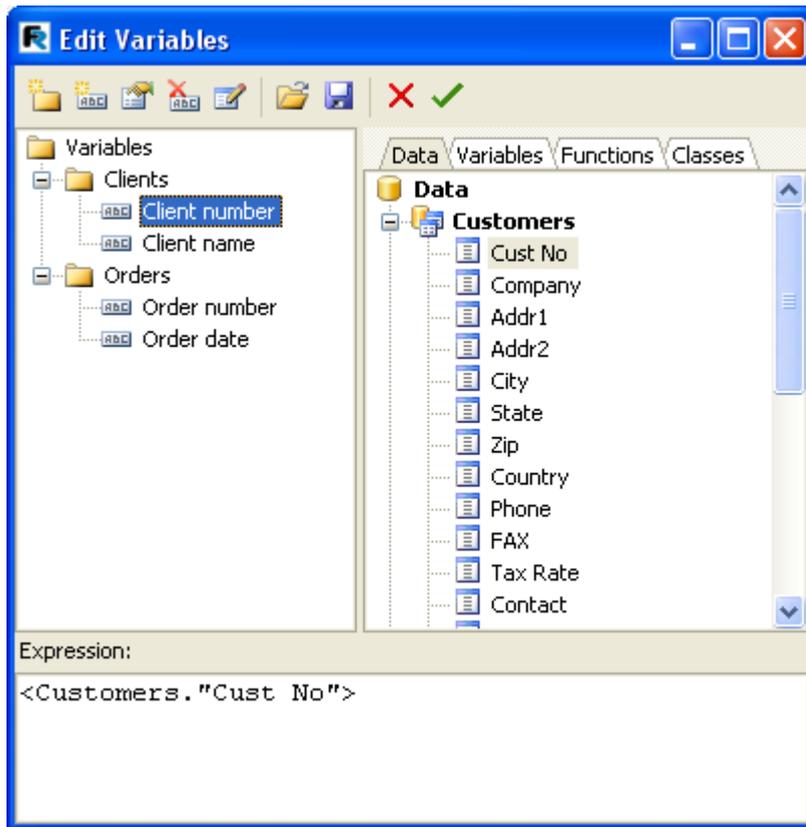
Client name

Orders

Order number

Order date

where “Clients” and “Orders” are two categories. Let us open the variables editor and create a required structure with the help of the “New category,” “New variable,” and “Edit” buttons. To associate the variables to the DB fields, let us select a variable and double-click on the required DB field in the right part of the window. The link to the DB field will be moved to the bottom pane of the window. The variable is now associated with the expression, so the value of variable becomes the value of the expression. If necessary, the expression can be edited or modified manually and any FR functions or other variables may be used within it. Remember that categories must not be associated to anything.



After the list of variables is created, close the variables editor. Now we can insert the variables into the report. In contrast to inserting DB fields, there are fewer variants here. We can either insert a variable into the object text manually by typing the “[Client number]” text, or drag a variable from the “Data” service window to the required place of the report. In the second case, it is required to switch to the “Variables” tab in this window.

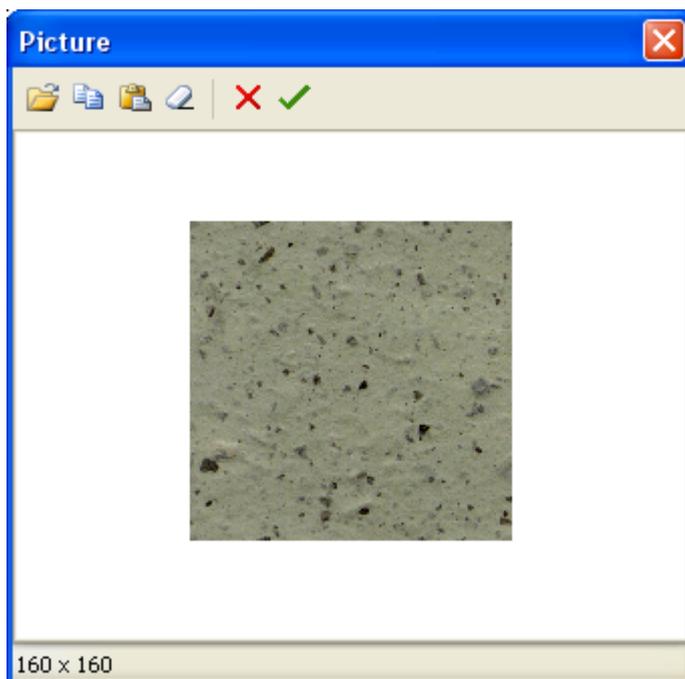
Let’s review what we have learned so far,

- A report design is composed of design pages.
- Pages may contain FR objects, either placed on the page or within a band.
- Bands are placeholders on the design page and depending upon their type, control where the objects they contain appear on the output page(s).
- Text Objects contain the text we want to output at a given position, they are multi-lined, and may contain static text, datafields, variables, expressions, or a combination of all.
- Data type bands, Master, Detail, Subdetail etc., when connected to an TfrxDBDataset control the number of times these bands appear (rows) and together with the report engine determine the number of finished pages output by a design page. Note: even though these data bands may have types like master detail etc., this is only a place relationship of the bands hierarchal position on the output pages(s). the actual data relationships are dependant on the table/query relationships to which the frxdbdatasets are connected. Each databand level requires an TfrxDBDataset or equivalent association..

2.13 “Picture” object

The next object to examine is the “Picture” object. It is also frequently used in reports. With the aid of this object, you can insert a logo, a photo of an employee or any other graphical information. The object is able to display graphics in “BMP,” “JPEG,” “ICO,” “WMF,” and “EMF” formats.

Let’s examine the capabilities of the object. Create a blank report and place a “Picture” object on the report page. in the object editor (if it does not open automatically, then doubleclick on the object). Load any desired picture and click “OK.” You can load a picture from a file or clear an existing picture



There are some options available in the object’s context menu, which correspond to names of the properties in the object inspector.

- AutoSize
- Stretch – enabled by default
- Center
- KeepAspectRatio – enabled by default

If the “AutoSize” option is enabled, the object will be resized, according to the size of the picture it contains. Sometimes such feature can be useful, if pictures of different sizes are to be displayed. This option is disabled by default, for convenience.

The “Stretch” option is enabled by default. This option stretches the picture inside the object. Modify the object’s size using the mouse and you will see, that the picture size always corresponds to the object’s size. If this option is disabled, the picture will be

displayed in its original size. This behaviour differs from the “AutoSize” option because the object dimensions are not adjusted according to the picture size, which means that the object can be larger or smaller than the picture it contains.

The “Center” option allows aligning a picture inside the object.

The “KeepAspectRatio” option is enabled by default it stops the picture from distorting when the object’s sizes are modified. This option works only together with the “Stretch” option. Therefore, when applying any object dimensions, a drawn circle will remain a circle, without turning into an oval. The stretched picture then occupies only the part of the internal space of the object needed to display it using correct ratios. If the option is disabled, a picture will be stretched to fill the object size, and if object’s size does not correspond to the initial dimensions of the picture, it will be distorted.

Another useful property available in the Object inspector only - "FileLink". You can put a filename (c:\picture.bmp) or variable containing a filename ([picture_file]) to this property. The picture will be loaded from given filename when you run a report.

2.14 Report with pictures

The “Picture” object, like many objects in FastReport, can display data from a DB. The connection of this object to a desired DB field is done by setting the “DataSet” and “DataField” properties in the object inspector. In contrast to the “Text” object, this is the only way to connect an object to data.

Let us demonstrate this with a report, which will have images of fishes, and their names. To do this, we will again need the “DBDEMOS” demo database, which is included in Delphi distribution kit.

We’ll create a blank project in Delphi, and then put the “TTable” component on the form and set its properties:

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

For working with the table from FastReport, add a “TfrxDBDataSet” component and set its properties:

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

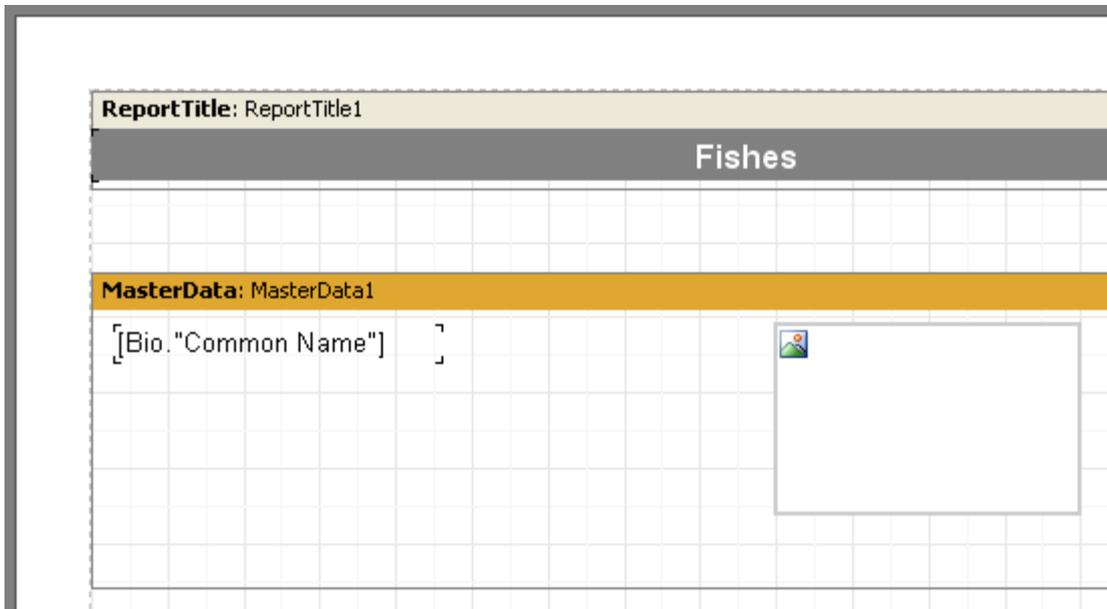
Finally, add a “TfrxReport” component on the form. Open the designer and click the “New report” button, so that FastReport will create a basic layout. Now we allow the frxDBDataset and it’s connected table to be used in the report. From the Menu select “Report|Data” select “Bio” dataset and click “Ok”.

Now we'll add objects to the report form. Place a "Text" object with the text "Fish" in the "ReportTitle" band. Connect the "Master Data" band to the data source (double-click on the band and select "Bio" from the list). Increase the band's height to 5 cm by dragging the bottom of the band down or use object inspector. Place a "Text" object in the band and connect it to the "CommonName" field using any of the methods described previously. After that, drop the "Picture" object alongside, and connect it to the "Graphic" field by setting its properties using the object inspector:

DataSet = Bio

DataField = 'Graphic'

Note, that both of these properties are of the "List" type, and one can select the required values using the mouse. To make room for the picture, stretch the object up to 4 x 2.5cm.



We are done. The report is finished (see the picture below):

Fishes	
Clown Triggerfish	
Red Emperor	

2.15 Multi-lined text displaying

We'll add to the previous example. In the "Biolife" table, there is a "Notes" field, which contains a detailed description of each fish. Update our report by adding this field into it.

At first glance, everything seems to be easy: add the "Text" object to the databand between the existing objects, connect it to the "Notes" field and set the object's size (8 x 2.5 cm). When previewing the report, you will see that the output is not exactly what we want:

Fishes		
Clown Triggerfish	Also known as the big spotted triggerfish. Inhabits outer reef areas and feeds upon crustaceans and mollusks by crushing them with powerful teeth. They are voracious eaters, and divers report seeing the clown triggerfish devour beds of pearl oysters.	
Red Emperor	Called seaperch in Australia. Inhabits the areas around lagoon coral reefs and sandy bottoms. 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	
Giant Maori Wrasse	This is the largest of all the wrasse. It is found in	

FastReport performed just what it was instructed to do. The "Notes" field contains multi-lined text, whose size may vary. At the same time, the "Text" object, which displays

the information from this field, has a fixed size. That is why some lines appear to be cut off. What to do in this situation?

Of course, either size of the object could be increased, or font size can be reduced. However, this may lead to wasted space on the output page, due to the fact that some fishes have long descriptions, while others have short ones. In FastReport, there are properties which allow us to resolve this problem.

This concerns both the band's and object's ability to automatically adjust their height in order to create the necessary space of a given record (row). To perform this, we just need to enable the "Stretch" property of both the "band" and the "text" object. However, that is not all, because a "text" object with longer text should be able to stretch by itself, we'll need to set some of it's properties also.

The "text" object can automatically set its height and width in order to find room for it's contents. One can use the "AutoWidth" and "StretchMode" properties for this. The "AutoWidth" property allows the "text" object width to vary so that all the lines find room without division of words. This mode is useful when an object has a single text line and growing to the right will not effect other objects. The "Stretch" property lets the object's height to grow to accommodate the text, without changing the objects width.. This property has several modes and you can select one in the object inspector:

smDontStretch – do not stretch an object, by default;

smActualHeight – stretch an object in order to find room for the whole text;

smMaxHeight – stretch an object so that its bottom would coincide with the bottom of the band in which the object is placed.. We will examine this mode later.

Now we are interested in the "Stretch" property of the "Text" object. Enable it in the object context menu or set the "StretchMode = smActualHeight" property value. Also, enable the "Stretch" band property. Preview the report and make sure that everything works as expected.

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



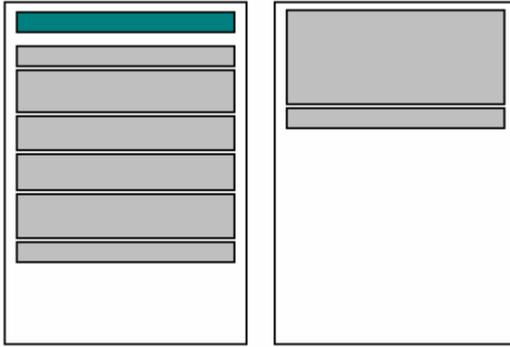
As you can see, when constructing a report, FastReport fills objects with data, stretches them with the "Stretch" option enabled, and then computes the band's height in order to find room for all the objects. If the band "Stretch" option is disabled, the height setting is not performed, and the band is displayed according to height specified in the designer. If we disable this option, we would see that the objects with longer texts are still stretched, although the band is not. This leads to text overlaying, since each next band is displayed right after the previous one.

2.16 Data splitting

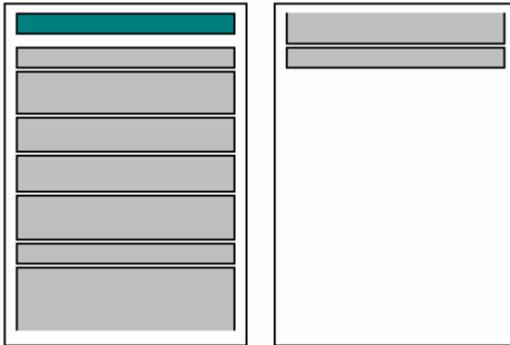
Let us pay attention to a peculiarity of this report: There is a lot of blank space at bottom of the pages. Why does it happen? When a report is constructed, the FastReport core fills whitespace of the page with bands. After displaying each band, the current position shifts down. When FastReport finds out that there is not enough space to display the next band (its height is larger than white space left on the page), then a new page is formed and band displaying continues there. This operation continues being performed as many times as there are records in the dataset.

Our report contains an object with large text, and that is why the band height is rather large. Furthermore, if a large band does not find room on a page, it is transferred to

the next one, and much unused space remains at the bottom of the page. This is shown at the following illustration:



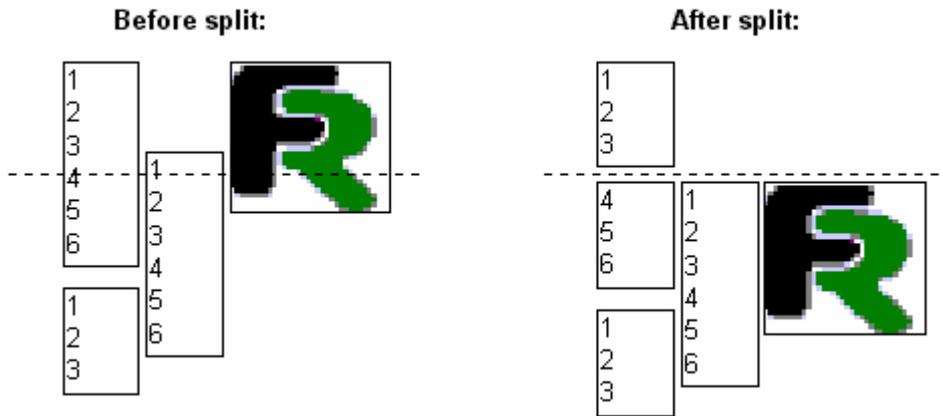
To limit paper wastage, let us use a FastReport feature, which paragraphs the band's contents. All we need do is to enable the "AllowSplit" option of the "First level data" band. You see that there is less white space at the bottom of report pages:



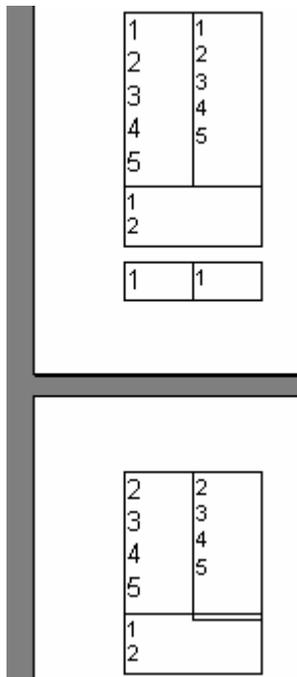
How does band splitting work? There are some objects in FastReport, which support this feature. They are the "Text," "Line," and "RichEdit" objects. They can be "split," while other objects cannot. When FastReport comes across the need to split a band, it performs it in the following way:

- displays the non-splittable objects, which find room on white space;
- partially displays splittable objects (text objects are displayed in a way that all lines find room in the object);
- forms a new page and continues object displaying;
- if a non-splittable object does not find room on whitespace, it is transferred to the next page; at the same time, all the objects located underneath, are shifted according to need;
- the process continues until all the band objects are fully displayed.

The splitting algorithm will become clearer if to look at the illustration:



It should be noted, that the splitting algorithm is not perfect and quality of the output report may not be as expected. You should use this option very carefully in cases when objects on the split band are grouped in a complicated way, and / or their font sizes differ. Here is the example of what could be received:

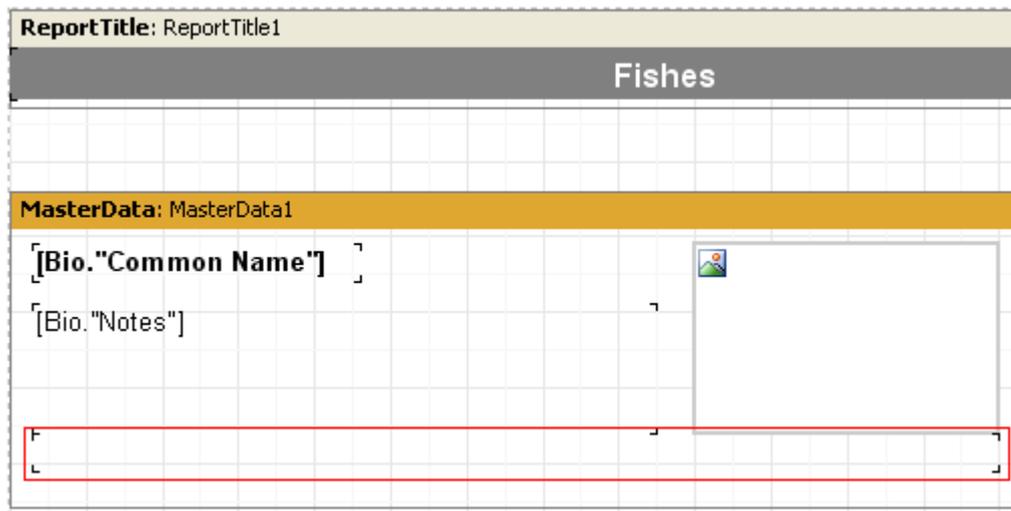


2.17 Text wrap of objects

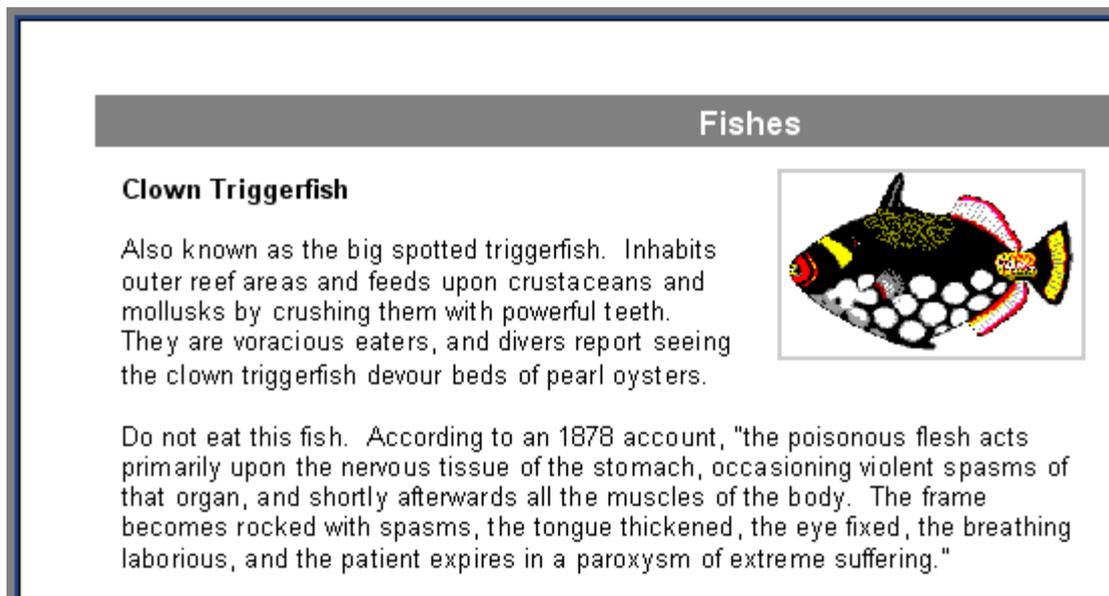
For report designing, in some cases it becomes necessary to wrap text around other objects (often, when using pictures). Let us demonstrate this FastReport feature with our current example.

Add one more “Text” object to databand below the “Bio.”Notes”” object, as

shown in the following illustration:



We will disable stretching for the “Bio.”Notes”” object. We will enable this property for the bottom object. To make the text “flow” from the “Bio.”Notes”” object to the bottom one, set the “FlowTo” property of the “Bio.”Notes”” object. This property is set in the object inspector and is of the “drop list” type. The bottom object’s name must be selected from this list. The result will look like the following illustration:



When constructing a report, if a text does not find room in the top object, the part, which does not fit the object, will be transferred to the bottom object. Since the objects are located around the picture, the effect of text wrapping is performed.

Attention: the main object should be inserted in the report before inserting the linked one. Otherwise, text wrapping may function incorrectly! If your report operates

incorrectly, select the linked object, and then transfer it to the forefront by the “Edit|Bring to front ” menu command.

2.18 Displaying data in the form of a table

Sometimes it is necessary to display a report in the form of a table with a frame. One of the examples of such a report might be a price list. To build such report in FastReport, a user just needs to enable framing function for the objects located in the “Data” band. Let us demonstrate several variants of frames with a test report example.

Let us create a report of the following kind:

MasterData: MasterData1		
[Bio."Specie"]	[Bio."Common Name"]	[Bio."Length"]

Place the “text” objects on the band side by side, and minimize band’s height.

The first and the simplest type of the table is a table with a full frame. To do this, one needs to enable all frame lines in every object:

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

The next type of framing would display only horizontal or only vertical lines. Such framing is performed in exactly the same way. Horizontal or vertical frames can be enabled in objects.

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

Finally, to show only the external framing, the report needs to be slightly modified:

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

As you can see, we have added two “Text” objects 1 in pageheader and 1 in pagefooter bands and enabled frame lines for the objects along the edges of the data-band. As a result, the report will look as follows:

90020	Clown Triggerfish	50
90030	Red Emperor	60
90050	Giant Maori Wrasse	229
90070	Blue Angelfish	30
90080	Lunartail Rockcod	80
90090	Firefish	38

All examples covered, contained bands, which had fixed sizes. But how is it possible to display a table, in a case where the band is stretched? Let us explain that, using the example below. Add a new field (a multi-lined text from “Bio.Notes”) to our report. As you already learned, the “Stretch” property must be enabled both for this object and for the band, in which the object is located. In this case, the band height is sized depending on size of the text in the “Text” object. Thus, we would output a report which appears like this:

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 Samoa.
90030	Red Emperor	60	Called seaperch in Australia. Inhabits the areas around lagoon coral reefs and sandy bottoms.

It is a little bit different from what we need; one would prefer the frames of the

neighboring objects to be able to stretch as well. FastReport allows one to solve this problem easily. For constructing such reports, it is enough to enable the “Stretch downwards” property (or StretchMode = smMaxHeight in the object inspector) for all objects, which are to be stretched. Thus, the FastReport core first calculates the maximum band height, then it “stretches” objects with the enabled option to the bottom band edge. Due to the fact that object frames stretches together with the object, the report’s appearance changes:

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

2.19 Printing labels

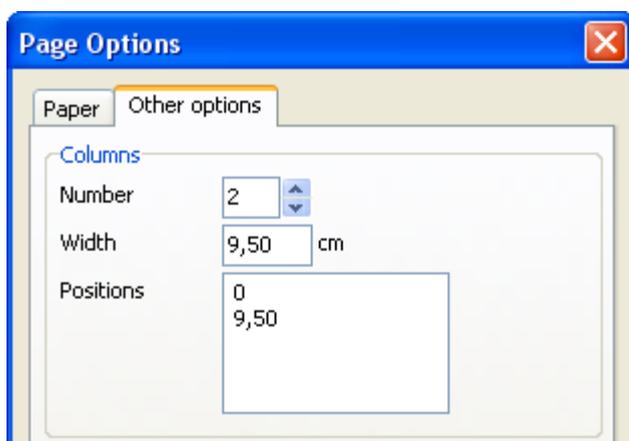
In contrast to table reports, data in reports such as “label,” are allocated one under another. Let us examine an example of such report, which displays data about fishes (see the previous example). The report is presented in the form of a label, and has the following structure:

MasterData: MasterData1	
Number:	[Bio."Specie"]
Category:	[Bio."Category"]
Name:	[Bio."Common Name"]
Length, cm:	[Bio."Length"]

When previewing this report, we would see the following output:

Number:	90020
Category:	Triggerfish
Name:	Clown Triggerfish
Length, cm:	50
Number:	90030
Category:	Snapper
Name:	Red Emperor
Length, cm:	60

Notice you can see, there is a lot of whitespace in the right part of the page. To fill the whole page, the number of columns, where the data will be displayed, can be set in report page settings. To perform this, you should either double-click on the area of white space on the page, or use the menu's "File Page|Options..." menu item.



In this pagetab, one can set such column parameters, as number of columns, its width, and position. In our case, it would be enough to specify a number = 2, since FastReport adjusts all the other parameters automatically. The column frame is displayed in the designer as a thin vertical line:

MasterData: MasterData1		Bio
Number:	[Bio."Specie"]	
Category:	[Bio."Category"]	
Name:	[Bio."Common Name"]	
Length, cm:	[Bio."Length"]	

Printing will now be performed in the following way. FastReport will display the "First level data" band as long as there is white space on the page. After that, a new column in the same page will be formed (in contrast to simple reports, in which a new page is created), and band would continue to be displayed on the top. However, now all the

objects are shifted to the right, according to column's width. It will continue until all the columns are displayed. After that, FastReport forms a new page and continues to display data from the first column.

Our report with two columns should appear as below:

Number:	90020	Number:	90140
Category:	Triggerfish	Category:	Cod
Name:	Clown Triggerfish	Name:	Lingcod
Length, cm:	50	Length, cm:	150
Number:	90030	Number:	90150
Category:	Snapper	Category:	Sculpin
Name:	Red Emperor	Name:	Cabezon
Length, cm:	60	Length, cm:	99

The “Columns” property, available in all data-bands, is another way to set number of columns. It allows to set number of columns for a particular band and not for the whole page (as it was in previous example). Thus, data will display from left to right, then from top to bottom, instead of, from the top to bottom, then from the left to the right.”

We'll disable columns in the page (set the columns number = 1) and enter “2” in the band 's “Columns” property. FastReport displays the column frames as dotted lines. We can modify column dimensions by setting the “ColumnWidth”, “Column Gap” properties:

MasterData: MasterData1		Bio
Number:	[Bio."Specie"]	
Category:	[Bio."Category"]	
Name:	[Bio."Common Name"]	
Length, cm:	[Bio."Length"]	

A report constructed in such a way, will differ from the previous one only by displaying the data “from-left-to-right, then from-top-to-bottom” order.

2.20 Child-bands

Let us examine the case when one of the lines in a report of “label” type, may have a variable size. To simulate the situation using our example, let us reduce the “Bio.”Common Name”” object width to 2.5 cm, and enable the “Stretch” option for it. Let us also enable stretching in the “First level data” band. Enable all the frame lines in all objects so that the principle of the stretching function will become clear. This will output a report with the following appearance:

Number:	90020
Category:	Triggerfish
Name:	Clown
Length, cm:	Triggerfish
	50
Number:	90030
Category:	Snapper
Name:	Red Emperor
Length, cm:	60

You see, that in the first case the first object contains longer text, and this is why it was stretched into two lines. Thus, the object (located underneath it and linked to the Bio."Length (cm)" field) was shifted down. This happens because all the objects have the "Shift" property enabled by default (ShiftMode = smAlways in the object inspector). Such objects shift downwards if there is a stretchable object above them (a "Text" object with the "Stretch" property enabled). The height value, by which the object shifts, depends on how the object from above is stretched.

However, it is unacceptable in this case, since we need the object with the "Length, cm." text to be shifted as well. To perform this, there is a special band type in FastReport, the "Child-band." It is linked to (and is displayed after) it's parent band. Add a "Child" band to our reports design layout and move the 2 Text objects into it.

MasterData: MasterData1	
Number:	[Bio."Specie
Category:	[Bio."Category"]
Name:	[Bio."Comm
Child: Child1	
Length, cm:	[Bio."Length

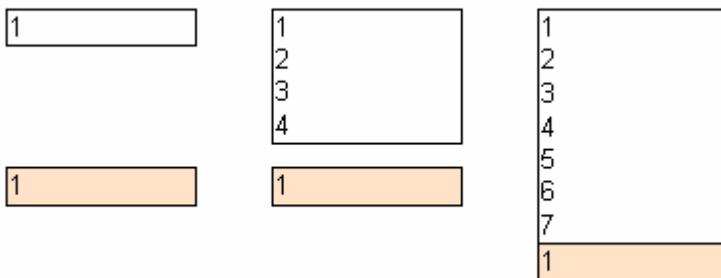
Link the Masterdata band to the Child band, by setting it's "Child" property in the object inspector. Now, each time the "Masterdata" band prints, the "Child" band will be printed immediately afterward:

Number:	90020
Category:	Triggerfish
Name:	Clown Triggerfish
Length, cm:	50
Number:	90030
Category:	Snapper
Name:	Red Emperor
Length, cm:	60

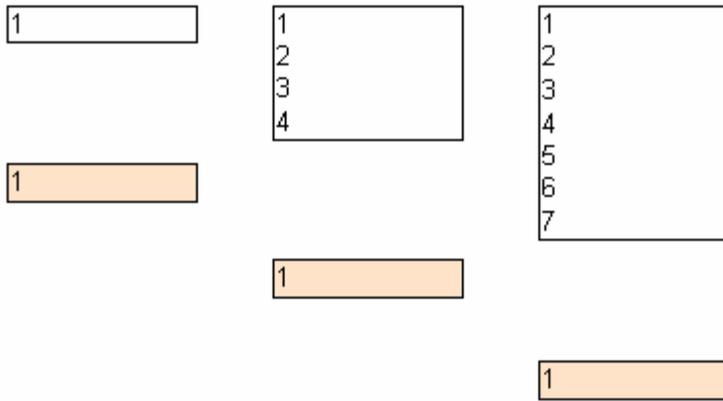
As you can see, now the title is typed exactly where it is supposed to be. In order to avoid child-band's transferring to the next page (which basically means, it will be widowed from the parent band), enable the "Keepchild" property for the parent band ("KeepChild" in the object inspector).

2.21 Shifting objects

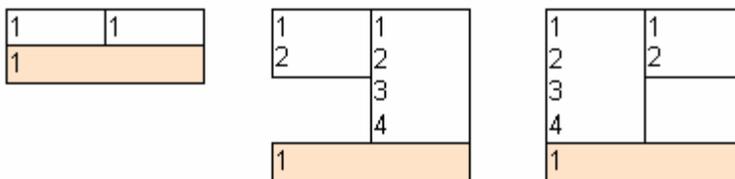
You have already seen how the "Shift" property works. Let us look at the next mode of shifting, "Shift on overlapping." In the object inspector, the "ShiftMode=smWhenOverlapped" property value corresponds to this mode. Thus, object shifting will be performed in case, when the object from above overlaps the given object during stretching. Three cases are shown in the illustration below. As you see, the bottom object with the enabled "Shift when overlapped" option shifts only in the latter case, i.e. when there is much text in the top object and it overlaps the bottom one.



If the "Shift" option is enabled, the bottom object will be shifted anyway:



This allows one to create rather complicated report output, by using the logic of the object's design properties, especially if an object overlaps several other objects at the same time. Thus, in the following example both of the upper objects contain stretchable texts, and the bottom one has the enabled "shifting when blocking" option. The bottom object will always be displayed closely to the object, which contains more text, irrespective of text size in the upper objects:



In this example, if the "Shift" option is enabled for this object, the bottom object will shift twice, since it is located underneath two objects and thus an unnecessary gap is formed.

2.22 Report with two data levels (master-detail)

So far our example reports have used only one data-band, ("First level data" or "Masterdata"), to control report output. This enabled output of data from one DB table. FastReport allows one to design report layouts having up to six data levels, on one design page. (It is possible to have more data levels by using the "subreport" object. This feature will be covered later). Generally, most reports are limited to 1-3 data levels with large numbers of data levels being rare.

Let us examine the two data level report design process. It will output data from the demo tables: "Customer" and "Orders." The first table is the list of clients; the second one is the list of orders placed by the clients. The tables contain data in the following fields:

Customer:

<u>CustNo</u>	<u>Company</u>
1221	Kauai Dive Shoppe
1231	Unisco
1351	Sight Diver
....	

Orders:

<u>OrderNo</u>	<u>CustNo</u>	<u>SaleDate</u>
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
....		

As you can see, the second table contains the list of all the orders placed by all companies. To view all of orders from table 2, placed by a company, selected from table 1, the tables are linked on the “CustNo” field, which is common to both tables. The report output from such data should appear as follows:

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

Let's design the report. Create a new project in Delphi, put two “TTable,” one “TDataSource”, two “TfrxDBDataSet” and one “TfrxReport” components on the form. Set the components properties as shown below:

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'

In the report designer, we'll connect our data sources in the "Report|Data..." window. Now add a "Master data" and "Detail data" bands on the page:

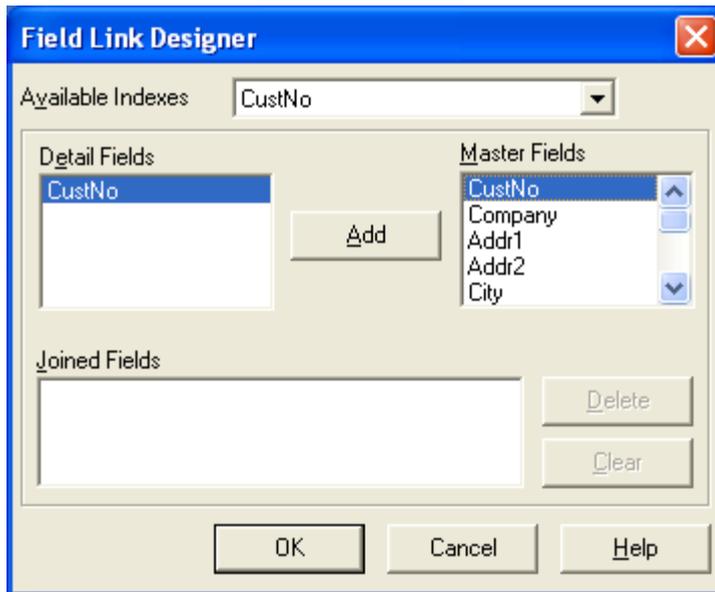
The screenshot shows two data bands in a report designer. The top band is labeled "MasterData: MasterData1" and is associated with the "Customers" data source. It contains two fields: "[Customers. 'Cust No']" and "[Customers. 'Company']". The bottom band is labeled "DetailData: DetailData1" and is associated with the "Orders" data source. It contains three fields: "[Orders. 'Cust No']", "[Orders. 'Order No']", and "[Orders. 'Sale Date']".

MasterData: MasterData1		Customers
[Customers. "Cust No"]	[Customers. "Company"]	

DetailData: DetailData1			Orders
[Orders. "Cust No"]	[Orders. "Order No"]	[Orders. "Sale Date"]	

Note that the "Master Data" band must be placed above the "Detail Data" band! If placed under it, FastReport will generate an error message when you preview the report.

If you previewed the report now, you would see that the list of orders remains the same for every customer and contains all records from the "Orders" table. This happens because we did not set the mastersource property of the "Orders" table. Set the "MasterSource = DataSource1" property in the "Table2" component on the Delphi form. Now, we have set a "master-detail" connection. After that, we select the fields to link on. Set the "MasterFields" property of the "Table2" component:



We need to link together two “CustNo” fields in both sources. To perform this, select the desired fields, and then click on “Add” button. Fields linkage will appear in the bottom pane. After that, close the editor by clicking “ .”

When previewing a report, FastReport does the following. After outputting a record from the master table (Customer), it will set the filter on the detail table (Orders). Only those records, which match the “Orders.CustNo = Customer.CustNo” condition will remain in the table. That means that for each customer only those orders which belong to the current customer will be displayed in the detail band. Note this is an important concept to grasp. Even though databands may be of master or detail type, they only control where the data appears on the output page (order and no of times of appearance). What data their objects display is dependant upon which fields the objects contain and the external tables relationship linkage.

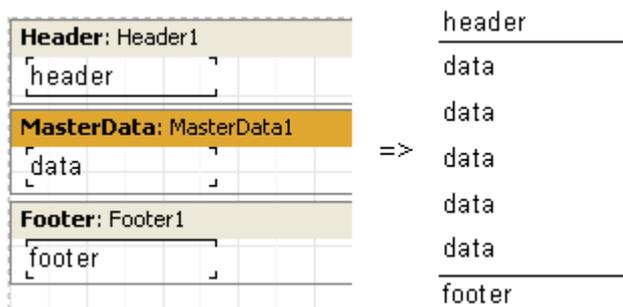
The illustration below shows the new output appearance.

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

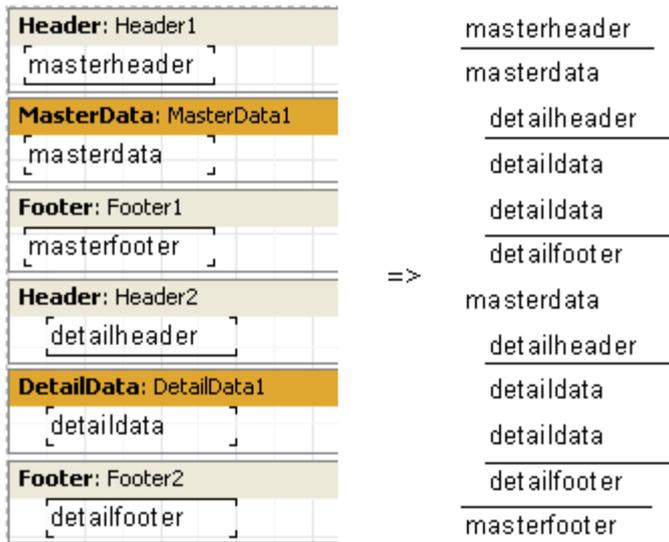
Reports, containing up to 6 data levels can be constructed in the similar way.

2.23 Headers and footers of a data band

Each data band may have header and footer. Headers are output before a data band, footers are output after all data records are output. Here is an example of how the headers/footers working in case of simple report:



Let's look at more complex example using two data levels - master-detail:



As we can see the header is output before all data-band records. Thus master data header is printed once at the beginning of a report, detail data header is output before each group of detail bands belonging to the current master record band. The detail footer is output after the group of detail bands belonging to the master record band, the Master footer is **not output** until after all the master databand records.

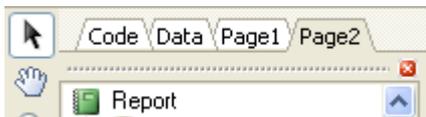
Using the FooterAfterEach property of the data band, we may override this behavior. Setting this property to True (you may also use context menu of the data band, "Footer after each row" item) will cause the footer to output after each data row. It may be useful in some cases when designing master-detail reports. The previous example with FooterAfterEach property of the master data set to True will appear this way:



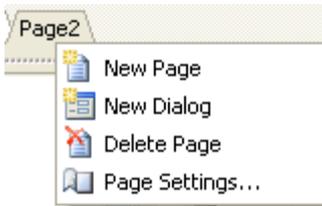
2.24 Multipage reports

FastReport reports can consist of several design pages. You can adjust such parameters as size and orientation for each page, as well as to place different objects and bands on it. When outputting this type of report, all bands from the first design page will be displayed, then the bands from the second one, etc.

When a user creates a new report in the designer, it already contains one page by default. You can add a new page by clicking on the  button in the toolbar or by selecting the “File|New page” menu command. Then you would see that a new page tab appears in the designer:



One can easily switch between pages by clicking on the required page tab. Page tabs can be dragged (“drag&drop”), to easily modify their order. An unnecessary page can be deleted using the  button in the toolbar or by selecting the “Edit|Delete page” menu command. One can also call the context menu by right-clicking on the page tab:



The number of design pages in a report is unlimited. As a rule, additional pages are used either for previewing title pages, or in more complicated reports, those which contain data from many sources.

A simple example of title page creation. Let us use our previous report with one data level. Add a new page to it, this will be the second page. To move it to the front of the report, grab the page tab using the mouse, and then drag it to the place near the first page. At that, the pages order will be changed. Switch to the new page and place a “Text” object (with the “Our report” text inside) in the middle of the page. That is all we need to do. The report with a title page is finished:

Our report		Customers			
Company	Address	Contact	Phone	Fax	
Action Club	PO Box 9401-P	Michael Spurling	813-870-0280	813-870-0282	
Action Diver Supply	Blue Spar Box #3	Marianna Miles	22-44-800-211	22-44-800-896	
Adventures Undersea	PO Box 744	Gloria Gonzalez	011-34-00-054	011-34-00-054	
American SCUBA Supply	1730 Atlantic Avenue	Lynn Cincipini	213-854-0002	213-854-0006	
Aquatic Cinema	821 Everglades Way	Gillian Owen	813-442-7694	813-442-7676	
Blue Ocean Hobbies	8342 W. Shure Lane	Christine Taylor	213-855-1854	213-855-1865	
Blue Jack Aqua Center	23-718 Paddington Lane	Ernest Benet	401-800-7823	401-800-9403	
Blue Sports	203 12th Ave. Box 748	Theresa Kunic	610-772-8704	610-772-8808	
Blue Sports Club	83395 Nic Plaza Street	Henry Bathzoni	612-897-0342	612-897-0348	
Catamaran Dive Club	Box 264 Pigeon Point	Nicola Dupont	213-223-0941	213-223-2324	
Cayman Divers World Unlimited	PO Box 941	Jon Bailey	011-8-897-044	011-8-897-054	
Central Underwater Supplies	PO Box 787	Maria Ewertosh	21-11-443-246	21-11-443-2329	
Dave Jones' Locker	248 South 18th Place	Tanya Wagner	803-600-0112	803-600-0893	
Dive's Gratta	34801 Universal Lane	Peter Owen	213-432-0003	213-432-4821	
Divine of Blue-green	834 Complex Ave.	Nancy Beale	205-895-7184	205-895-6089	
Divine of Cortia, Inc.	Memrose Plaza 54	Charles Lopez	30-851-88394	30-851-88943	
Divine of Venice	220 Elm Street	Simona Grasin	813-443-2388	813-443-8942	
Divine-Tan-His	G.D. P Box 811	Jon Hester	879-804876	879-859348	
Flamingo Aquatics	222 680 420-77 A.A.	Bruce Wong	887-1-773-434	887-1-773-421	
Fisherman's Eye	PO Box 7642	Bethan Lewis	800-895-4880	800-895-4880	
Frank's Divers Supply	1485 North 44th St.	Lloyd Falouts	803-895-2778	803-895-2789	
George Bain & Co.	#73 King Salmon Way	Bill Wynn	803-438-2771	803-438-3003	
Gulf Coast Supply	223-B Houston Place	Elaine Falls	208-895-2940	208-895-4094	
Island Finders	8133 1/2 Stone Avenue	Diamond Ortega	713-423-8776	713-423-8776	
Jamaica SCUBA Centre	PO Box 68	Barbara Hervey	011-8-897-045	011-8-897-043	
JamaicaBun, Inc.	PO Box 643	Jonathan West	808-895-2746	808-895-0203	
Kauai Dive Shoppa	4-676 Sugarloaf Hwy	Erica Norman	808-895-0289	808-895-0278	
Kirk Enterprises	42 Aqua Lane	Rudolph Claus	713-895-4357	713-895-1073	
Larry's Diving School	3982 NW Buca Street	Isabel A Nascia	803-403-7777	803-403-0689	
Makal SCUBA Club	PO Box 8834	Donna Sikus	317-849-0088	317-849-8787	
Melina SCUBA Center	PO Box 82482 Zulu 7831	Stephan Bryant	88-33-891222	88-33-891049	
Mermoz Divers Club	872 Ocean St.	Joyce Marsh	418-895-0369	418-895-0369	
Neptune's Trident Supply	PO Box 103	Louise Parks	778-387-6146	778-387-6146	
Newsstar SCUBA Limited	PO Box 8834	Angela Jones	778-123-0746	778-123-0705	
Ocean Adventures	PO Box 498 Kihai	Paul Still	778-898-0334	778-898-0883	
Ocean Paradise	PO Box 8748	Paul Gardner	808-895-8231	808-895-8480	
On-Target SCUBA	7-797 83 Naniakona Road	Brian Phillips	418-448-0288	418-448-0223	
Princess Island SCUBA	PO Box 32 Whakaroa	Anna Madachl	879-3115233	879-3111203	
Professional Divers, Ltd.	4104 Avenida St.	Gregory Williams	202-895-8133	202-895-1694	
Safari Under the Sea	PO Box 7686	Anna Rack	800-400-4223	800-400-3002	
San Pedro Dive Center	1701-D N Broadway	Patricia O'Brien	823-444-2910	823-444-2660	
SCUBA Heaven	PO Box O-8874	Robert Michalind	011-32-00-488	011-32-00-488	
Shang-La Sports Center	PO Box D-5485	Frank Panagou	011-32-08874	011-32-44638	
Sight Diver	1 Neptune Lane	Phyllis Spozner	387-6-878708	387-6-878704	
The Daphn Charge	8243 Underwater Hwy.	Sam Whitherspoon	800-895-3768	800-895-1283	
The Diving Company	PO Box 8835	Brian Miles	22-44-801088	22-44-800788	
Tom Sawyer Diving Centre	852-1 Third Frydenhøj	Chris Thomas	804-768-3022	804-768-7772	
Tony Tony Tony	PO Box H-4873	Kevin Ritar	800-898-0043	800-898-9884	
Underwater Fantasy	PO Box 942	Gwnt Answorth	800-895-2234	800-895-2234	

It is necessary to pay attention to one feature of multipage reports. If the “Print to previous page” option is enabled in the second page (use the “PrintToPreviousPage” property in the object inspector), then the second page objects will start printing not on a new output page, but on the white space of the previous one. This allows to print the pages’ contents “line-to-line.”

Chapter



Groups, aggregates

3.1 Report with groups

In the previous example we constructed a two-leveled report based on the data from two tables. FastReport allows designing of reports which appear the same, based on one set of data, the result of a joined query.

To perform this, one needs to create a query using SQL language, which would return data, arranged according to a certain condition, from both of the tables. In our case, a condition will be the value of the “CustNo” fields in both of the tables. An SQL-query may look as follows:

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

The "order by" line is needed to sort the records on the “CustNo” field. The example below shows how the query data would be returned:

<u>CustNo</u>	<u>Company</u>	...	<u>OrderNo</u>	<u>SaleDate</u>
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

How can a multi-leveled report be constructed on the basis of this data? In FastReport there is a special band – “Group Header”. A special condition is established for the band (DB field value or an expression); the band is output every time the field's value is changed. The following example illustrates this.

Let us create a new project in Delphi, put the “TQuery,” “TfrxReport,” and “TfrxDBDataSet” components on the form. Set their properties as shown below:

Query1:

DatabaseName = 'DBDEMOS'

SQL =

*select * from customer, orders*

where orders.CustNo = customer.CustNo

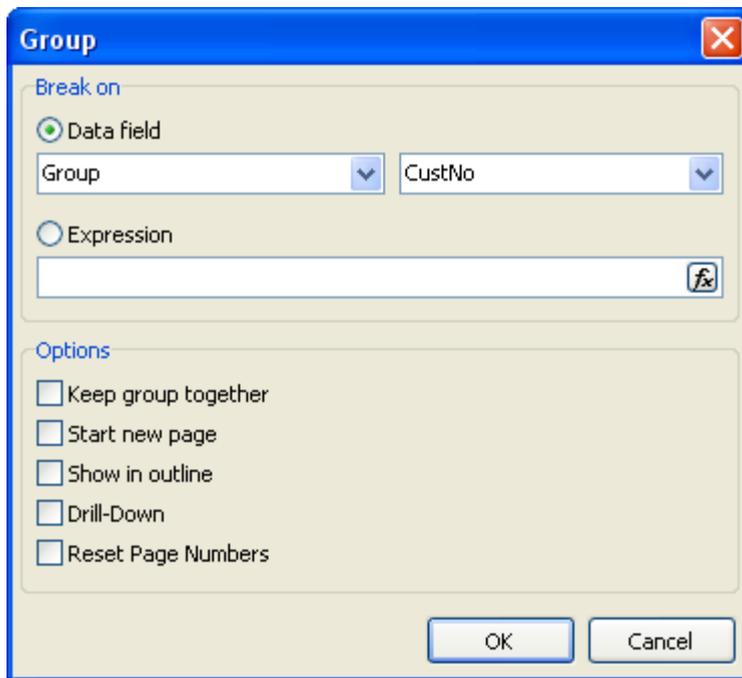
order by customer.CustNo

frxDBDataSet1:

DataSet = Query1

UserName = 'Group'

Let us open the designer and connect our data source to the report. After that, add the “Group header” and “Master data” bands to the report. Set a condition (in this case, it is “Group.CustNo” data field) in the “Group header” band editor:



Let us link data-band to the “Group” data source and place the objects in the following way (note, that the group header must be placed above the data-band):

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

On previewing, we would see a report similar to the one shown below:

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

As you can see, the “Group header” band is output only when the field, to which it is associated, changes its value. Otherwise, the data-band connected to the group is displayed. If compare this report to the master-detail report, which we constructed earlier, it should be obvious that order numbers are not sorted in ascending order. This is easily corrected by changing the SQL query’s order by clause text:

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

Similarly, reports with nested groups can be constructed. The number of nested groups is unlimited in such reports. Reports using groups, have some advantages over reports of the master-detail type:

- the whole report needs only one dataset (query);
- the number of the data grouping levels is unlimited;
- the additional data sorting feature;
- more optimal usage of the DB resources (the query returns only one dataset, which can be output, without having to filter the data).

The only disadvantage is the need to write queries in SQL language. However, knowledge of SQL basics is obligatory for a programmer working with databases.

3.2 Other group features

Let's look at how the group is transferred to the next output page:

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

When looking at the printout of the report, it is unclear which client the list of orders on the very top of the second page refers to. FastReport allows repeating the group header output (which in our case contains information about the client) on the next page. To perform this, the “Reprint on new page” menu item (or the “ReprintOnNewPage” property in the object inspector) should be enabled in the “Group header” band. This will then display the following output:

1380	Blue Jack Aqua Center
1006	06.11.94

1380	Blue Jack Aqua Center
1079	03.05.89
1106	23.09.92
1153	16.04.94
1253	26.11.94

1384	VIP Divers Club
1007	01.05.88

There is another way, which allows one to avoid breaking of groups. This is the “Keep together” group header property (or “KeepTogether” context menu) It should be enabled. Then, if the whole group does not find room on the output page, it is transferred to a new page. In our example, it will appear as follows:

1356	Tom Sawyer Diving Centre
1005	20.04.88
1059	24.02.89
1072	11.04.89
1080	05.05.89
1105	21.07.92
1180	06.08.94
1266	15.12.94
1280	26.12.94
1305	20.01.95

1380	Blue Jack Aqua Center
1006	06.11.94
1079	03.05.89

Thus, much blank space may appear on several pages, but the group will be displayed as a whole on the page if possible.

In conclusion, the "StartNewPage" group header property allows output of each group on a separate page. This probably will lead to a waste of paper, however it might be useful in some cases.

3.3 Reset page numbers

Group has a "ResetPageNumbers" property which allows us to reset page numbers when printing a group. What is it for?

For example, you have created the report with groups. In the group header you have the customer name, inside the group - customer's orders. Now you need to print a report and send it to customers (each customer should get only its own pages of report). Unfortunately the page numbering in such report is continuous. Some customer may get the pages with numbers 50, 51, 52 (and where is the first 49 pages - he will ask?). To avoid such situation you have to give a number to each customer's page separately. Inside one report you will have pages with their own numbering for each group.

Pay attention to the following: if you set ResetPageNumbers property to True, you also should set the StartNewPage property to True. Thus each group will start a new page. To print a page number or total pages, you may use [Page], [TotalPages] system variables.

3.4 Drill-down groups

The group header has a property called DrillDown. If you set it to True, the group becomes interactive. This means you can click on the group header in the preview window. The group will expand (display all group records) or collapse (display only header and footer, if ShowFooterIfDrillDown is True).

Here is an example of such group with one expanded header:

Customers				
Company	Address	Contact	Phone	Fax
A				
B				
C				
Catamaran Dive Club	Box 264 Pleasure Point	Nicole Dupont	213-223-0941	213-223-2324
Cayman Divers World Unlimited	PO Box 541	Joe Bailey	011-5-697044	011-5-697064
Central Underwater Supplies	PO Box 737	Maria Eventosh	27-11-4432458	27-11-4433259
				Count: 3
D				
F				

You can control whether to display all groups collapsed or expanded when you run a report first time. By default all groups are collapsed. You can set `ExpandDrillDown` property to `True` if you want it expanded. You can also use the preview window's context menu to expand or collapse all groups.

3.5 Lines numbering

Let us use our example in order to show how to number lines in the group. To perform this, we add a "Text" object with a system variable `[Line]` to both of our bands (it is easier to perform this using the drag&drop method from the "Variables" page tab of the "Data Tree" tool window).

GroupHeader: GroupHeader1				
<code>[Line]</code>	<code>[Group."CustNo"]</code>	<code>[Group."Company"]</code>		
MasterData: MasterData1				
<code>[Line]</code>	<code>[Group."OrderNo"]</code>	<code>[Group."SaleDate"]</code>		

When previewing the report, we can see that both the data levels now have their line numbers:

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

For continuous numbering of the second level data, we use the “Line#” variable instead of “Line” in the text object on the data-band. The result will then appear as below:

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

3.6 Aggregate functions

In most cases, group reports need to display some summary result information (such as: “total of a group,” “number of group elements,” etc). FastReport provides these aggregate functions this purpose. With their use, one can retrieve the defined aggregate value over a data span. The list of aggregate functions:

SUM	Returns the total of the expression
MIN	Returns the minimal value of the expression
MAX	Returns the maximal value of the expression
AVG	Returns the average value of the expression

COUNT Returns the number of lines (rows) in the data span

The syntax of all aggregate functions (except COUNT) is as shown below, using an example of the “SUM” function):

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

The parameters assignment is as follows:

expression – the expression, the value of which will be handled

band – the name of data band, on which the value(s) to be handled originally reside

flags – a bit field, which can be the following values or their sum

1 – include invisible bands in calculation

2 – accumulate the value or running total(do not reset the result value when the current data span resets)

An expression is the only mandatory parameter; all the rest are optional. Nevertheless, it is recommended to always use band parameters, to avoid mistakes.

The “COUNT” function has the following syntax:

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

The parameters assignment is similar to the one described above.

There is a general rule for all aggregate functions: a function can be counted only for the data-band and used only in that band’s footer (the following bands refer to the latter: footer, page footer, group footer, column footer, and report footer(summary band)).

How do aggregate functions work? We will examine it using our group report example. Let us add some new elements to the report:

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

The Group."ItemsTotal" field on the data-band will display the current order total. Place a “Text” object in the group footer containing the aggregate SUM call, as shown

above,. It will display the total of all orders placed by the given client when previewed. Using a calculator, we can check to make sure that everything works as expected:

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

So, how do the aggregate functions work? Before outputting a report, FastReport scans the “Text” objects’ contents in order to find the aggregate functions. The functions found will be associated to the corresponding data-bands (in our example, the “SUM” function is associated with the “MasterData1” band). During outputting of a report (when the data-band is displayed) the value of the aggregate functions linked to it is counted up. In our case, the “Group.”ItemsTotal” field’s values are accumulated. After outputting a group footer (the one where the accumulated value of the aggregate function is displayed) the function value is reset, and the cycle is repeated for the next groups.

The purpose of the optional “Flags” parameter in the aggregate functions. In some reports, some of data-bands (or all of them) may be hidden, however, we may need to result value which considers all data-bands, visible or not. In our example, set the “Visible” property of the data-band to false; after that it will stop displaying. To count a total on the hidden data-band, we add the optional parameter to the function call:

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

It will give us a report, which would look as follows:

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

When the “Flag parameter” value = 2, the accumulated value will not be reset right after it is displayed. The result will become a “running total” on each successive output. Let’s modify the call of the function as shown below:

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

The “3” value is a bit combination of “1” and “2,” which means that we need to take into consideration the invisible bands without resetting the total. As a result, we have:

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

3.7 Page and report totals

Quite often, one needs to display summary total values of a page or a whole report. We can use the aggregate functions in this situation as well. We’ll show this by making some changes to our example.

GroupHeader: GroupHeader1		Group."CustNo"
[Group."CustNo"]	[Group."Company"]	
MasterData: MasterData1		Group
[Group."OrderNo"]	[Group."SaleDate"]	[Group."ItemsTotal"]
GroupFooter: GroupFooter1		
		[SUM(<Group."ItemsTotal">,MasterData1)]
ReportSummary: ReportSummary1		
		Total: [SUM(<Group."ItemsTotal">,MasterData1)]
PageFooter: PageFooter1		
		Total this page: [SUM(<Group."ItemsTotal">,MasterData1)]

As you can see, we added the “Report Summary” band and a “Text” object with the aggregate sum to the “Report Summary” and the “Page Footer” bands. That is all we need.

6812	Waterspout SCUBA Center	
1040	04.09.1988	3 632,00p.
1140	12.12.1993	1 240,00p.
		<hr/> 4 872,00p.
9841	Neptune's Trident Supply	
1149	14.03.1994	12 900,75p.
1045	16.10.1988	787,80p.
1049	13.12.1988	1 809,85p.
1145	17.01.1994	4 229,80p.
		<hr/> 19 728,20p.
		Total: 2922666,1
		<hr/>
		Total this page: 320872,8

3.8 Inserting aggregate function

So far we inserted the aggregate functions into the “Text” object manually. Now we will look at other ways to insert aggregate functions.

First of all, we can use the “System text” object for output of the aggregate function value. As a matter of fact, it is the same “Text” object, but one that has a special editor for easier insertion of system variables or aggregate functions.

System Memo

System variable

Aggregate value

Function: SUM

Data band: MasterData1

DataSet: Group

DataField: ItemsTotal

Expression:

Count invisible bands

Running total

SUM(<Group."ItemsTotal">,MasterData1)

Text

OK Cancel

You should step by step select a function type, a data-band (according to which it will be counted), and a DB field or an expression, value of which will be computed. You can also set the “Count invisible bands” and “Running totals” flags.

The second way is to use the “Text” object and the button in its editor, to invoke the additional window, similar to the “System text” object editor. When clicking the “OK” button, the call of the aggregate function is inserted into the object's text.

Chapter

IV

Formatting, highlight

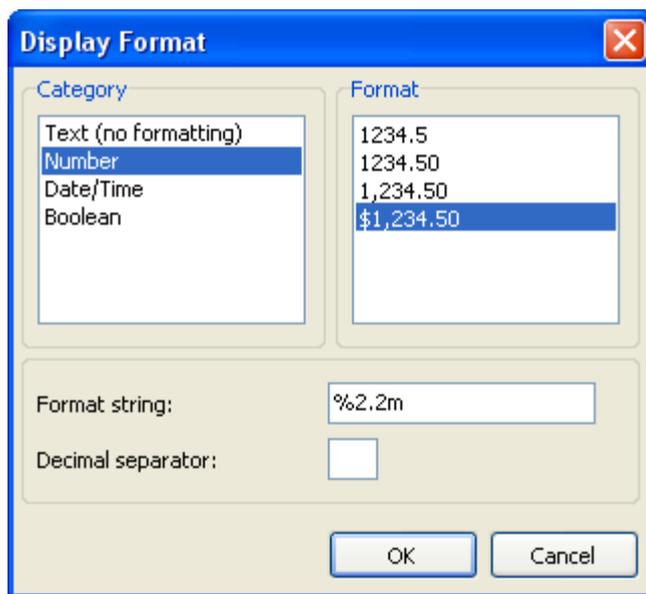
4.1 Values formatting

A peculiarity of the aggregate functions returned numerical values, is that they are not formatted. It should be evident, when referring to the first example with the “SUM” function:

1176	26.07.94	\$4 178,85
1269	16.12.94	\$1 400,00
		51450,8

This happens because, as a rule, the data fields return a formatted value, which is simply displayed by the “Text” object, with no changes applied. To apply formatting to the “SUM” function result, let us use the value formatting tools of FastReport.

Select the object with the sum and call its context menu. The format editor is called either by using the “Formatting...” menu command, or with the “DisplayFormat” property editor in the object inspector.



Here you can see, the list of formatting categories on the right and the list of the available category formats on the left. We’ll select the “Number” category, and the “\$1,234.50” format. At that, the format string corresponding to the selected format and the decimal separator character will be displayed. The format string is an argument of the Delphi “Format” function, which FastReport uses to accomplish formatting of numbers. You can modify a format string as well as a separator.

After clicking the “ ” button and report previewing, you will see that the sum total in the report is now formatted correctly:

1176	26.07.94	\$4 178,85
1269	16.12.94	\$1 400,00
<hr/>		\$51450,80

4.2 Inline formatting

In the example, formatting was applied to the object, and any expression located in the object. Everything worked correctly because there was only one expression in each object. However, if we have more than one expression and, they require different formatting, we can use inline formatting.

Using the example, resize the footer, it's object and the objects text as shown below.

Total: [SUM(<Group."ItemsTotal">,MasterData1)]
Number: [COUNT(MasterData1)]

The total and number of orders will display in the object.:

When previewing both of the values are presented in monetary format (which we had previously set), which is incorrect:

1269	16.12.94	\$1 400,00
<hr/>		Total: \$51 450,80
		Number: \$6,00

To display each value correctly, they should be formatted individually. To accomplish this, we use format tags. They are added before the closing square bracket of the expression. For our example, disable the object formatting (select the “Text (without formatting)” category in the format editor). Now we need to modify the format of the first variable, since the second will definitely be displayed correctly (without formatting, i.e. as an integer, and this is what we need). Modify the object text in the following way:

Sum: [SUM(<Group."ItemsTotal">,MasterData1) #n%2,2m]
Number: [COUNT(MasterData1)]

Preview, to make sure that the object displays correctly:

1269	16.12.94	\$1 400,00
		Total: \$51 450,80
		Number: 6

When using format tags, the general syntax is as follows:

[expression #tag]

Note that space between the expression and the “#” sign is mandatory! The tag itself might look as follows:

#nFormattingLine – the numerical format
 #dFormattingLine – date/time format
 #bFalse,True – boolean format

“FormattingLine” in every case is the argument to the function, by which formatting is accomplished. Thus, for numerical formatting, Delphi’s Format function is used, for date/time it is the FormatDateTime function. One can find the possible values from the Delphi help system. Below are several values used in FastReport:

for the numerical formatting:

%g – a number with the minimal signs number after decimal point
 %2.2f – a number with the fixed number of signs after decimal point
 %2.2n – a number with bits delimiter
 %2.2m – a monetary format, accepted in the Windows OS, depending on the regional settings in the control panel.

for the date/time format:

dd.mm.yyyy – date of the 23.12.2003 type
 dd mmm yyyy – date of the 23 Nov. 2003 type
 dd mmmm yyyy – date of the 23 November 2003 type
 hh:mm – time of the 23:12 type
 hh:mm:ss – time of the 23:12:00 type
 dd mmmm yyyy, hh:mm – time and date of the 23 November 2003, 23:12 type

It is acceptable to use a comma or dash instead of period in the line for the numerical format. This symbol will be used as a separator between the integer and the fractional parts of the value. Usage of other separators is not acceptable.

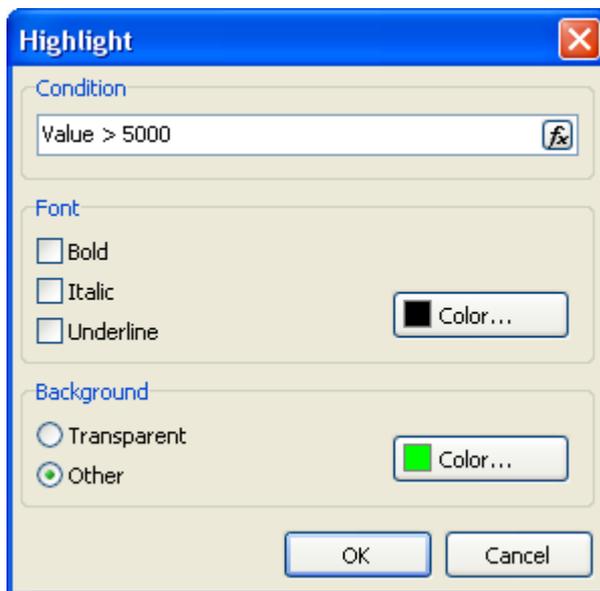
Formatting of the “#b” type (boolean), the formatting line is presented as two values separated by comma. The first value corresponds to “False,” the second one corresponds to “True.”

In order to avoid the necessity to memorize all these tags and their meanings, there is a convenient resource for formatting insertion in the “Text” object editor. When clicking the  button, the format editor (which we have already examined) is called. After

selecting a format, it is inserted to the text. Thus, if the cursor is placed before or after the closing square bracket, the format will be inserted correctly.

4.3 Conditional highlighting

This feature of the "Text" object allows one to color an object according to a specified condition. Any expression can be a condition. We'll use the example with groups to demonstrate. Let the order totals, which are larger than 5000, be green-colored. Select the object with the "<Group.ItemsTotal">" field and click on the "Conditional highlighting"  button in the designer toolbar. In the conditional highlight editor, enter a condition, which after the value is exceeded, the object will be highlighted, and specify the color attributes to change (font parameters and background color).



The preview result will appear as follows:

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

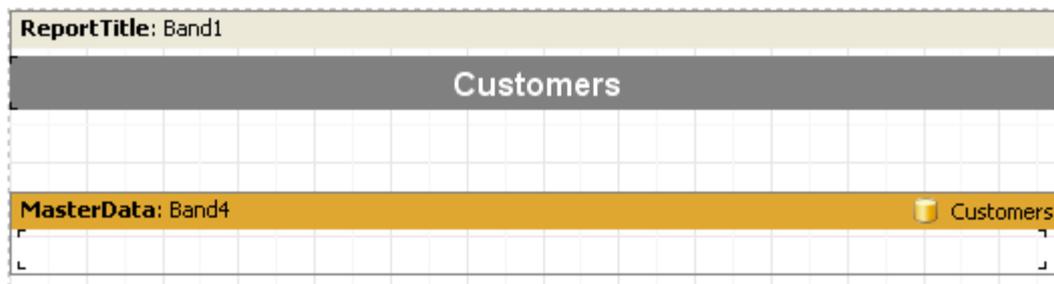
Note the conditional expression we specified (Value > 5000). Value is the DB field value, to which the object is linked. In a similar way, the "<<Group.ItemsTotal">> > 5000"

condition may be set. In general, any expression, which is correct in terms of FastReport, may be specified here.

4.4 Alternate color every other data row

Using conditional highlighting, it is easy to create a report having this appearance, by “coloring” every second data line. Using the “Customer List” report example which we constructed previously, to save effort.

Remove all the “text” objects from the “Master data” band. Put a “Text” object on the data-band and stretch it, so that it would occupy practically all the band space:



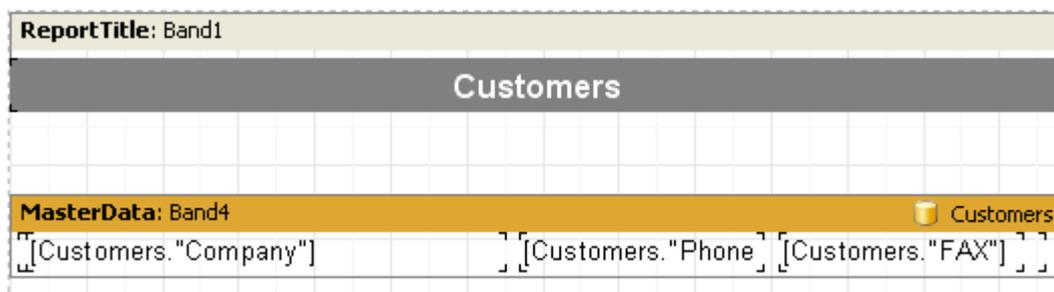
This object will modify its color depending on the data line number. Select the object and set the following conditional expression in the highlight editor:

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

Attention: if you have selected C++Script as a script language (see more details in the "Script" chapter), you should write the condition using C++Script:

```
<Line> % 2 == 1
```

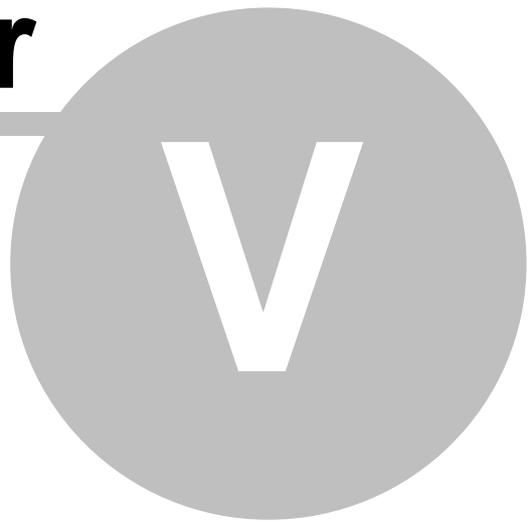
Select gray color for highlighting, but not too saturated (closer to white). Now other objects can be added to the data-band on top of the empty “text” object:



When previewing the report, we will can see the following output:

Customers		
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
Blue Jack Aqua Center	401-609-7623	401-609-9403
Blue Sports	610-772-6704	610-772-6898

Chapter



Nested reports (subreports)

5.1 Nested reports (subreports)

Sometimes it is required to display in a particular place, on a design page additional data, which may represent a separate report with a complicated structure. One may construct such report by using a set of FastReport bands, but it is not always possible. In such cases, the “Subreport” object is used .

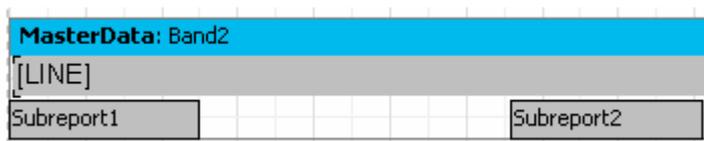
After inserting such an object to a report, we can see that FastReport automatically adds a new page, connected to this object. A nested report resembles a multipage one in terms of structure. The only difference is that the nested report is displayed in a specified location on the basic design page, and not after it. When outputting a report, as soon as the “Subreport” object is encountered, the report engine, outputs the associated design page, until it is complete. After that, basic design page output will continue.

One can also place a “Subreport” object in a Subreport design page, increasing the nesting level. An example of such a report can be found in the demo program, the “Subreports” report.

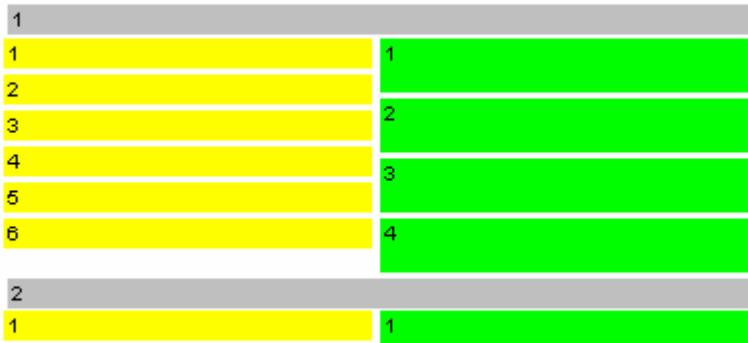
It should be noted that FastReport’s ability to construct subreports enables it to increase nesting levels of data. Remember that number of data levels in FastReport is limited to six when you do not use the “Subreport” object.

5.2 Side-by-side subreports

You can have two or more “Subreport” objects side by side on the same band:



This allows one to design reports, where the data output by each has different lengths (rows/records), or stretching or heights :

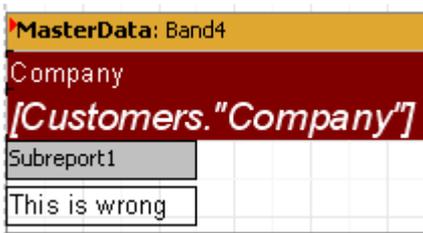


As you can see, FastReport continues to output the basic design page, after the longest Subreport is finished. One can also use the Vertical alignment property to adjust text object alignment within each subreport.

5.3 Limitations on using subreports

Since subreports are substituted on the basic design page, it **cannot** contain the following bands: “ReportTitle/ReportFooter,” “PageTitle/PageFooter/PageBackground,” and “ColumnTitle/ColumnFooter.” It is possible to put these bands on the nested report page, but they however will not be handled. For the same reason, there is no sense in modifying nested report pages options, inasmuch as the options of basic report’s page are used during outputting of a report.

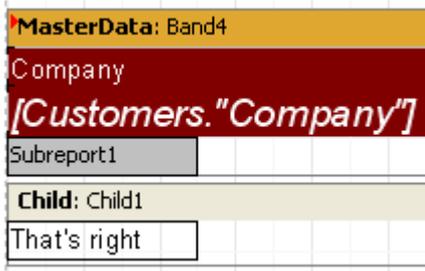
Do not put objects below the “Subreport” object:



When displaying a nested report, the nested report objects will overlay everything placed below, and the user will see something like this:



To display the objects below/after the nested report, use a child-band:

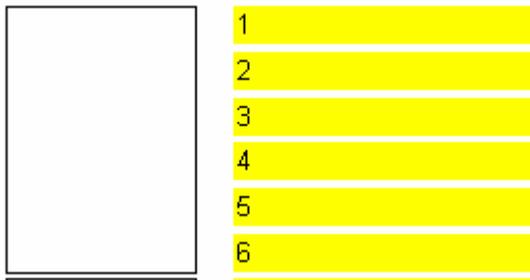


This method is also used when it is necessary to display several Subreports one under another, use a child band for each Subreport and chain them together. Child1's child property is set to child2 and so on.

5.4 PrintOnParent option

The "Subreport" object has the "PrintOnParent" property which can be useful in some cases. This property is False by default.

Usually a subreport is output as a set of bands on the basic report page. In this case the parent band (which contains a "subreport" object) do not depend on the subreport bands, i.e. can't stretch. If the "PrintOnParent" property is True (you can set it from the object inspector or in the context menu), subreport's objects are printed physically on the band which contains the "subreport" object. You can make this band stretched and put on it stretched objects:



Chapter



Script

Script is a program written in a higher-level language, which is a part of a report. As a report runs, the script runs as well. A script is able to perform data handling, which cannot be performed via regular means of the FastReport core, for example, to hide useless data according to any predefined condition. The script is also used for controlling properties of dialog forms, which are the components of the report.

The script should be written using one of the languages, which are the components of the script engine (FastScript). Currently, the following languages are supported:

- PascalScript
- C++Script
- BasicScript
- JScript

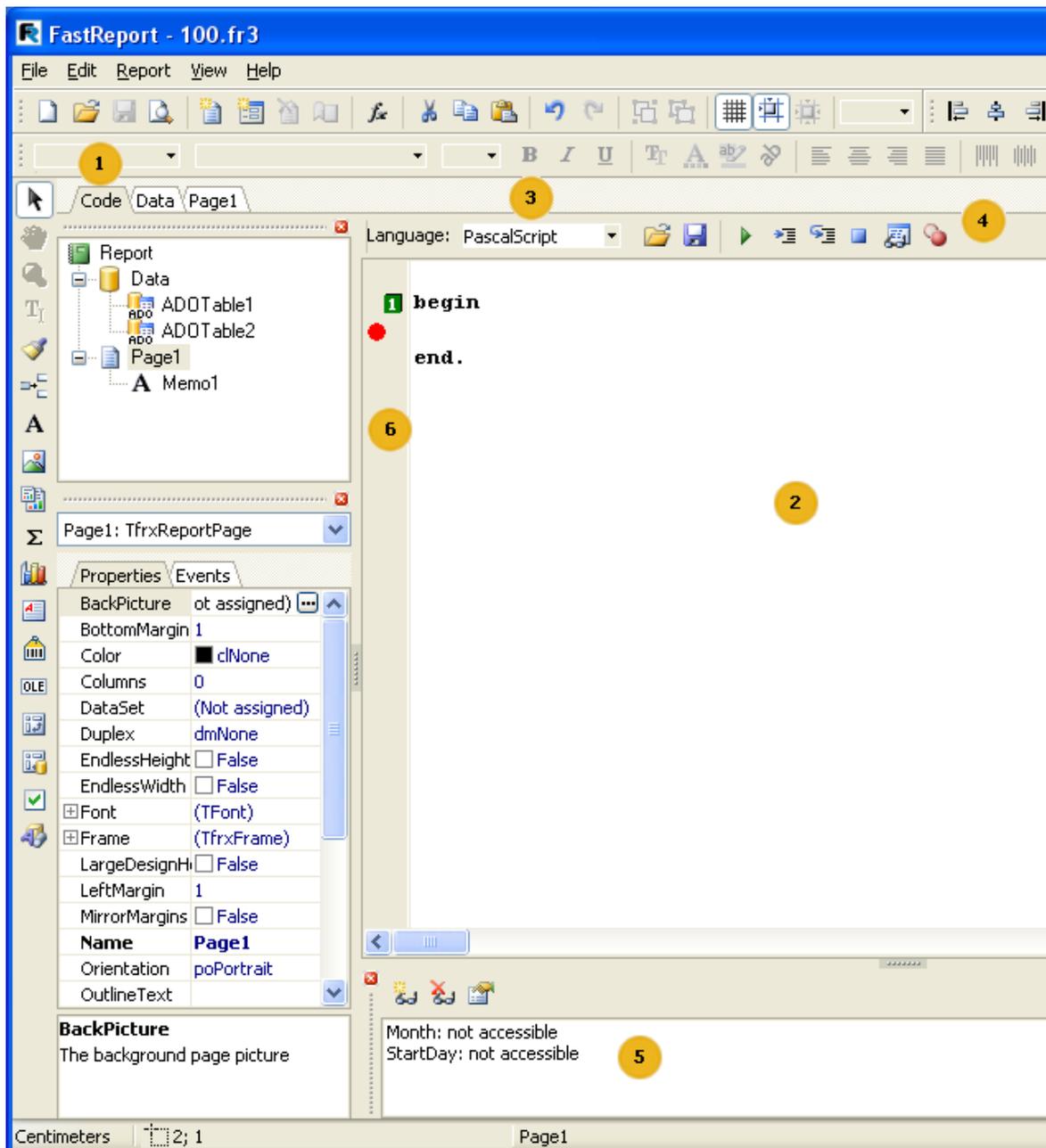
The following FastScript features available in the script engine:

- standard language set: variables, constants, procedures, functions (with nesting capability) with variables, constants, default parameters, all standard operators (including case, try, finally, except, with), types (integral, fractional, logical, character, line, multidimensional arrays, set, variant), classes (with methods, events, properties, indexes, and default properties);
- declarations of the following types absent: records, classes in the script; no records, no pointers, no sets (however, the 'IN' - "a in ['a'..'c','d']" operator usage is possible), no shortstring type, no unconditional jump (GOTO);
- types' compatibility checking;
- ability to access any report's object.

You can create scripts in the FastReport designer, which contains the scripts' editor with syntax's highlighting. Also there is an embedded debugger, which possesses the following functions: "Step," "Breakpoint," "Run to cursor," and "Evaluate."

6.1 Taste of script

Tools for working with the script are located in the "Code" tab of the FastReport editor. When switching to this tab, the designer appears as follows:



In the illustration above, the figures denote:

- 1 – “Code” tab;
- 2 – script’s editor window;
- 3 – a dropdown for selecting a language, in which the script is to be written;
- 4 – debugger’s toolbar:
 - ▶ - run report in debugging mode (F9);
 - ▶ - run to cursor (F4);
 - ▶ - perform the regular code line (Step into, F7);
 - ▶ - interrupt script’s work (Ctrl+F2);

 - preview expressions' evaluation (Evaluate, Ctrl+F7);

 - toggle breakpoint (F5).

5 - "Watches" window;

6 – bookmarks and breakpoints are displayed in this field; in addition, the lines, possessing the executable code are highlighted there.

Below there is the list of the keys, which can be used in the script editor.

Key	Meaning
Cursor arrows	Move the cursor
PageUp, PageDown	Go to the previous/next page
Ctrl+PageUp	Go to the beginning of the text
Ctrl+PageDown	Go to the end of the text
Home	Go to the beginning of the line
End	Go to the end of the line
Enter	Go to the next line
Delete	Delete the symbol at cursor's position; delete the selected text
Backspace	Delete the symbol to the left from the cursor
Ctrl+Y	Delete the current line
Ctrl+Z	Undo last action (up to 32 events)
Shift+Cursor arrows	Select a text block
Ctrl+A	Select the whole text
Ctrl+U	Shift the selected block by 2 symbols to the left
Ctrl+I	Shift the selected block by 2 symbols to the right
Ctrl+C, Ctrl+Insert	Copy the selected block to the clipboard
Ctrl+V, Shift+Insert	Paste the text from the clipboard
Ctrl+X, Shift+Delete	Cut the selected block to the clipboard
Ctrl+Shift+<number>	Set a bookmark with the 0..9 number on the current line
Ctrl+<number>	Jump to the set bookmark
Ctrl+F	Search a line
Ctrl+R	Replace a line
F3	Repeated search/replacement from the cursor's position
F4	Set the breakpoint and script's running (Run to cursor)

Ctrl+F2	Reset the program
Ctrl+F7	Preview variables' values (Evaluate)
F9	Run the script (Run)
F7 or F8	Execute code line (Step into)

6.2 Structure of a script

Script's structure depends on the language you use; however there are some common elements. They are the script's title, body, and the main procedure, which will be executed when the report runs. Below there are examples of the scripts for all four supported languages:

PascalScript's structure:

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

// the "uses" chapter should be located before any other chapter
uses 'unit1.pas', 'unit2.pas';

var // the "variables" chapter can be placed anywhere
  i, j: Integer;

const // "constants" chapter
  pi = 3.14159;

procedure p1; // procedures and functions
var
  i: Integer;

  procedure p2; // nested procedure
  begin
  end;

begin
end;

begin // main procedure.
end.
```

C++Script's structure:

```
#language ++Script // optional

// the "include" chapter should be placed before any other chapter
#include "unit1.cpp", "unit2.cpp"

int i, j = 0; // the "variables" chapter can be placed anywhere
```

```
#DEFINE pi = 3.14159 // "constants" chapter

void p1() // functions
{ // no nested procedures
}

{ // main procedure.
}
```

JScript's structure:

```
#language JScript // optionally

// the "import" chapter should be before any other chapter
import "unit1.js", "unit2.js"

var i, j = 0; // the "variables" chapter can be located
anywhere

function p1() // functions
{ //
} // main procedure.

p1();
for (i = 0; i < 10; i++) j++;
```

BasicScript's structure:

```
#language BasicScript ' optionally

' the "imports" chapter should be located before Any other chapter
imports "unit1.vb", "unit2.vb"

Dim i, j = 0 ' the "variables" chapter can be placed anywhere

Function p1() ' functions
{ '
} ' main procedure.

For i = 0 To 10
    p1()
Next
```

More detailed description of the FastScript script engine can be found in its documentation. The author did not duplicate the following in this manual:

- syntactic charts of all the supported languages;
- supported data types;
- operations with classes, properties, methods, and events;
- nested functions;
- enumerations and sets.

Later, we will examine examples of scripts written in “PascalScript” language. When a new report is created, this language is selected by default.

6.3 "Hello, World!" script

We have already examined an example of the "Hello, World!" report; now let us view, how to create a simple script, which would display a window with a greeting.

Enter the designer and click on the “New report” button for FastReport to automatically create a blank template. Switch to the "Code" page tab and write the following script:

PascalScript:

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

C++ Script:

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

After that, run the report. As we expected, FastReport displays a little window with a greeting:



Let us explain some details. We created a script consisting of a single “begin..end” block. Thus, our script has a very simple structure; it consists of a main procedure only (see the “Structure of a script” in this chapter). The main procedure is executed as soon as the report runs. In this case, it displayed a greeting window; the procedure ends right after the window is closed. After the main procedure is finished, report building starts.

6.4 Using objects in the script

One can address any report's object from the script. So, if there are, for example, the "Page1" page and the "Memo1" object, one can use them in the script, calling them by names, for example:

PascalScript:

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

C++Script:

```
Memo1.Color = clRed
```

The list of the report's objects available from the script is displayed in the "Report tree" service window. What objects' properties are available in the script? The answer is simple: those ones, which are visible in the object' inspector. At the same time, at the bottom of the inspector, there is a hint concerning the selected property. Both windows (report's tree and inspector) are available during working with the script. To get a detailed help about objects' properties and methods, use the FastReport help file, which is included in distribution kit.

A simple example. Put a "Text" object with the "MyTextObject" name and the "Test" text on the report's design page. Then write in the script:

PascalScript:

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

C++Script:

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

Run the report and see that our object's color became red.

6.5 Calling the variables from the report's variables list

One can call any variable, which is specified in the list of the report's variables ("Report|Variables..." menu item), from the script. Variable's name should be enclosed in angle brackets:

PascalScript:

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

C++ Script:

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

An alternative way is to use the “Get” function:

PascalScript:

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

C++ Script:

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

Modification of such variable’s value is available only via the “Set” procedure:

PascalScript:

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

C++ Script:

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

One should address the system variables, such as “Page#,” in exactly the same way:

PascalScript:

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

C++ Script:

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

6.6 Calling the DB fields

Just as with variables, one should use angle brackets for calling the DB fields:

PascalScript:

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

C++ Script:

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

And just as well, one can use the “Get” function (as a matter of fact, this function is always used in implicit way for calculating expressions, enclosed in angle brackets).

6.7 Using aggregate functions in the script

An idiosyncrasy of an aggregate function is that it should be used inside the “Text” object; one can call it in the script after it is used. If an aggregate function is used in the script only, (without using it in the “Text” object), an error message will appear. That happens due to the fact that an aggregate function must be connected with a definite band, and only then will it work correctly.

6.8 Displaying the variable’s value in a report

To display the contents of any script variable in a report, one should describe this variable and bind a value to it. Here is a simple example of a script variable:

PascalScript:

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

C++ Script:

```
string MyVariable;

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

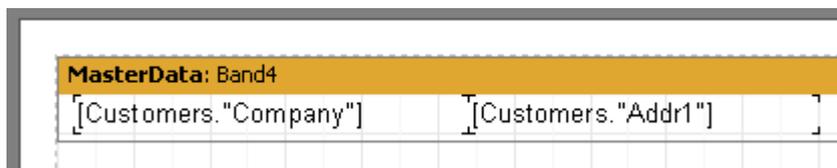
The variable’s value can be displayed in the “Text” object, for example, by placing the [MyVariable] text into it.

A variable’s name should be unique, which means that it should not coincide with the names of the report’s objects, standard functions, and constants. If there is an error in the script, a message will be displayed, and report construction process will be stopped.

6.9 Events

So far we have examined scripts with only one main procedure, which is performed when a report starts running. In the main procedure, one can perform any initial settings, initialize variables. However, this is not enough for total control over the process of report's forming. To control a report as much as possible, every report object has several events, to which a handler (i.e. a procedure from the script) may be assigned. For example, in the handler, connected to the data-band, one can perform records' filtering, which means that the band will be hidden or displayed according to any specified conditions.

Let us demonstrate the process of creation of a report and of events, which are generated during it, with the example of a simple report, which contains one page, one "Master data" band, and two "Text" objects on the band:



As stated previously, the main script's procedure is called in the very beginning of a report. After that, the essential process of report construction starts. In the beginning of the report, the "OnStartReport" event of the "Report" object is called. Before the page is being formed, the "OnBeforePrint" page event is called. This event is called once for each design page of the report's template (it should not be confused with the output pages of a report!). In our case, the event is called once, as the report's design consists of one design page.

Then output of data-bands begins, in the following order:

1. the band's "OnBeforePrint" event is called;
2. the "OnBeforePrint" events of all the objects, belonging to the band, are called;
3. all the objects are filled with data (in our case with values of the "Company" and "Addr1" DB fields); after that, the "OnAfterData" events of all the objects are called;
4. such actions as positioning of objects on the band (if there are stretchable objects among them), calculating of the band's height, and stretching (if it is stretchable) are performed;
5. the band's "OnAfterCalcHeight" event is called;
6. a new output page is formed, if the band does not find room in white space of the page;
7. the band and all of its objects are displayed on the report's output page;
8. the "OnAfterPrint" event of all the band's objects is called;
9. the "OnAfterPrint" event of the band itself is called.

Bands are printed as long as there are data in the source connected to the band. After that, forming of a report stops; the "OnAfterPrint" report's page events and, finally, the "OnStopReport" event of the "Report" object are called.

Thus, by using events of different objects, one can manage practically every moment of report's formation process. A key to using events is a thorough understanding

of the bands' output process, discussed in the next nine sections. Most of the actions can be performed using the band's "OnBeforePrint" event only; any modifications made to an object are displayed immediately. However, in this event it is impossible to analyze, in which page the band will be printed, if it is stretchable, since calculation of band's height will be performed in the step 4. This can be performed either by the "OnAfterCalcHeight" event in the step 5, or the "OnAfterPrint" event in step 8, but in the latter event a band will already have been output so any modification of objects will not have any effect.

One should clearly understand "where and when" bands are output and the timing (firing order) of each of their events, as well as those of objects contained in the band.

6.10 Example of using the "OnBeforePrint" event

To demonstrate create a report, which represents the list of clients. This report will include only those companies, whose names begin with the letter "A."

Let us create a new project in Delphi, put the "TTable," "TfrxDBDataSet," "TfrxReport" components to the form and set them:

Table1:

DatabaseName = 'DBDEMOS'

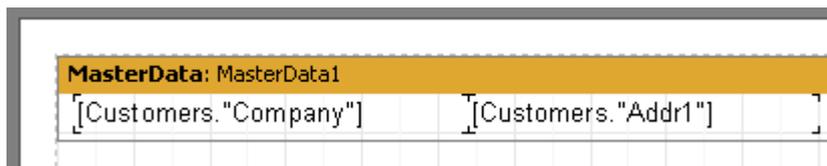
TableName = 'customer.db'

frxDBDataSet1:

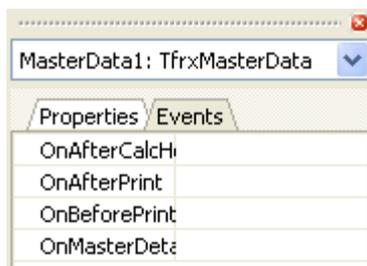
DataSet = Table1

UserName = 'Customers'

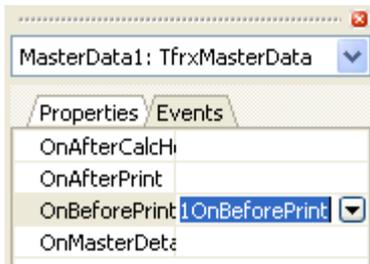
Enter the report's editor and create a report of the following type:



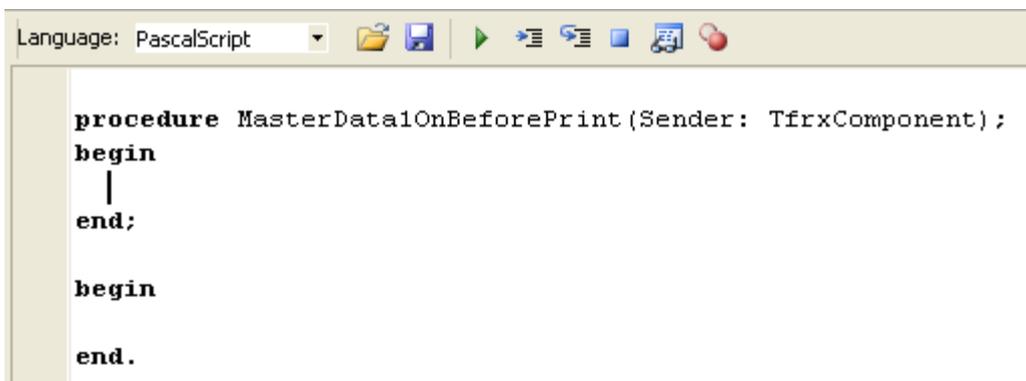
Select the data-band and switch to the "Events" page tab in the objects' inspector:



To create the “OnBeforePrint” event’s handler (this is exactly what would be most appropriate to us), double-click on the blank field to the right of the event’s name:



At the same time, a blank handler is added to the script’s text, and the designer switches to the “Code” page.



The only thing we should do after that is to write the following code in the handler’s body:

PascalScript:

```

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

C++Script:

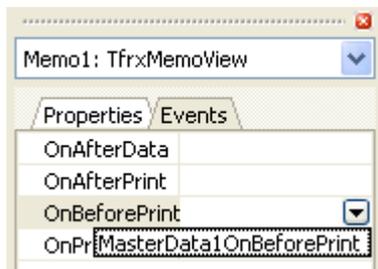
```

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

Run the report and make sure, that the script works correctly:

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

Let us explain several details. You can assign one handler to several events of different objects at once; in this case the “Sender” parameter defines the object, which has initiated the event. To assign a name of the already existing handler to the event, one can either enter it manually in the objects’ inspector, or select it in the pull-down:



The link to the handler can be easily deleted. To do that, select a required property and click the “Delete” key.

6.11 Printing the group’s sum total in the group’s header

This quite often-used method requires use of scripts because total value in an ordinary report becomes available only after all group's records are handled. To display a sum in the group's header (before the group is handled), the following algorithm is used:

- the two-pass option of the report is turned on ("Report|Options..." menu item);
- in the first pass, the sum of each group is calculated and saved in an array;
- in the second pass, the values are extracted from the array and typed in the group's header.

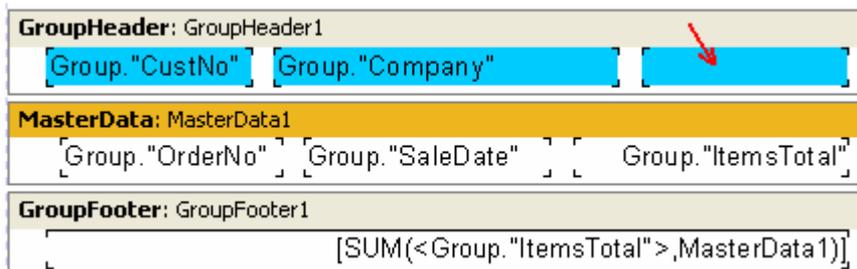
Let us show, two ways of how this task may be accomplished. First of all, let us create a new project in Delphi, put the “TQuery,” “TfrxReport,” and “TfrxDBDataSet” components to the form. Set them in the following way:

```
Query1:
DatabaseName = 'DBDEMOS'
SQL =
select * from customer, orders
```

where orders.CustNo = customer.CustNo
 order by customer.CustNo, orders.OrderNo

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

Enter the designer and connect our data source to the report. Enable the double pass in report's settings (the "Report|Options..." menu item). Add two bands to the report: "Group header" and "Master data." In the "Group header" band's editor, specify the condition ("Group.CustNo" data field). Connect the data-band to the "Group" data source, and then arrange objects in the following way:



For entering sum value, we use the selected object in the picture (in our example its name is "Memo8").

The first way.

We use the "TStringList" class as an array for sums' storage. We will store numeric values as strings. At the same time, the first line in the list corresponds to the value of the first group, etc. The integer-valued variable (which we will augment after printing the next group) is used for calculating the group's number.

Thus, our script will look as follows:

PascalScript:

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

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

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

```

end;

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

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

procedure GroupFooter1OnBeforePrint(Sender: TfrxComponent);
begin
    if not Engine.FinalPass then
        List.Add(FloatToStr(SUM(<Group."ItemsTotal">,MasterData1)));
        Inc(i);
end;

begin

end.

```

C++ Script:

```

TStringList List;
int i;

void frReport1OnStartReport(TfrxComponent Sender)
{
    List = TStringList.Create();
}

void frReport1OnStopReport(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++;
}

```

```
{  
}
```

Looking at the names of the procedures, you can easily find out the events we have used. They are: “Report.OnStartReport,” “Report.OnStopReport,” “Page1.OnBeforePrint,” “GroupHeader1.OnBeforePrint,” and “GroupFooter1.OnBeforePrint.” As for the first two events, they are called, as it was said, in the beginning and in the end of the report respectively. To create handlers for these events, one should select the "Report" object in the "Report tree" window; its properties will appear in the objects' inspector. After that, we would switch to the inspector's "Events" page tab and create the handlers.

Why didn't we use the main procedure for creation of the “List” list in the “OnStartReport” event? That is because the created object should be cleared after a report is finished. This is logical to create objects in the “OnStartReport” event and clear them via the “OnStopReport.” In other cases (when memory does not need to be emptied) one can use the main procedure for initialization of variables.

Everything concerning creation and clearing of the “List” object seems to be quite obvious. Now let us examine the work of the script. In the beginning of the page, the counter of the current group (the “i” variable) is reset to “0” and increments after printing each group (in the “GroupFooter1.OnBeforePrint” event). The calculated sum's value is added to the list in this event. The “GroupHeader1.OnBeforePrint” event does not trigger during the first pass (the “Engine.FinalPass” verification). During the second pass (when the “List” list is filled with values), the value, which corresponds to the current group is retrieved in this event, and it is output to the “Memo8” object's text, which displays the sum total in the group title. In a finished report, it appears as follows:

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

As we can see, the algorithm is rather simple. Nevertheless, it can be simplified.

The second way.

We use the list of report's variables as an array for sums' storage. As we remember, reference to such objects is performed via the “Get” and “Set” functions. This saves us

from having to create extra objects and to free them. Our script will look as follows:

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)));  
}  
  
{  
  
}
```

As you can see, the script was rather simplified. A code in the “GroupFooter1.OnBeforePrint” handler sets a variable's value with a name similar to the client's number (one can use any identifier, which unambiguously identifies the client, for example, his name <Group."Company">). If there is no such variable, it would be created; if there is, its value would be changed. In the “GroupHeader1.OnBeforePrint” handler, a variable's value with the number of the current group is computed.

6.12 “OnAfterData” event

This event is generated after the report's object is filled with the data, to which it is connected. Use this event for analyzing either a DB field value, or an expression contained in the object. The fact is that this value is placed to the “Value” service variable, the value of which is available in this event only. So, having two "Text" objects with the [Table1."Field1"] and [<Table2."Field1"> + 10] contents, one could analyze the value of these expressions referring to the “Value” variable:

PascalScript:

```
if Value > 3000 then
  Mem01.Color := clRed
```

C++ Script:

```
if (Value > 3000)
  Mem01.Color = clRed;
```

instead of writing something like this:

PascalScript:

```
if <Table1."Field1"> > 3000 then
  Mem01.Color := clRed
```

C++ Script:

```
if (<Table1."Field1"> > 3000)
  Mem01.Color = clRed;
```

The use of “Value” instead of an expression provides you with a possibility to write one multipurpose handler of the “OnAfterData” event, and to connect it to several objects.

One more thing is to be noted. If there are several expressions in an object (for example, [expr1] [expr2]) a value of the last expression is transferred to the “Value” variable.

6.13 Service objects

In addition to the objects included in the report (pages, bands, "Text" and other objects), some service objects are available in the script, which may be of some use when managing report's construction. The “Engine” object, which we used in the previous chapter, refers to this kind of objects. The list of service objects is given below:

- Report - the "Report" object;

- Engine - the link to the report's slider;
- Outline - the link to the "Report tree" control element in a preview window.

Let us examine each of these objects.

6.13.1 "Report" object

It represents a link to the current report. The property of this object can be seen when selecting the "Report" element in the "Report tree" window.

Methods:

Method	Description
function Calc(const Expr: String): Variant	Returns the "Expr" expression's value, for example, Report.Calc('1+2') returns "3." Any expression, which is correct in terms of FastReport's, can be transferred as an expression.
function GetDataSet(const Alias: String): TfrxDataSet	Returns a data set with a specified name. The data set should be included into the list of the report's data ("Report Data..." dialogue).

6.13.2 "Engine" object

This is the most useful and interesting object, it represents a link to the engine (FastReport's core, which manages report construction). By using the engine's properties and methods one can construct really exotic report design layouts.

The methods and properties of this object.

Property	Type	Description
CurColumn	Integer	The number of the current column in a multi-columned report. A value can be bound to this property.
CurX	Extended	The current shift of the coordinates on the X-axis. A value can be bound to this property.
CurY	Extended	The current shift of the coordinates on the Y-axis. A value can be bound to this property.
DoublePass	Boolean	Equal to "True," if the report is a two-pass one. Analogous to Report.EngineOptions.DoublePass.

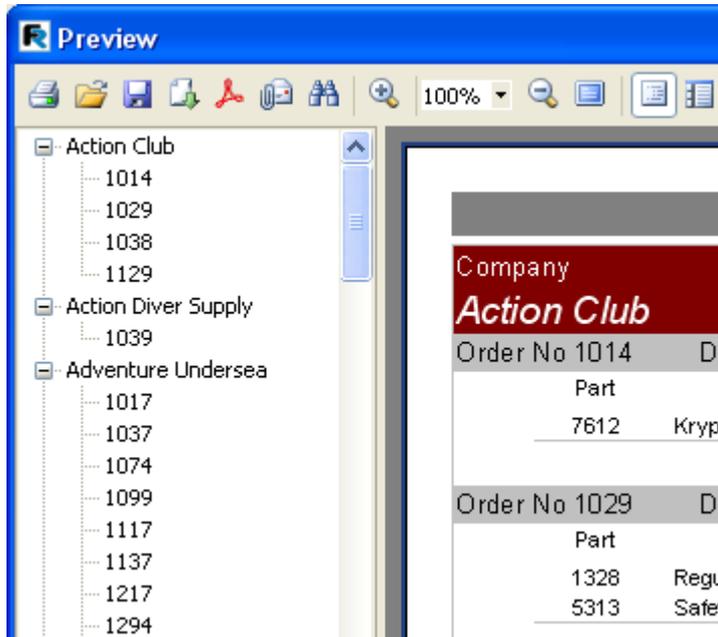
FinalPass	Boolean	Equal to “True,” if the last pass of the two-pass report is performed.
PageHeight	Extended	Printable region’s height, in pixels.
PageWidth	Extended	Printable region’s width, in pixels.
StartDate	TDateTime	Time of report running. A counterpart of the <Date> system variable.
StartTime	TDateTime	Time of report running. A counterpart of the <Time> system variable.
TotalPages	Integer	A number of pages in a report. A counterpart of the <TotalPages> system variable. The report should be a two-pass one, so that this variable can be used.

Methods:

Method	Description
procedure AddAnchor (const Text: String)	Adds “anchor” to the list of anchors. See more below.
procedure NewColumn	Creates a new column in a multicolumn report. After the last column, page break is automatically inserted.
procedure NewPage	Creates a new page (page break).
procedure ShowBand(Band: TfrxBand)	Displays a band with a specified name. After displaying a band, the “CurY” position is automatically shifted.
function FreeSpace: Extended	Returns height value of white space left on a page, in pixels.
function GetAnchorPage(const Text: String): Integer	Returns the number of the page, in which the specified anchor is placed.

6.13.3 "Outline" object

This object represents the "Report tree" control element in a preview window.



This element displays a treelike structure of a finished report. When clicking on any tree node, there is a jump to the page connected to this node. To display the tree, you should either enable it by clicking the  button in the toolbar of the preview window, or specify it with the help of the "Report.PreviewOptions.OutlineVisible=True" property. The control element's width in pixels can be specified there as well: Report.PreviewOptions.OutlineWidth.

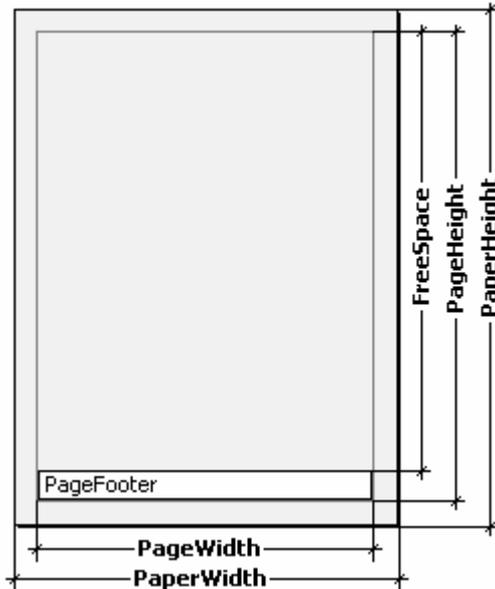
Let us examine this object's methods.

Method	Description
procedure AddItem(const Text: String)	Adds an element with the "Text" name to the current tree position. The current report's page and the current position on the page are associated with the element.
procedure LevelRoot	Shifts the current position in the tree to the root level.
procedure LevelUp	Shifts the current position in the tree on one level up.

6.14 Using the “Engine” object

We have already stated that the “Engine” object represents the report's engine, which manages report's construction. By using the engine's properties and methods, one can manage the process of band(s) arrangement on a page. First some theory.

The picture below displays the report's page and properties' names, which return different dimensions.



The page has the “PaperWidth” and “PaperHeight” physical dimensions. These dimensions correspond to page's properties of the same name that are visible in the objects' inspector when selecting a page. So, size of an A4-format page would be 210x297mm.

The “PageWidth” and “PageHeight” parameters define the dimensions of a printable region, which is usually less than physical dimensions of a page. The size of printable region is defined by the page's fields, which depend on such report page properties as “LeftMargin,” “TopMargin,” “RightMargin,” “BottomMargin.” The printable region's size in pixels is returned by the “Engine.PageWidth” and “Engine.PageHeight” properties.

Finally, the “FreeSpace” parameter defines the height of free space on a page. If there is a "Page Footer" band on the page, its height is taken into account when calculating FreeSpace. This parameter is returned in pixels by the “Engine.FreeSpace” function. Note that after displaying the next band, free space reduces on a page, and this is what is considered during calculating FreeSpace.

How do ready report's pages form? The FastReport core produces bands on the page as long as there is enough free space. When there is no free space left the “Page Footer” band is printed (if available) and a new blank page is formed. As it was already

said, after displaying the next band, the height of free space reduces. Moreover, displaying of a next band begins from the current position, which is defined by coordinates on X-axis and Y-axis. This position returns in the “Engine.CurX” and “Engine.CurY” properties respectively. After printing the next band, the CurY position automatically increases by height value of the printed band. After a new page is formed, the “CurY” position is equal to “0.” The “CurX” position is modified when printing multi-column reports.

The “Engine.CurX” and “Engine.CurY” properties are available not only for reading, but also for writing. That means that bands can be shifted manually by using one of the appropriate events. For example, when you have a report that resembles the illustration below.

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

it can be printed in the following way:

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

This is a result of the script assigned to the band's “OnBeforePrint” event:

PascalScript:

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

C++ Script:

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

Manipulation of the “CurY” property allows, for example, to print bands in splice:

Action Club	Michael Spurling	813-870-0239
Action Diver Supply	Marianne Miles	924-44-500271
Adventure Undersea	Gloria Gonzales	011-342-88644
American SCUBA Supply	Lynn Cinciripini	213-654-0092
Aquatic Drama	Shirley Owen	613-877-7684
Blue Glass Happiness	Christine Taylor	613-886-1684
Blue Jack Aqua Center	Ernest Barratt	401-609-7673
Blue Sports Club	Theresa Kunec	819-827-8374

The corresponding script:

PascalScript:

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

C++ Script:

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

The “Engine.NewPage” method allows page breaks at any required place of a report. At the same time, printing continues from a new output page. In our example one can insert a break after printing the second record:

PascalScript:

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

C++ Script:

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

Note, that we used the “OnAfterPrint” event (that is to say, after the band is already printed). Note the fact that the “Line” service variable returns the sequence number of the record.

The “Engine.NewColumn” method breaks a column in multi-columned reports. As soon as there is no column left, this method creates a new page

6.15 Anchors

Anchor is one of the elements of the hyperlink system, which allows one to jump to any element, connected to the finished report's object by clicking on it (in the preview window).

Anchor is a special tip, which is set via the "Engine.AddAnchor" method. Anchor has a name, which corresponds with the page number position of the page. To jump to an anchor with a specified name, put the following line into the URL property of any report's object:

#AnchorName

or

#[AnchorName]

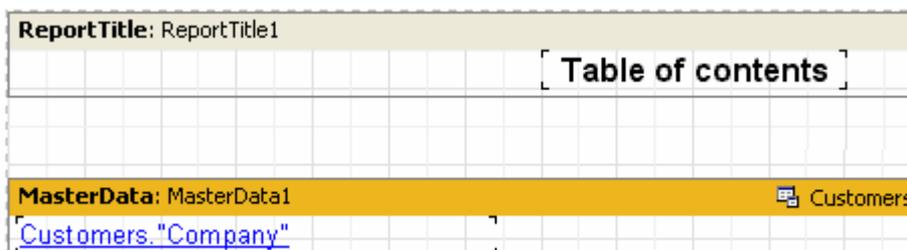
In the latter case, FastReport will substitute a value for the expression.

Clicking on this object executes a jump to the part of the report, where the anchor was added.

Use anchors when constructing the "Contents" chapter, for example with links to corresponding chapters. Let us illustrate this by the following example. To perform this, we need the familiar "Customer" table.

Our report will be a multipage one (with two design pages). We will place the "Contents" chapter on the first page, and the list of clients on the second one. Clicking on the content line executes jumping to a corresponding report's element.

The first design page:



Let us place the following text in the URL property of the "Text" object, which belongs to the data-band

#[Customers."Company"]

and set the font's properties: to blue color and underlining to simulate a hyperlink's appearance.

The second design page:

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

To add an anchor, let us create the “MasterData2.OnBeforePrint” event handler in the band’s script:

PascalScript:

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

C++ Script:

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

That is all we needed. Preview the report, to make sure that our “hyperlinks” work.

The last thing to be mentioned is the “Engine.GetAnchorPage” function. This function returns the number of the page, where the corresponding anchor was added. This function is useful when creating the “Contents” chapter as well. A report must be a two-pass one; otherwise this function cannot be used.

6.16 Using the “Outline” object

The “Outline” object, as previously stated, represents a report’s tree, which can be displayed in a preview window. Clicking on a tree’s element executes jumping to the report’s output page, which is associated to the tree’s element. It is not necessary to use the script for operating with the “Outline,” since some bands have a mechanism, which enables automatic forming of a tree. Let us examine two examples of how the “Outline” can be used with the help of bands and the script.

Almost all bands have the “OutlineText” property, into which a text expression can be entered and this in turn helps to automatically create a tree. The expression will be calculated when forming a report, and its value will be added to the tree when printing the band. Thus, elemental hierarchy in the tree is similar to the bands’ hierarchy in a report. That means, that in the tree there will be main and subordinate elements, corresponding to main and subordinate bands in a report (a report with two levels of data or with groups can exemplify the point). We will illustrate operating with a tree using our previous example of the report with groups..

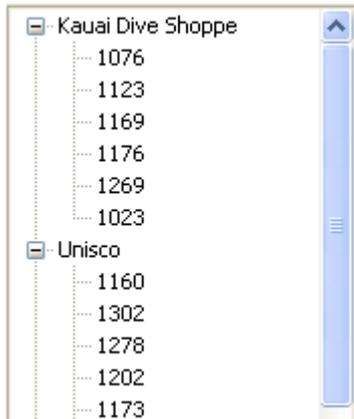
GroupHeader: GroupHeader1		Group."CustNo"
Group."CustNo"	Group."Company"	
MasterData: MasterData1		Group
Group."OrderNo"	Group."SaleDate"	

Specify a value for the “GroupHeader1.OutlineText” band’s property as “<Group."Company">.” To make the tree be displayed automatically as soon as the preview window opens, one should set the “Report.PreviewOptions.OutlineVisible” property = True”. When previewing the report, you would see the following:

Company	OrderNo	SaleDate
Jamaica SCUBA Centre	1651	
	1015	25.05.1988
	1028	07.07.1988
	1128	08.10.1993
	1215	16.11.1994
	1315	26.01.1995
Island Finders	1680	
	1093	01.06.1989
	1016	02.06.1988
	1084	11.05.1989
	1116	21.03.1993
	1034	13.08.1988

Clicking on any element of the tree executes jumping to the corresponding report’s page, and, as a result, the selected element occurs at the top of the window.

Let us add the second level to the report’s tree. Just set the “MasterData.OutlineText” band’s property as “<Group."OrderNo">.” Thus, the tree will look as follows:



As you might notice, the navigation even in the orders' numbers is possible, and hierarchy of the tree's elements resembles the report's hierarchy.

Now we will create a similar tree using script code without using the "OutlineText" property. In our report, clear the "OutlineText" properties of both bands and create two event's handlers: "GroupHeader1.OnBeforePrint" and "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;  
}  
  
{  
  
}
```

Preview the report, to make sure that it works in the same way as the previous report, where the tree was formed automatically. Let us examine, how a tree is formed.

The “Outline.AddItem” method adds a child block to the current tree block, and then makes the child block a current one. Thus, if “AddItem” were called several times in a row, we would obtain a “ladder” as shown below:

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

The “LevelUp” and “LevelRoot” Outline methods are used for controlling the current element. The first one moves the cursor to the element, which is located on a higher level. Thus, the script

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

constructs a tree like this

```
Item1  
Item2  
Item3  
Item4
```

This means, that “Item4” will be a child element in relation to the “Item2” element. The “LevelRoot” method shifts the current element to the root of the tree. For example, the script

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

constructs a tree, like the one below

```
Item1  
Item2
```

Item3

Item4

Thanks to these explanations, it becomes clear, how our report works. Every time when outputting a group title, the root of the tree becomes the current element, where a company's name is added. After that, the list of orders is output, and each order is added as a child element of the company. To make the number of orders located on one level, not display as a "ladder", the transition to the upper level via the "Outline.LevelUp" method is performed in the script.

6.17 "OnManualBuild" page's event

The FastReport core is usually responsible for report construction. It displays a report's bands in a definite order, as many times, as there are data, thus forming a finished report. Sometimes it is necessary to display a report in a non-standard form, which the FastReport core is unable to create. In this case, one can use the ability to construct a report manually via the "OnManualBuild" event, of the report's design page. If the handler of this event were defined, then when forming an output page the FastReport core would transfer control to it. At the same time, the report's core automatically displays the bands located in the page, such as "Report title," "Page title," "Column title," "Report footer," "Page footer," "Column footer," and "Background." The core also handles the process of forming of new pages and columns. The task of the "OnManualBuild" event's handler is to display data bands and their titles and footers in a definite order.

That is to say, the "OnManualBuild" handler's essence is to give a command for displaying particular bands to the FastReport's core. The core will do the rest itself: it will form a new page, as soon as there is no free space on the current one; handle the scripts attached to events; etc.

Let us demonstrate a handler with a simple example. In the report, there are two master data bands, which are not connected to data:

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

The handler will display these bands in alternate order (six times for each one). After six bands are created, a small gap will be inserted.

PascalScript:

```

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

```

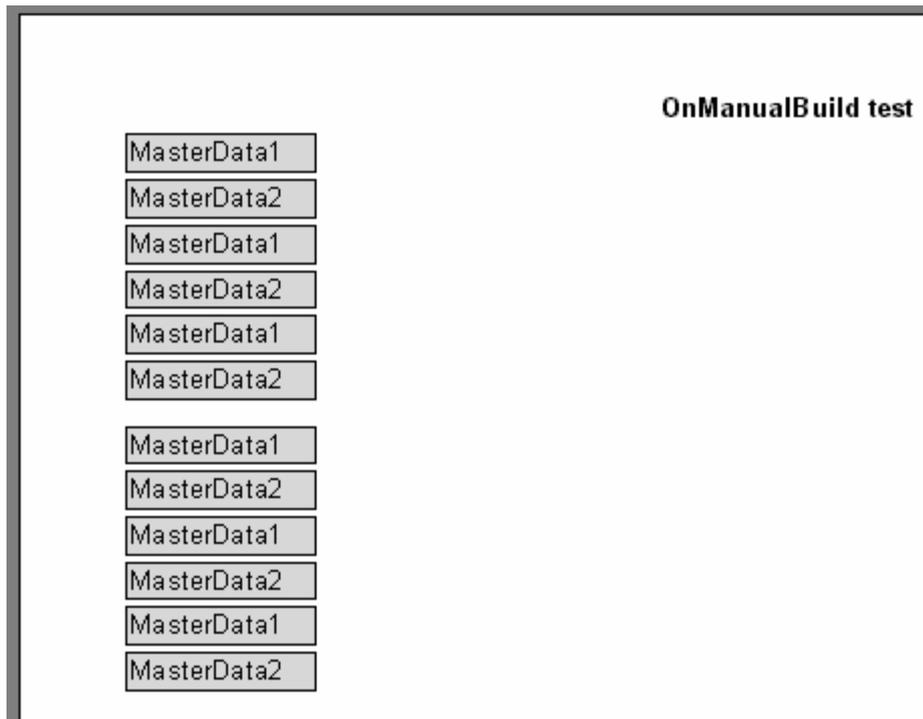
C++ Script:

```

void Page1OnManualBuild(TfrxComponent Sender)
{
    int i;

    for (i = 1; i <= 6; i++)
    {
        //
        Engine.ShowBand(MasterData1);
        Engine.ShowBand(MasterData2);
        //
        if (i == 3)
            Engine.CurY = Engine.CurY + 10;
    }
}

```



The following example displays two bands' groups next to each other.

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:

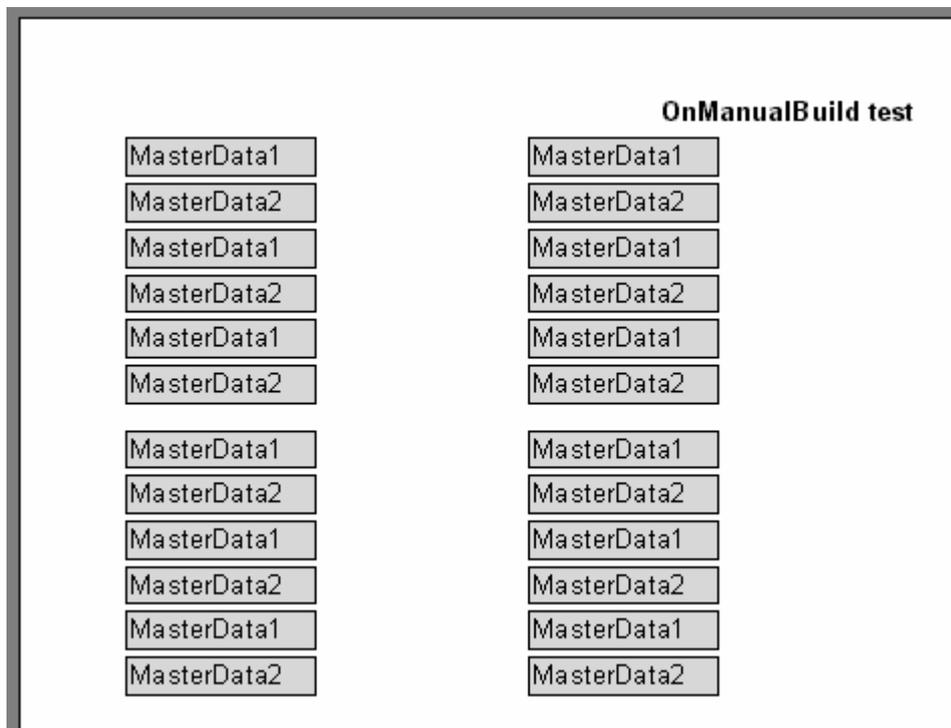
```

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;
}
}
```



As you can see, in these examples we controlled output of data-bands only. All the rest bands (for example, in our case, it was “Report title”) were printed automatically.

Finally, we will demonstrate, how to construct a report of the “List of clients” type (we have constructed it several times in this manual) using the “OnManualBuild” event. In our example, connect the data-band to the data source.

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

Event's script is the following:

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();
    }
}

```

Preview the report, to make sure that the result of the script's work does not differ from a standard report. Note how we got a link to the Dataset; in our example we

connected a band to the data source, and used the script code

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

line to return a link to the data source. If a band is not connected to the source, the link to the required source can be achieved in the following way:

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

Of course, the source, we are interested in, must be added to the report in the menu “Report|Data...” dialogue.

6.18 Creation of objects in the script

One can add new objects into a report by using the script. Let us demonstrate with a simple example, how it is performed. Create a blank report, and then write in the main script's procedure:

PascalScript:

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

C++ Script:

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

Preview the report:

This memo is created in code

Note, that we never destroy the created **FastReport objects** in these examples. It is not required, since FastReport objects are automatically destroyed after the report is completed.

Chapter



**Cross-tab
reports**

This kind of report has a table structure, which means that it consists of rows and columns. At the same time, it is not known beforehand, how many lines and columns a table would possess. That is why a report grows not only downwards (as the report types examined above) but sideways as well. A typical example of a report of such type is shown below.

Let us examine the elements of the table:

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

In the illustration, we see a table with two lines (rows) and four columns, where “a” and “b” are line titles, “1,” “2,” “3,” and “4” are column titles, and “a1”..”a4,” “b1”..”b4” are cells. To construct a report like this, we need just one set of data (a query or a table), which has three fields and contains the following data:

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

As you can see, the first field contains a line number, the second one has a column number, and the third one contains the cell contents at intersection of the table with the selected number. When outputting a report, FastReport creates a table in memory and fills it with data. Thus, the table expands dynamically, if a line or a column with a specified number does not exist.

Titles can consist of more than one level. See the following illustration:

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

In this illustration, the number, or index of the column is composite, i.e. it consists of two values. This report requires the following data:

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

In this example, the first field contains the line index, as it was before; the second and the third fields contain column indexes. The last field contains the cell value. Examine the following picture in order to make it clear, how FastReport constructs a tables with complex titles:

	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

Before handling is accomplished, our table would look like the table shown in the picture. During handling, FastReport unites the title cells with equal values, which are allocated on one level.

The next table element, which is shown in the following picture, displays intermediate totals and totals:

	10			20			Total
	1	2	Total	1	2	Total	Total
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)
Total	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)

This report is constructed using the same data, as we used in the previous one. The columns, highlighted with gray in the picture, are calculated automatically and are not included in the initial data set.

7.1 Construct a cross-report

Now let us turn from theory to practice. We will construct a simple cross-report, which will display employees' salary during four years. To do this, we need the "crosstest" table, which is available in the FastReport "DEMOS\MAIN" folder. The table contains

data of the following kind:

<u>Name</u>	<u>Year</u>	<u>Salary</u>
Ann	1999	3300
Ben	2002	2000
....		

Create a new project in Delphi, put the “TTable,” “TfrxDBDataSet,” and “TfrxReport” components on the form and set them:

Table1:

```
DatabaseName = 'c:\Program Files\FastReport 4\Demos\Main'  
TableName = 'crosstest.db'
```

the DatabaseName property value of course must correspond with the path to your FastReport installation folder!

frxDBDataSet1:

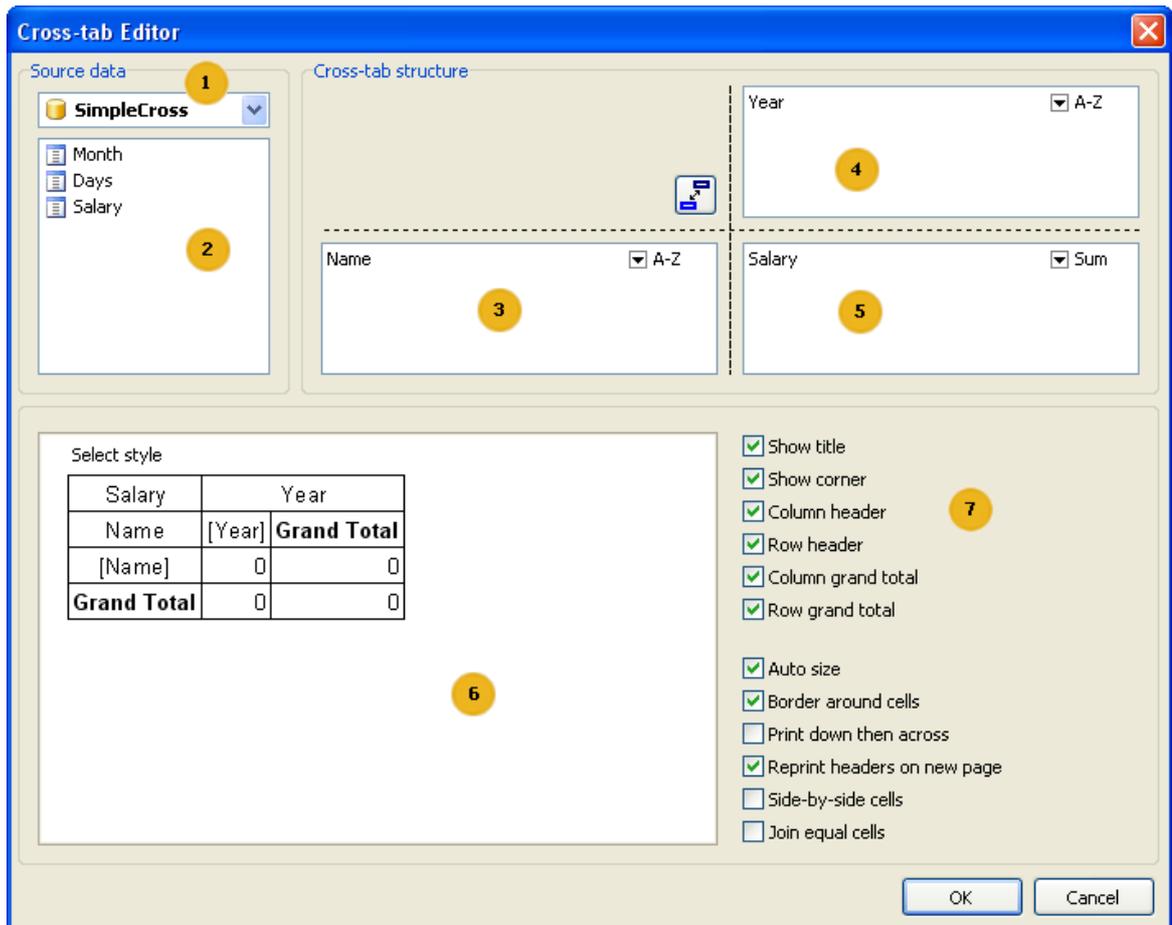
```
DataSet = Table1  
UserName = 'SimpleCross'
```

For cross-reports designing, one should use the “TfrxCrossObject” component  from the FastReport component palette. Just put it on the Delphi form; it is not required to set anything. At the same time, the “frxCross” unit, which contains all necessary functionality, will be added to the "uses" list.

Enter the report design mode. First of all, connect our data source using the “Report|Data...” menu. Select the “DB cross-tab” object  from the list: Click on the design page to place the object:



All settings are specified using the object’s editor. Call it by double-clicking on the object:



The Numerical list of the Editor's items from the illustration above:

- 1 – A drop-down list with available data sources;
- 2 – The list of fields in the selected data source. The fields from this list can be dragged to the “3,” “4,” and “5” lists;
- 3 – The list of fields, which generate a line (row),header;
- 4 – The list of fields, which generate a column header;
- 5 – The list of fields, which generate a table cell;
- 6 – Table structure preview;
- 7 – Structure options here one can specify whether it is necessary to display titles and totals.

You can only use the mouse in this editor to make modifications. For our demo, it is enough to drag fields from the “2” list to the “3,” “4,” and “5” lists, as shown in the illustration above. After that, close the editor by clicking the “ ” button. We can see that the object displays its structure now:



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

When previewing the report now, you will see a table resembling the one shown below:

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

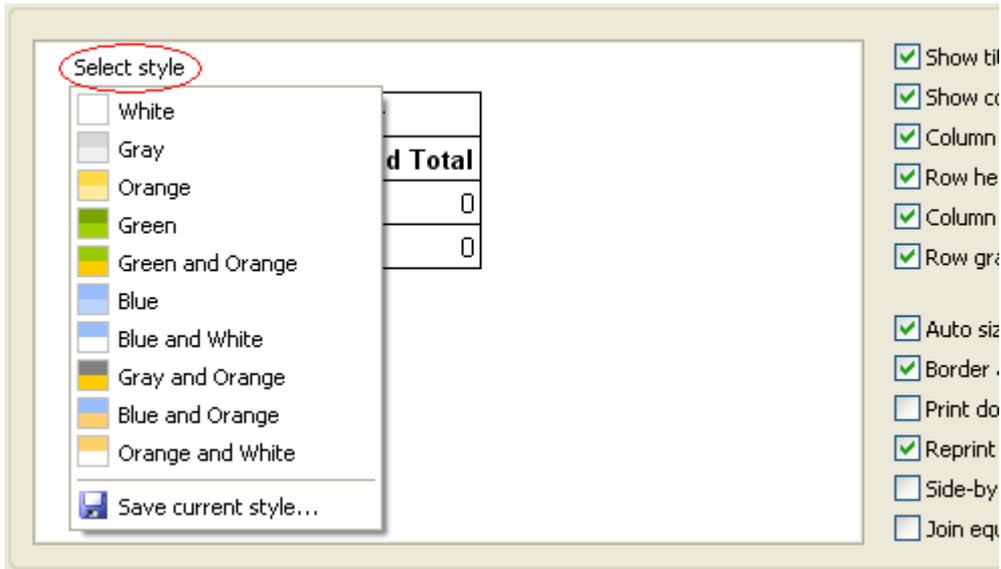
7.2 Changing appearance

Let's modify the object's appearance. The first thing we want to do, is to modify the titles' colors and to display "Total" instead of "Grand total." It is very easy to do. To change the title color into gray, click on the "Year," "Name," and "Grand Total" elements one after another, and then select the desired color using the  button in the toolbar.



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

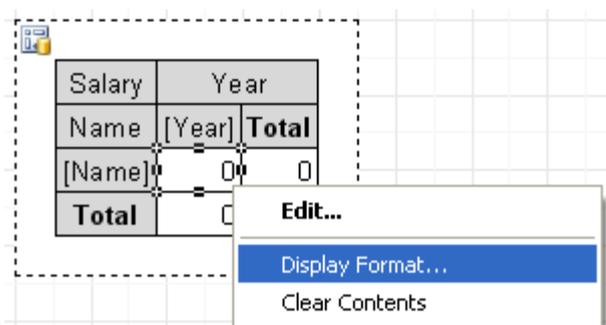
You also can use a set of predefined styles. It is available in the cross-tab editor: click the "Select style" and choose one you like.



To change the “Grand Total” text, double-click on the cell, and then you will see the familiar text editor, where one can type “Total”:

Salary	Year	
Name	[Year]	Total
[Name]	0	0
Total	0	0

To set the format of the currency values, select the first cell (on cross of [Name] and [Year] in our case), right-click to display a context menu and select the "Display Format...":



Then select the required format and close format editor. You will get the following result:

Salary	Year				
Name	1999	2000	2001	2002	Total
Ann	\$3 300,00	\$2 700,00	\$3 100,00	\$1 700,00	\$10 800,00
Ben	\$3 900,00	\$2 100,00		\$1 800,00	\$7 800,00
Catherine	\$6 100,00	\$3 200,00			\$9 300,00
Den		\$3 999,00	\$8 100,00		\$12 099,00
Total	\$13 300,00	\$11 999,00	\$11 200,00	\$3 500,00	\$39 999,00

7.3 Using functions

In our previewed example, we see the sum total of each employee's salary during four years in the "Total" line. One can use the following functions:

SUM – sum of values

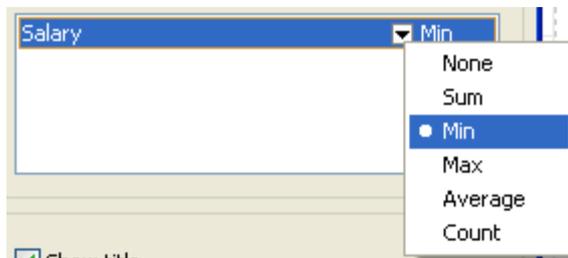
MIN – minimal value

MAX – maximal value

AVG – average value

COUNT – number of values

Let's use the "MIN" function in our example. Open the cross-object editor in area (6) the "Salary" field item, click the down arrow.



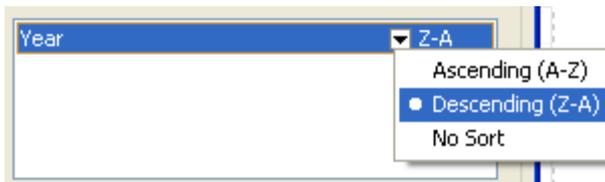
Select the "MIN" function in the menu. Now one can modify a text in the cell of totals from "Total" to "Minimum." A finished report will look as follows:

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

7.4 Sorting values

Lines and columns values are arranged in ascending order, if values have numerical type, they are sorted by value, and if they have line type, they are sorted alphabetically. We can separately set our own sorting mode for each line and/or column value. The following modes are available: “arrange in ascending order,” “arrange in descending order” and “perform no sorting.” In the latter case, values in lines/columns will be displayed depending on their entries.

Let us modify column sorting in our example. Let years be arranged in decreasing order. To perform this, let us enter the cross-object editor and select the “Year” column element. To modify sorting, click on the item’s down arrow select descending:



Close the editor and preview the report. It will look as follows:

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

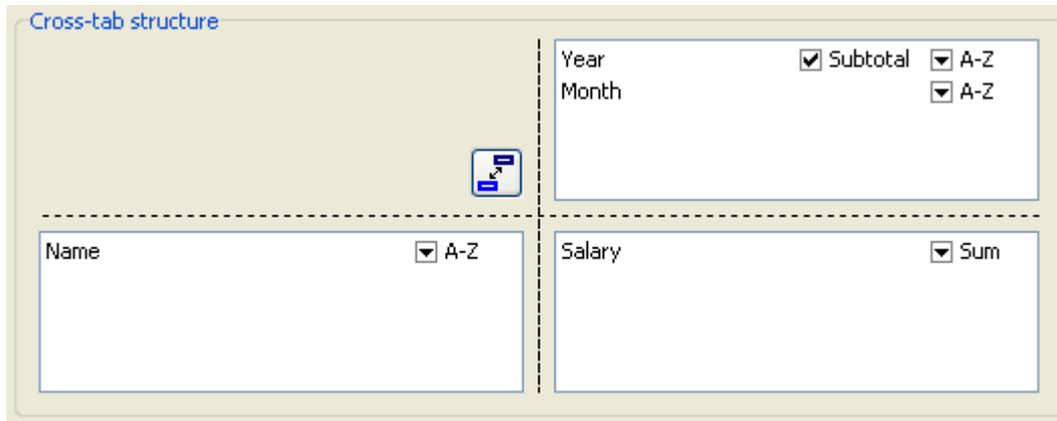
7.5 Table with composite headers

Our previous example contained one value per line, and column headers. Let us examine cross table design using a complex header, which means that it will contain two or more values. The table contains data of the following kind:

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

We have added the “Month” and “Days” fields, which contain month number and the number of working days respectively. One can construct several reports on the basis of this data, for example, salary of all the employees during all years, broken down by months.

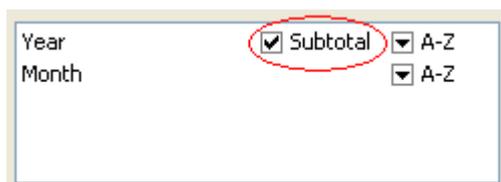
What kind of a report we are going to get? It must resemble the report from the previous example, but at the same time it must be broken down by months as well. The cross-object must be set in the same manner. We'll add the "Month" field into the column header by dragging it to the column header list. See the illustration below.



As a result, when previewing, we would see the following report:

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

Note, that FastReport automatically added a column of the intermediate totals, which are displayed after each year. This option can be set in the cross-object editor, by selecting the "Year" column element and disable the "Subtotal" flag:



In addition, one can note that there is no intermediate total in the bottommost column element (the same is true in cases, when this element is the only one). Actually, in our example, we do not need intermediate totals for each month.

Another feature of intermediate totals: In our example, it is desirable to display "Year + year total" instead of the "Total" text. In the cross-object editor, select the required object in the bottom part of the editor, and then enter the following text to it:

Total for [Value]

During construction, the “Value” expression will be replaced by the actual value of the column header value, located above:

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

7.6 Adjusting cell width

When looking at the previous illustration, it should become obvious that FastReport automatically adjusts cell width in a way, which allows the longer lines to fit the cells. It is not desirable in some cases however, since values with very long text lines become ugly. What can be done in such case? Let's look at 3 ways of changing the width.

The simplest way is to break lines in the text of object with intermediate totals, i.e. to insert a line into it:

*Total
for[Value]*

You see that the table appears more compact now:

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

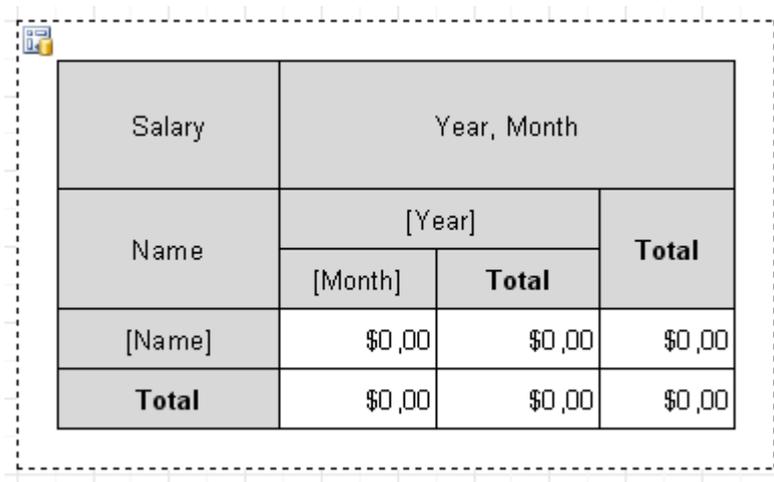
However, this method cannot be used if the lines'/columns' values are rather long, they cannot be corrected by breaking the line manually. This is why the cross-object has the “MinWidth” and “MaxWidth” properties (minimal and maximal cell width respectively). Both these properties are accessible only via the object inspector.

The “MinWidth” value is “0,” and the “MaxWidth” value is “200” by default. This is quite enough in most cases. You can set your values, according to any special requirement you desire.

Thus, in our example, we can set the following: MinWidth = MaxWidth = 50. This would signify that table cell width must be 50 pixels at any rate. If a cell is smaller, it is “adjusted” to the “MinWidth” value, if it is bigger, its width is fixed according to the “MaxWidth” value, and the text in the cell is divided. In our example, it would appear as follows:

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

The third way is to change table width manually. To do this, set AutoSize property to False. Now you are able to resize the cross-tab using the mouse. When moving the mouse cursor over cross elements, you will see that cursor shape changes. Here is an example of what we can do:



Salary	Year, Month		
Name	[Year]		Total
	[Month]	Total	
[Name]	\$0,00	\$0,00	\$0,00
Total	\$0,00	\$0,00	\$0,00

Remember that if you turn off the auto size, the cross-tab will not adjust the widths/heights of the table elements. You may get something like this when previewing our table:

Salary			
Name	1999		
	2	10	11
Ann	\$1 000,0 n		\$1 100,0 n
Ben		\$1 900,0 n	\$2 000,0 n
Catherine		\$3 000,0 n	\$3 100,0 n

In this case, just increase a cell width a little.

7.7 Font colors and highlighting

Sometimes it is necessary to highlight values and/or change font color. We have examined such a task in the group report example. Where we used conditional highlighting for the “Text” object, this can be useful for us now as well.

To add highlighting: using our example, assume that we need to change font color for the values, which are more than 3000. Click on the object, representing the table cell. To set highlighting parameters, click on the  button in the toolbar. The already familiar highlighting editor window will open, where one can set the following condition:

Value > 3000

and the font color to red:



Salary	Year, Month		
Name	[Year]	Total	Total
[Name]	0	0	0
Total	0	0	0

Highlight ✕

Condition

Value > 3000 fx

Font

Bold

Italic

Underline

Color...

Background

Transparent

Other

Color...

OK Cancel

This is all we need. Close the editor by clicking on the “OK” button and preview our report:

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

In the same way, a user is able to highlight total values, columns and lines, if

necessary.

7.8 Managing a cross-table from the script

If setting table visual resources is not enough, one can use the report's script to adjust settings for the appearance of the table. The "Cross-table" object has the following events:

Event	Description
OnAfterPrint	Event is called after printing a table.
OnBeforePrint	Event is called before printing a table
OnCalcHeight	Event is called before calculating length of a row in the table. The event handler can return either the required value of height, or "0" when the row needs to be hidden.
OnCalcWidth	Event is called before calculating column's width in a table. The event handler can return either the required value of width, or "0" when the column needs to be hidden.
OnPrintCell	Event is called before displaying a table's cell. The event handler can modify the cell's design or its contents.
OnPrintColumnHeader	Event is called before displaying a title of the table's columns. The event handler can modify design or contents of the title's cell.
OnPrintRowHeader	Event is called before displaying a title of the table's rows. The event handler can modify design or contents of the title's cell.

We can use the following methods of the "Cross-table" object in events:

Method	Description
function ColCount: Integer	Returns the number of columns in a table.
function RowCount: Integer	Returns the number of rows in a table.
function IsGrandTotalColumn (Index: Integer): Boolean	Returns "True," if the column with specified number is the total one.

function IsGrandTotalRow (Index: Integer): Boolean	Returns “True,” if the row with specified number is a total one.
function IsTotalColumn (Index: Integer): Boolean	Returns “True,” if the column with specified number is a column with intermediate totals.
function IsTotalRow (Index: Integer): Boolean	Returns “True,” if the line with specified number is a line with intermediate totals.
procedure AddValue(const Rows, Columns, Cells: array of Variant)	Adds a value to the table.

Let us show, how one can highlight the third column (in our example it is the “November 1999” date). To do this, select the cross-table object from the report design page, in the object inspector click on the events page tab, locate the OnPrintCell event and create the handler’s on the script page by dbl-clicking in the empty list to the right of the event name, the script editor will appear with the basic declaration created for you then add the code required in the empty begin end block of the declaration:

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; }
}
```

We will see the following result when the report is previewed:

	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

To highlight a column title, create an “OnPrintColumnHeader” event handler, as described above :

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;
  }
}

```

Result would appear as follows:

	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

How the scripts work. The “OnPrintCell” event handler is called before printing a cell included in the table’s body (when printing cells from the table title, either the “OnPrintColumnHeader,” or the “OnPrintRowHeader” handler is called). At the same time, a link to the “Text” object, which represents a table’s cell (“Memo” parameter), and the cell’s “address” in two variants: the number of row, column and cell (the last is relevant, if your cross-table contains multi-leveled cells) in the “RowIndex,” “ColumnIndex,” and “CellIndex” parameters respectively, are transmitted into the “OnPrintCell” handler. The “RowValues” and the “ColumnValues” parameters are the second variant of the “address.” The “Value” parameter is the cell’s contents.

To specify an “address,” you can use the second variant (RowValues, ColumnValues), since it is easier in the given case (as well as the first one (RowIndex, ColumnIndex)). In our case, it was necessary to highlight the third column; therefore, it would be more convenient to analyze the first variant. Since numbering of columns and rows begins with “0,” the “ColumnIndex = 2” checking allows us to define the third column. One could do it in a different way, i.e. by analyzing the required column by its data (we need the 11th month of 1999):

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;  
    }  
}
```

Values, which are transferred in the “RowValues” and the “ColumnValues” parameters, are arrays of the “Variant” type with a zero base. The zero element is a value of the highest level of the table’s title; the first one is a value of the next level, etc. In our case, the “ColumnValues[0]” contains years, and the “ColumnValues[1]” contains months.

Why is “VarToStr” function necessary? This guarantees absence of errors during type conversion. When operating with the “Variant” type, FastReport attempts to automatically cast the strings to number format, which, in its turn, can lead to an error when attempting to cast the “Total” and “Grand Total” columns’ values.

The “OnPrintColumnHeader” event handler is called during output of column title cells. The set of parameters is similar to the parameters of the “OnPrintCell” handler, although in this case the cell’s “address” (the “HeaderIndexes” and “HeaderValues” parameters) is transferred in a different way. The “HeaderValues” parameter returns the same values, as the “ColumnValues” and “RowValues” parameters in the “OnPrintCell” handler. The “HeaderIndexes” parameter is also an array of values of the “Variant” type, which contains an address of the title’s cell in a different form: the zero element is the serial number of the highest level of the table’s title, the first one is the number of the next level, etc. To make the principle of cells numbering clear, refer to the picture below:

	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

In our case, it is easier to analyze the “HeaderValues” value, but one can write the following handler as well:

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; }
}

```

7.9 Adjusting rows/columns size

The user can adjust width and height of the table's rows and columns using the "OnCalcWidth" and "OnCalcHeight:" events' handlers. Let us show how to increase width of the column, which corresponds to the 11th month of 1999 by the following example. To do this, create the "OnCalcWidth" event's handler:

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;
  }
}

```

And the result would appear as follows:

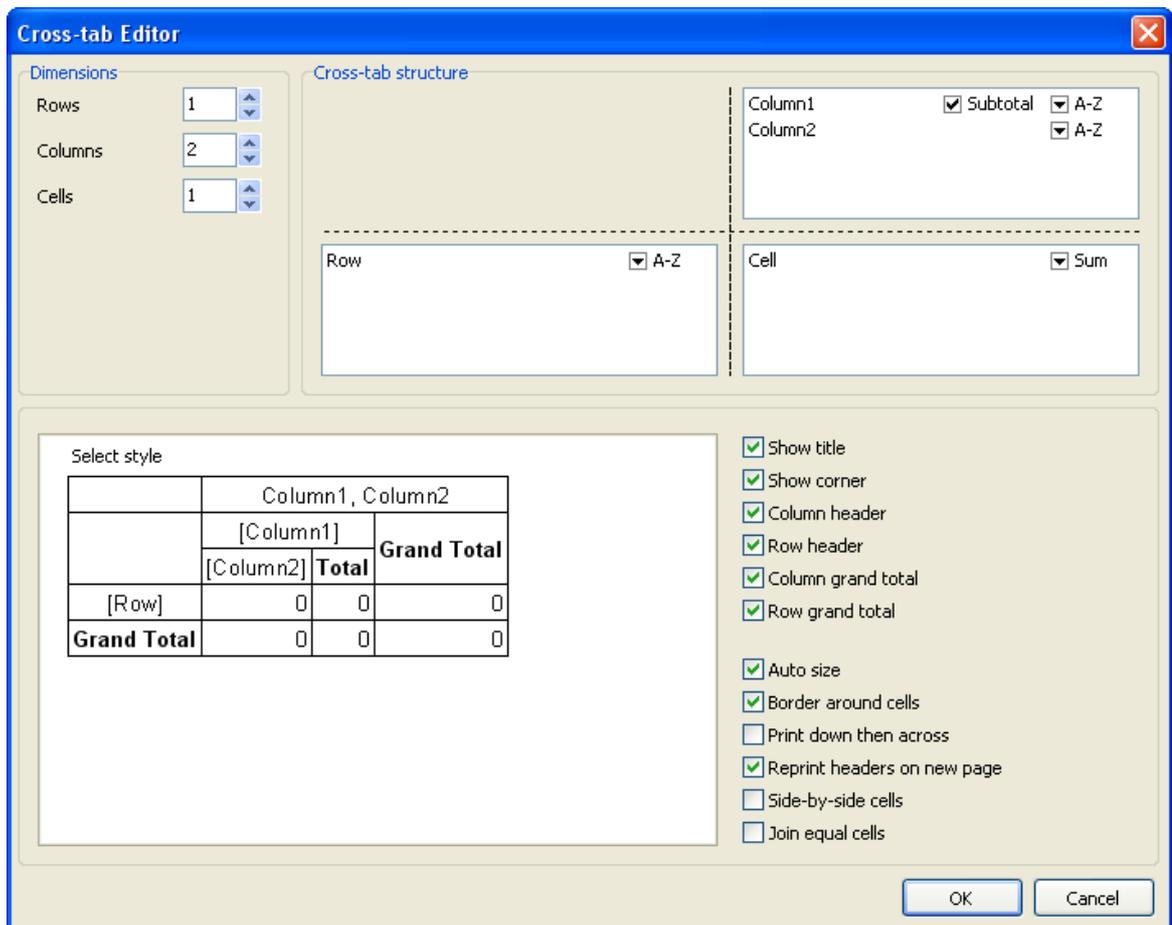
	1999				
	2	10	11	12	Total
Ann	1000		1100	1200	3300
Ben		2100	2200		4300
Catherine		3000	3100		6100
Den					0
Grand Total	1000	5100	6400	1200	13700

In our example, to hide a column, it is enough to return the Width := 0. Note, that the sums are not recalculated at the same time, since the matrix is already full of values at this time.

7.10 Filling a table manually

There are two versions of the cross-table: the “DB cross-table” and the “Cross-table.” All this time we’ve been working with the first object attached to the data from the DB table and fills itself automatically, as soon as the report runs. Let us examine the second object, “Cross-table.”

This object is not attached to the data from a DB. Therefore, you have to fill the cross-table with data manually. This object possesses a similar editor, but you will have to select the number of dimensions in the table’s titles and in its cells instead of DB fields:



Let us demonstrate using the “Cross-table” object with an example. Put an object on the report design page and set it as shown in the illustration above: the number of levels in the strings’ title is “1,” in the columns’ title – “2,” in the cell – “1.” To fill the table with data, let us use the “OnBeforePrint” object’s event handler:

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]);
}

```

In the handler, it is necessary to add the required data into the table via the “TfrxCrossView.AddValue” method. This method has three parameters; each of them is an array of values of the “Variant” type. The first parameter is the row's value, the second one is the column's value, and the third one contains the cells' values. Note that the number of values in each array should correspond to the object's setting! In our case, the object has one level in the rows' title, two levels in the columns' title, and one level of cells. Therefore, we transfer one value for rows, two values for columns, and one value for cells into the AddValue.

When running the report, we would see the following:

	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

One can use the “AddValue” method for the “DB cross-table” object as well. This allows adding the data (which are not in the data source attached to the object) into the cross-table. Otherwise, if there are such data, they are summarized with the data in the

table.

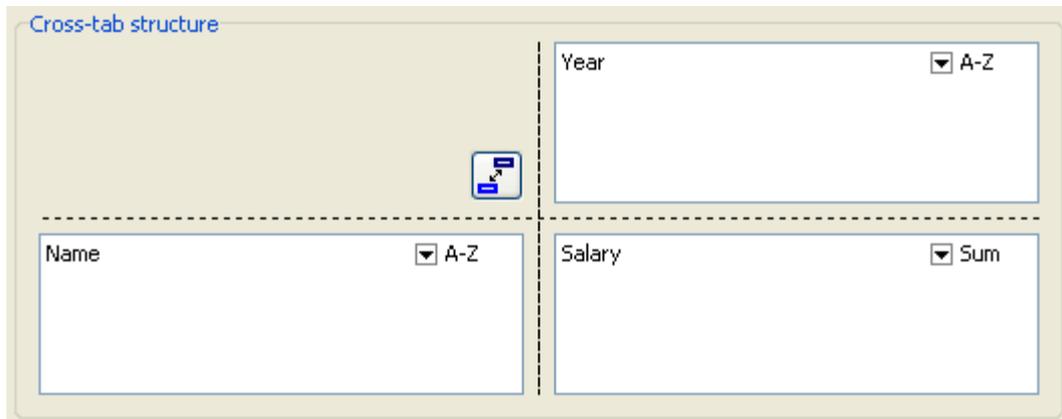
7.11 Add external objects to the table

You can put external objects (such as lines, shapes, pictures) into the cross-table. What for? For example, you may need to show some values in a graphic form. Let's look at an example that uses shapes to display a little progress bar:

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

Red bar is displayed if cell value is less than 100, yellow - less than 3000, green - more that 3000.

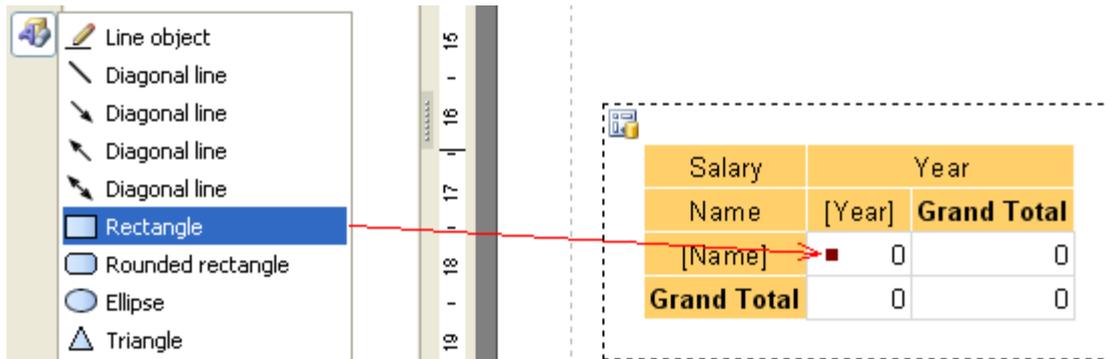
Let's start with our report. Put the "DB Cross-tab" object on a report page and setup it's properties.:



Turn off the "Auto Size" property and setup the column widths as shown below:

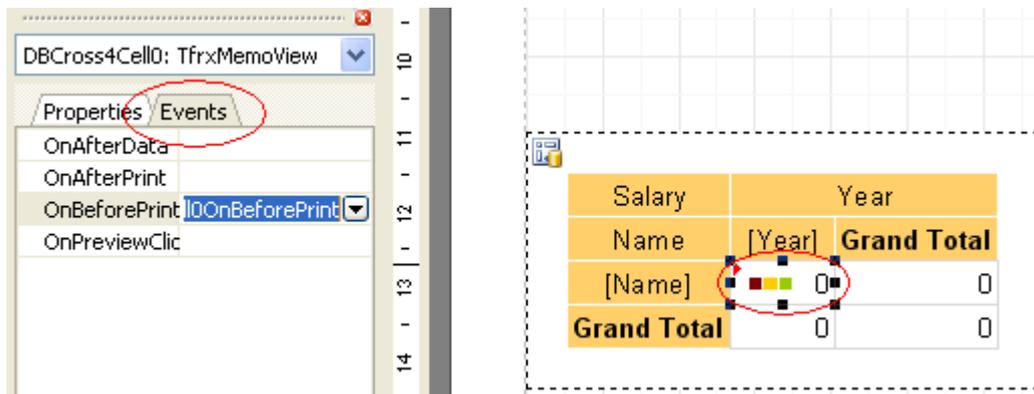
Salary	Year	
Name	[Year]	Grand Total
[Name]	0	0
Grand Total	0	0

Now we add the shape object into our table. To do this, select the "Rectangle" object and put it inside the cell:



In the same way put another 2 rectangles.

Now create a script that will display the needed number of colored shapes (depending on cell value). To do this, select the cell and create OnBeforePrint event handler:



Write the following code in the event handler (pay attention to the shape names: our inserted shapes have exactly these names):

```

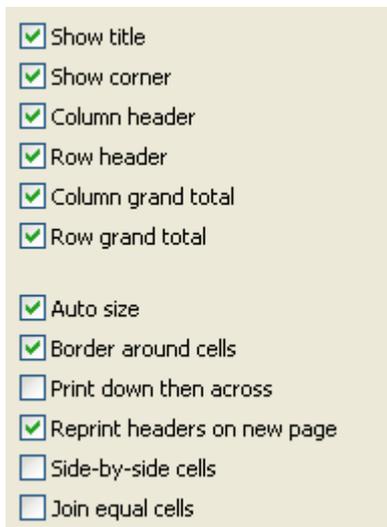
procedure DBCross1Cell10OnBeforePrint(Sender: TfrxComponent);
begin
    // Value it's a current cell's value
    
```

```
if Value < 100 then
begin
  // first shape object
  DBCross1Object1.Color := clMaroon; // red
  // second shape object
  DBCross1Object2.Color := clWhite;
  // third shape object
  DBCross1Object3.Color := clWhite;
end
else if Value < 3000 then
begin
  DBCross1Object1.Color := $00CCFF; // yellow
  DBCross1Object2.Color := $00CCFF;
  DBCross1Object3.Color := clWhite;
end
else
begin
  DBCross1Object1.Color := $00CC98; // green
  DBCross1Object2.Color := $00CC98;
  DBCross1Object3.Color := $00CC98;
end;
end;
```

That's all - run our report and will see the same picture as at the beginning of this chapter.

7.12 Some useful settings

Let's look at some settings available in the cross-table editor.



First six options allow you to show or hide some table elements.

The "Auto size" option is already familiar. It allows us to set table width and height manually.

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

The "Reprint headers on new page" option determines if it is necessary to print table headers on each new preview page.

The "Side-by-side cells" option is used if you have two or more cell values in a table. It determines if it is necessary to print cells side-by-side or stacked (default).

The "Join equal cells" option allows printing side-by-side cells with equal values as one big cell:

Days	Year				
Name	1999	2000	2001	2002	Grand Total
Ann	3	4	2		12
Ben	4	2		2	8
Catherine	6	3			9
Den		4	7		11
Grand Total	13	12	11	4	40

Using Object inspector you can also setup the following properties:

- AddWidth, AddHeight - adds specified amount of space to the cell width or height. It will be used when calculating cell size (the AutoSize options must be on);
- NextCross - pointer to the next crosstab that will be displayed side-by-side to this one;
- NextCrossGap - gap between side-by-side crosstabs.

Chapter



Charts

FastReport allows us to insert charts into the report. For this purpose, the “TfrxChartObject”  object from the FastReport component palette is used. The component is based on the “TeeChart” library, which is included in Delphi distribution kit. One can also use the “TeeChartPro” library, which can be obtained separately.

Let us illustrate a simple construction of a chart using the following example. To perform this, we would need the “country” table from the “DBDEMOS” demo database distribution kit. The table contains data about countries, their area and population:

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

Create a new project in Delphi. Put the “TTable,” “TfrxDBDataSet,” “TfrxChart” and “TfrxReport” components on the form and then customize them:

Table1:

DatabaseName = 'DBDEMOS'

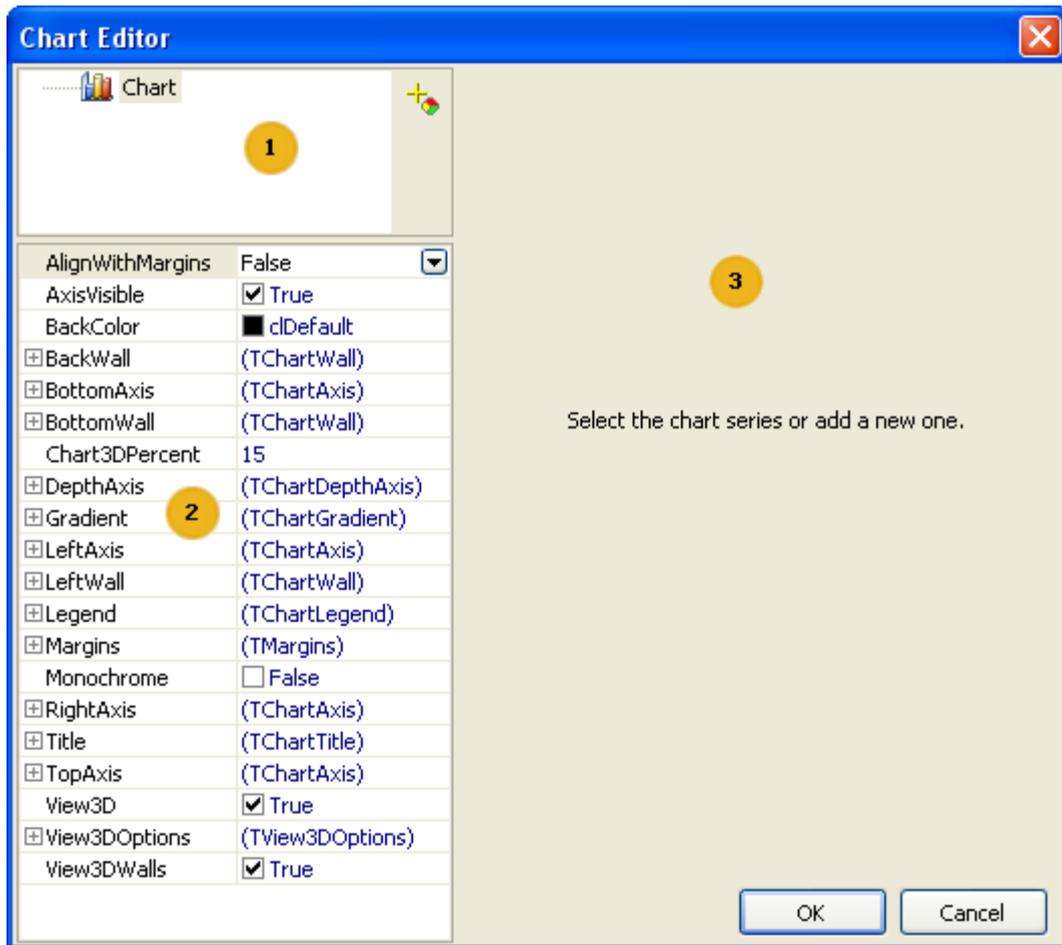
TableName = 'country.db'

frxDBDataSet1:

DataSet = Table1

UserName = 'Country'

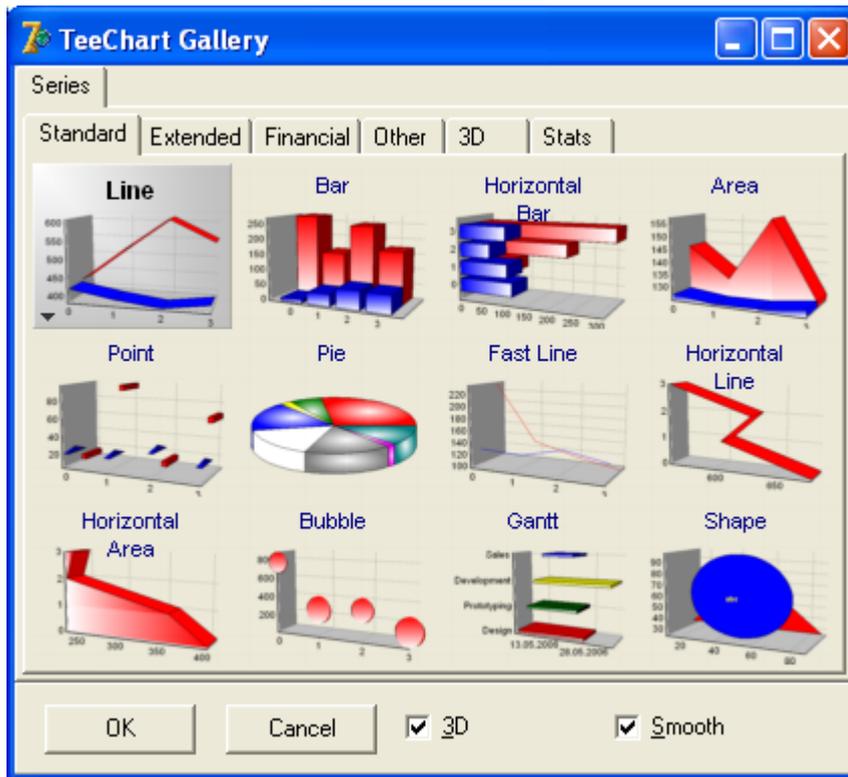
Let us enter the report designer and connect the data source in the “Report|Data...” window. Add the “Chart” object  to the report design page. Set the object size (18x8 cm). To customize the object, call its editor by double-clicking on it.



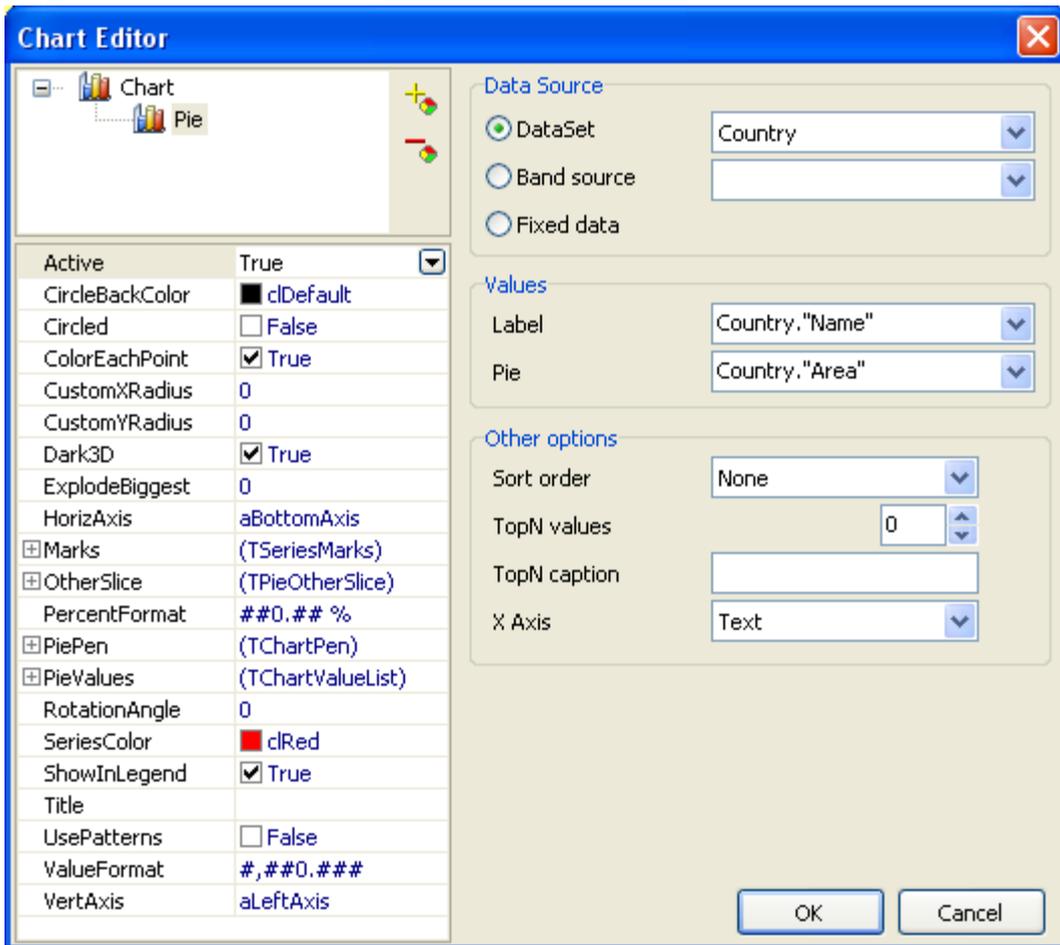
The areas of the chart editor in the illustration:

- 1 – chart structure. A chart can contain either one or several series.
- 2 – object inspector, which displays the properties of the element selected in the window. Teak the chart's properties here.
- 3 – toolbar for connection the series to data; it is activated once the series in the window 1 is selected.

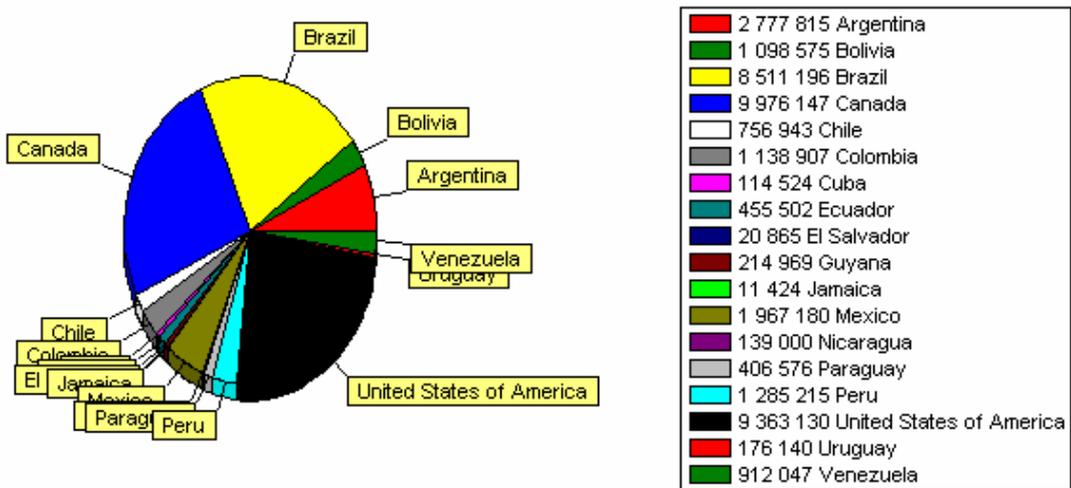
On the first activation, the editor window will appear as in the image shown above. The first thing to be done is to add one or several series (one series in our example). To perform this, click the  button and select the pie chart:



There are many different types of series available. After adding the series, the bar 3 becomes active. Here you specify, which data should be used for plotting. First of all, let us select the data set in the "DataSet" pulldown. Fill the "Label" and "Pie" fields using their respective pulldowns:



Click “OK” to close the editor and then preview the report:



What can be improved in this report? First of all, it would be nice to sort values in descending order. Again, we enter the chart editor and select the series in the upper part of

the window. Now we select the required sorting mode:



The screenshot shows a dialog box titled "Other options" with the following settings:

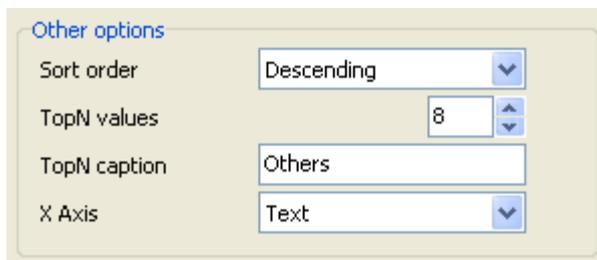
Sort order	Descending
TopN values	0
TopN caption	
X Axis	Text

If we previewed the report now, we would see that the data in the legend table is sorted.

8.1 Limitation of number of chart values

Our chart looks rather crowded, since there are too many small values in the chart, which are invisible anyway. FastReport allows limiting of the number of values displayed in a chart by a predefined value. Thus, all the values, which do not belong to the limit set, would be displayed as a single value, representing the sum of values, which did not fit the chart.

In our example, the chart has 18 values, and only 8 of them can be displayed. Let us enter the editor and set limiting:

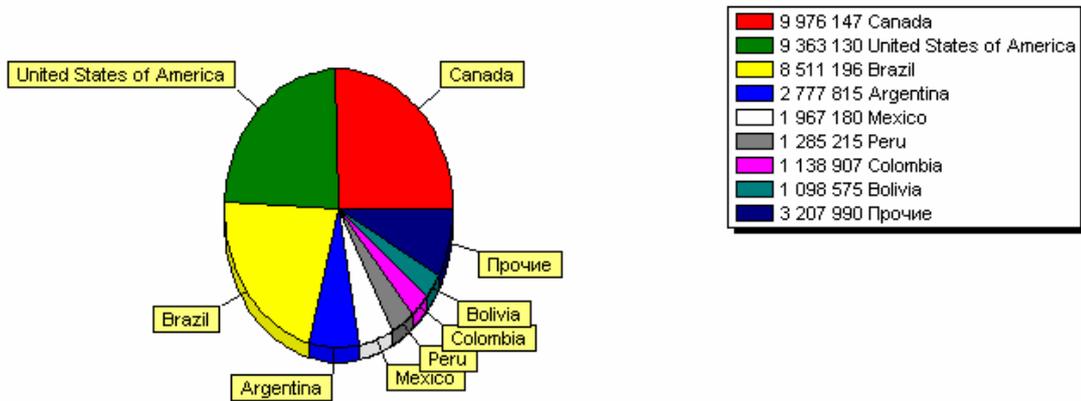


The screenshot shows the "Other options" dialog box with the following settings:

Sort order	Descending
TopN values	8
TopN caption	Others
X Axis	Text

The limiting will work if the "TopN" is not zero. The name in the "TopN title," which will be displayed opposite to the sum value, should be specified. Sorting mode is not significant; values will be sorted by default.

As a result, the report will look as follows:



8.2 Some useful settings

Let us examine several settings, which can be useful for setting chart appearance. These settings can be specified in the object inspector only.

The following basic properties are available when selecting a chart in the top of the list:

- Gradient – settings for gradient background filling. Enable the “Gradient.Visible” property for gradient displaying.
- Legend – settings for explanatory table appearance. The table can be disabled with the help of the “Legend.Visible” property. The table position is set with the help of the “Legend.Alignment” property.

The following properties are available when selecting a series:

- ColorEachPoint – color each value with different colors.
- ExplodeBiggest – select the largest value (only for the series of the “circle chart” type).
- Marks – settings for the explanatory hints appearance.
- ValueFormat – the line for formatting values.

It is necessary to note that all charting capabilities are accessible in the TeeChart Pro library (you can buy it separately from teechart.com). This library contains many types of charts and has convenient chart and series editor.

8.3 Chart with specified values

In the previous example, we constructed a chart on the basis of the DB table data. There is another way of constructing a chart: to enter the necessary data manually. This way is convenient when constructing small charts.

Let us demonstrate how it works with a simple example. Put a chart on the report design page and enter its editor. Add the series of the “Bar chart” type and set its properties:

Data Source

DataSet

Band source

Fixed data

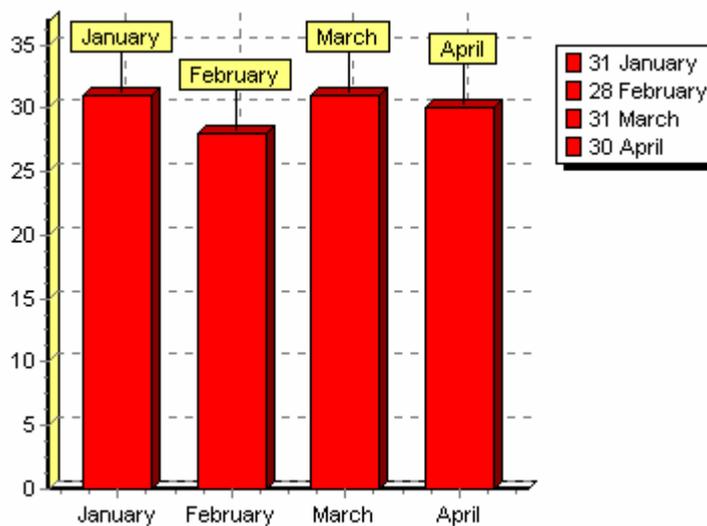
Values

Label: January;February;March;Apr

Y: 31;28;31;30

X (optional):

The result:



8.4 Chart completion from Script

Let us examine the last chart completion from script. To perform this leave empty XValues and YValues fields in chart editor. In report script write the following:

PascalScript:

```
begin
  Chart1.SeriesData[0].XSource := 'Jan;Feb;Mar;Apr';
  Chart1.SeriesData[0].YSource := '31;28;31;30';
end.
```

C++Script:

```
{  
  Chart1.SeriesData[0].XSource = "Jan;Feb;Mar;Apr";  
  Chart1.SeriesData[0].YSource = "31;28;31;30";  
}
```

SeriesData[0] in this case allows us to set parameters for the first series in the chart. If chart has several series, you can address them via SeriesData[data_number].

8.5 Printing of a chart built in Delphi

If you have already built a chart in Delphi code and want to print it in the report, you need a “Picture” object. Place it in the required place of report design page and write the following TfrxReport.OnBeforePrint event handler in Delphi code:

```
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;
```

where Picture1 – “Picture” object name, Chart1 = your Delphi chart.

Note: When you have external Delphi code assigned to the event handlers of the TfrxReport component you must run from the compiled exe. Not by previewing from within the report designer.

Chapter



IX

**Dot-Matrix
Reports**

Earlier we examined reports intended for printing with ordinary printers (stylus, laser, etc.). If sent to a dot-matrix printer their printing will be very slow. FastReport allows us to create special reports for dot-matrix printer where only standard font symbols without graphic elements are printed. That is why printing is rather fast.

Let us examine report building of "List" type which is intended for dot-matrix printing. Earlier we created such kind of report, see "List of clients" report". We need the same data for report.

So, create a new project in Delphi, place TTable, TfrxDBDataSet, TfrxReport and TfrxDotMatrixExport components on form and set their properties:

TTable:

DatabaseName = 'DBDEMOS'

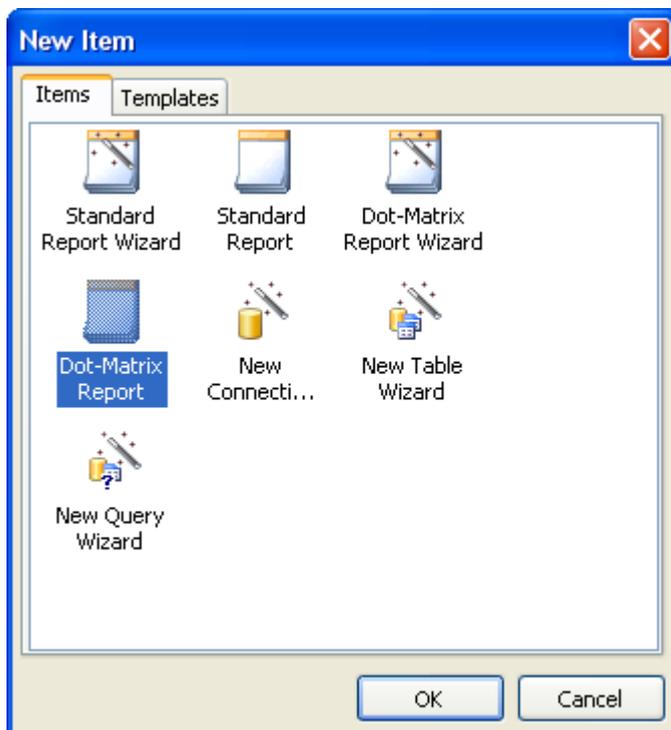
TableName = 'Customer.db'

TfrxDBDataSet:

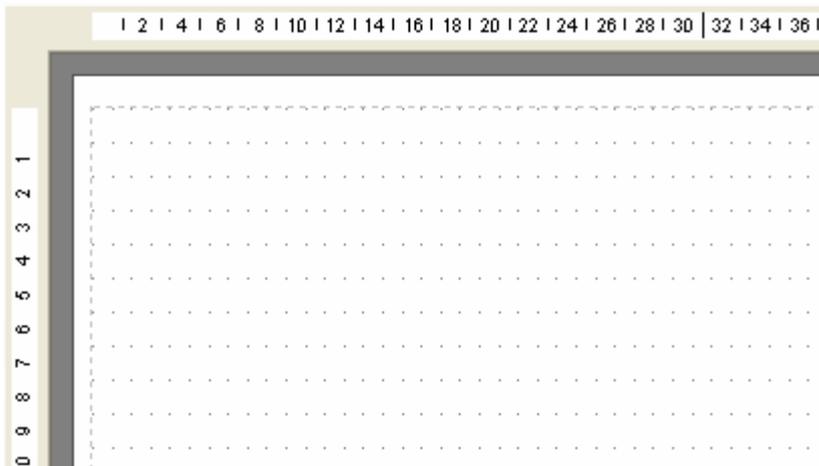
DataSet = Table1

UserName = 'Customers'

Enter report designer and select "File|New..." menu item. The report wizard dialogue appears with a report wizard list. Select the "dot-matrix report" item:



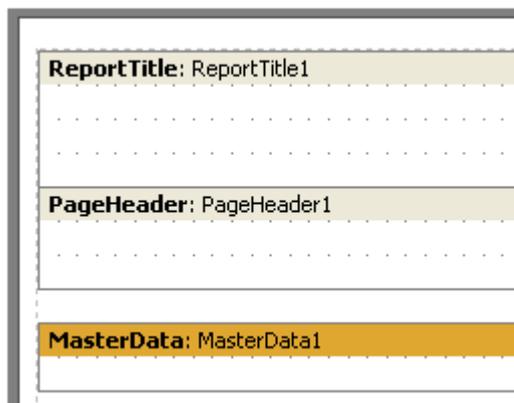
On clicking  you see empty design page layout marked for dot-matrix font:



The list of objects available for inserting has changed – now they are the “Band”, “Text”, “Line”, “ESC-Command”, “Subreport” and “Cross-tab” objects. Other objects cannot be used in dot-matrix printer.



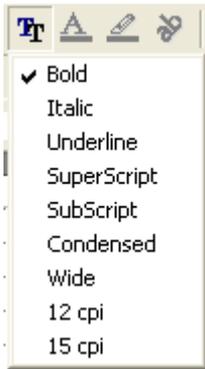
Place Report title, Page header and Master data bands on report page:



Place “Text” objects on bands in the following way:

ReportTitle: ReportTitle1	
Customer list	
PageHeader: PageHeader1	
Company	Address
MasterData: MasterData1	
[Customers."Company"]	[Customers."Addr1"]

Dot-matrix objects placing principle is the same as in ordinary report. Difference is in the fact that objects are strictly bound to netting, and it is impossible to set another font size or color for them. But some font attributes can be modified. To perform this select "Text" object and click "Tt" on toolbar:



As you can see, here you can set font attributes which are specific for dot-matrix printing. Report page and all dot-matrix objects with the exception of bands have these attributes.

Attention! In designer and preview only "Bold", "Italics", "Underline" attributes are displayed. The whole set of attributes is only printed.

Let us modify our report appearance with "Bold" style for headings. Report is ready, you can run preview mode:

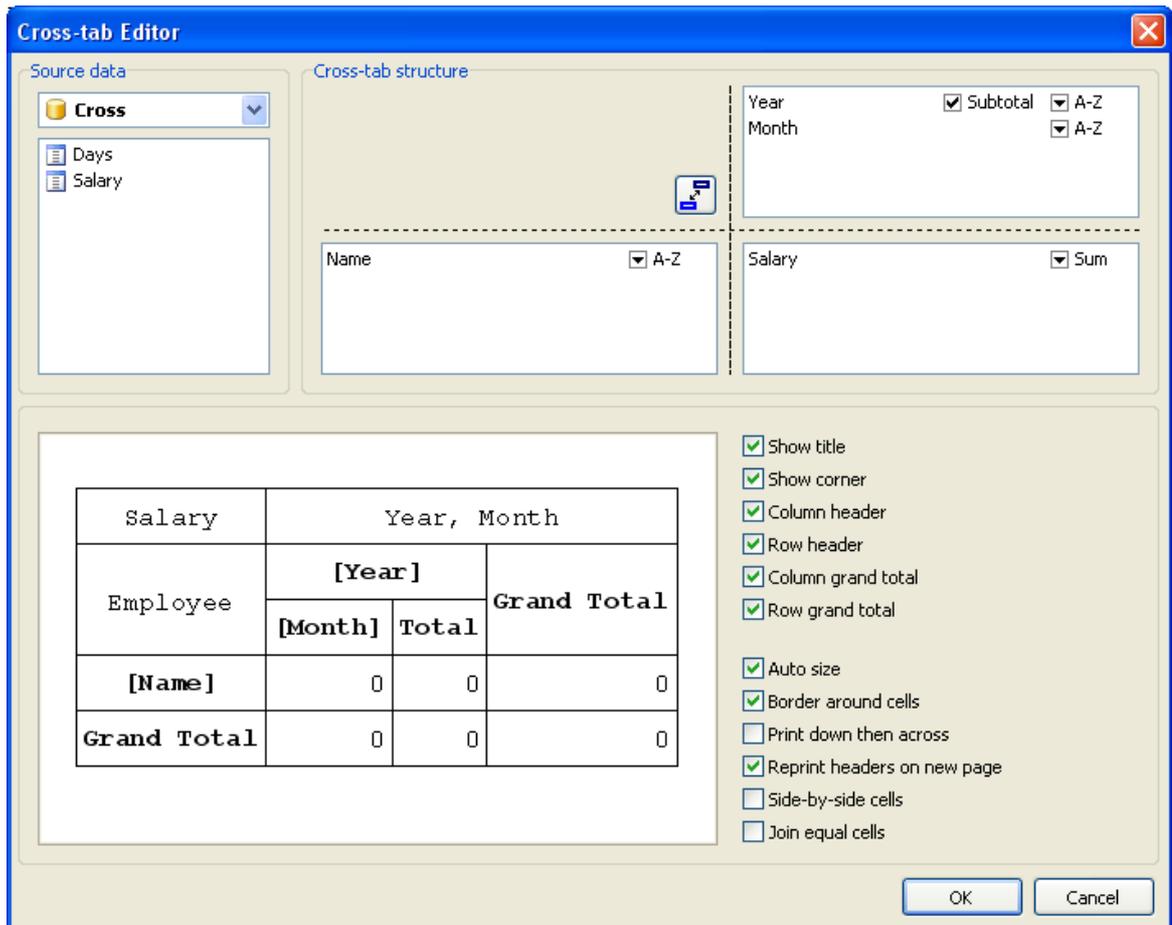
Customer list

Company	Address
Action Club	PO Box 5451-F
Action Diver Supply	Blue Spar Box #3
Adventure Undersea	PO Box 744
American SCUBA Supply	1739 Atlantic Avenue
Aquatic Drama	921 Everglades Way

9.1 Cross-tab in dot-matrix

The number of objects for dot-matrix report is restricted only by those which can be displayed in textual form. Among them there is “Cross-tab” object. Let us examine simple cross-report creation which is similar to one built earlier in “Table with composite headers” chapter.

For dot-matrix report creation perform the same steps like in the previous chapter – call “Empty dot-matrix report” wizard. Put “DB cross-table” component on report page and enter its editor:

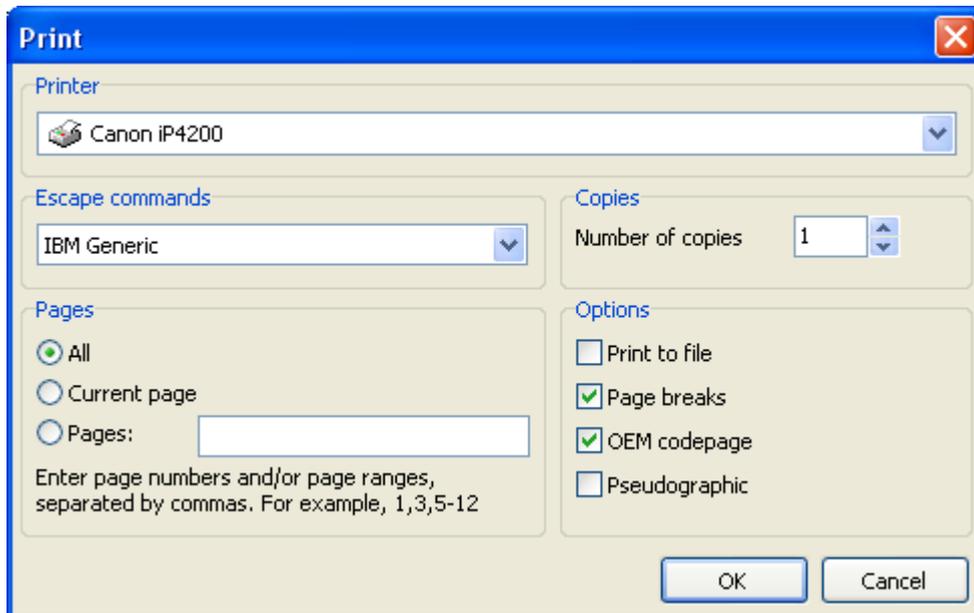


As can be seen, the editor shows structure of output table in dot-matrix mode. Cross cells style can be set via using “Tt” button in the toolbar. In all other respects working does not differ from the one earlier described. The previewed report will appear in the following way on the screen:

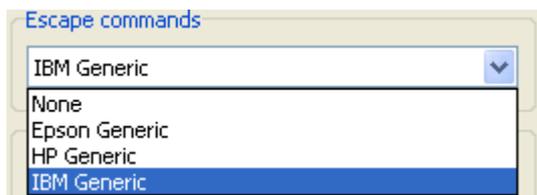
	1999					2000				2001		
	2	10	11	12	Total	1	2	3	Total	1	2	3
Ann	1000		1100	1200	3300	1300	1400		2700		1500	1600
Ben		2100	2200		4300		2400		2400			
Catherine		3000	3100		6100			3200	3200			
Den					0	3999			3999	4000	4100	
Grand Total	1000	5100	6400	1200	13700	5299	3800	3200	12299	4000	5600	1600

9.2 Dot-matrix reports printing

To print a dot-matrix report in text mode (i.e. with maximum speed) it is required to put TfrxDotMatrixExport  component on your project form from “FastReport 3.0” component palette. This component is charged with report converting to text form and further printing in text mode. At the same time it replaces standard printing dialogue:



Printing dialogue resembles a standard one, but dot-matrix printer specifics are added here. So, it is necessary to select system of printer commands before printing (ESC-commands). The following commands are available:

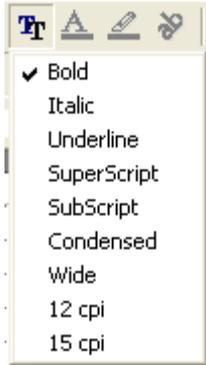


Also there is a set of flags setting options of dot-matrix printing:

- Print to file – defines whether it is necessary to send printing stream to file on hard disk. If flag is enabled, a window with file name query appears;
- Page breaks – defines whether it is necessary to send “Page break” control command on reaching page bottom. If the flag is disabled, it allows to print on roll stationery;
- -codepage – defines whether it is necessary to perform symbol conversion;
- Pseudographic – defines how to draw vertical and horizontal lines. If flag is disabled, lines are drawn with the help of -, |, + symbols.

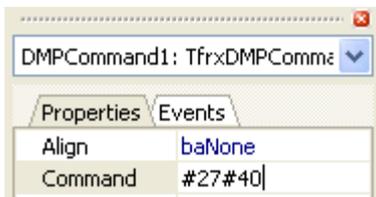
9.3 “Command” object

As it was described earlier, you can determine the following attributes set for dot-matrix report objects:



This is a standard set which is understood by all models of dot-matrix printers. Meanwhile, a specific printer model can support commands not present in standard set, for example, printing with 20 character per inch resolution. To send such a command on report printing use “ESC-Command” object `#1B`.

The object is placed in required place of page (for example, in top left corner or before objects group which is to be depicted with non-standard attributes). To set a command edit Command property of object (in object inspector):



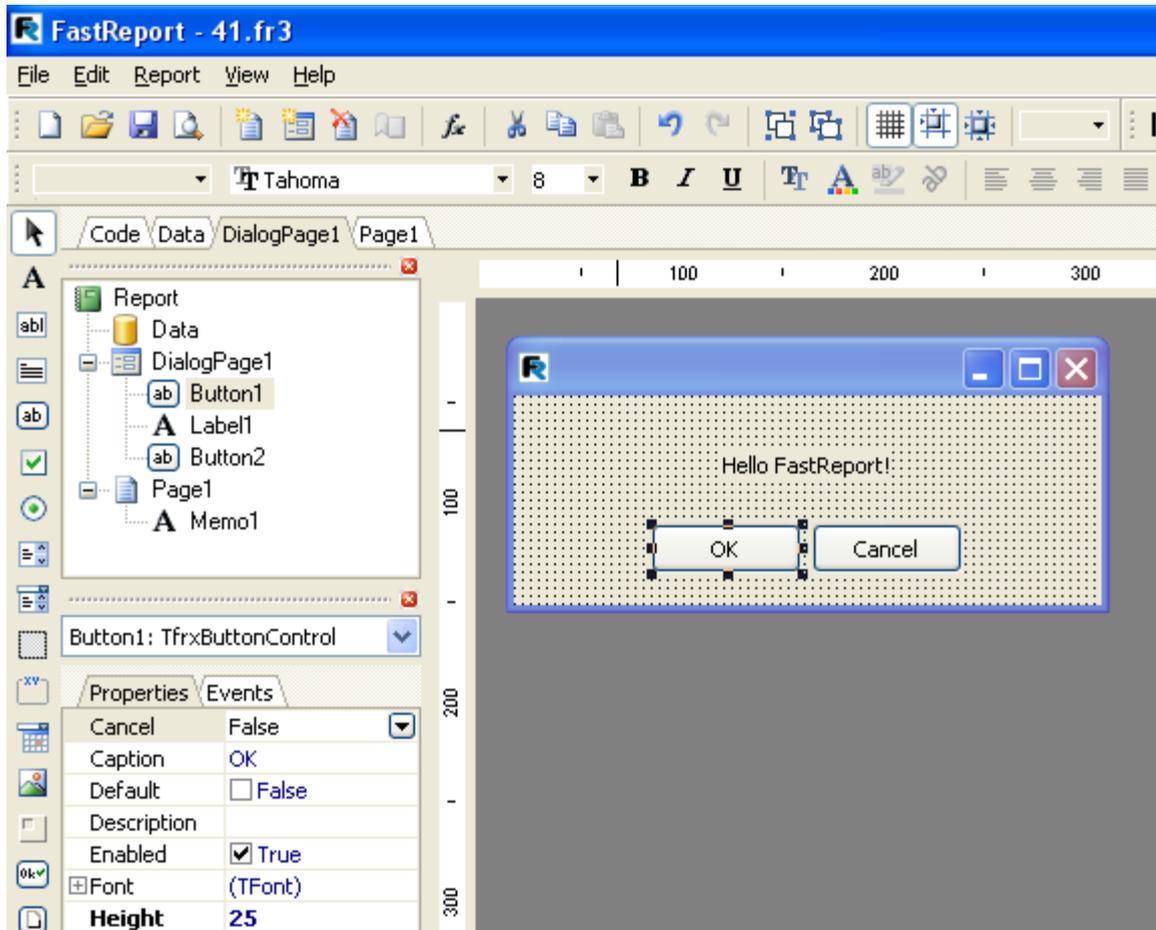
In the property you can set a command in one of the three forms: decimal (for example, #27#40) or hexadecimal (1B28).

Chapter



Dialogue forms

In addition to usual report design pages, you can use dialogue forms in a report. For dialogue form creation, the same designer as for report pages is used. The  button in the designer toolbar is used for creating a new dialogue form; it adds a new dialogue design page to a report. When switching to the page with the dialogue form, the designer workspace changes, thus becoming a form where control objects can be placed:



10.1 Controls

For Dialogue form controls use in a report the TfrxDialogControls  component from the Delphi FastReport component palette, should be added to the Delphi form in your project or add "frxDCtrl" into the "uses" list. The following controls will then be available for use in the reports:

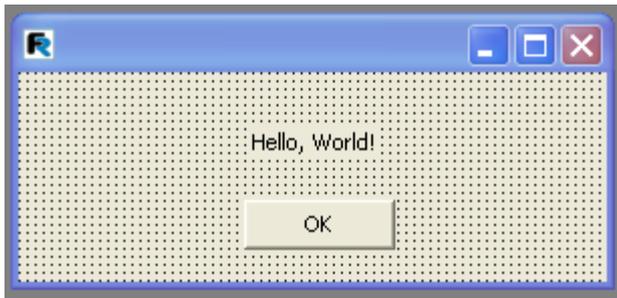
Element	Name	Description
	TfrxLabelControl	This control is used for displaying explicative inscription on the dialogue form.
	TfrxEditControl	This control is used for entering a text line with the help of the keyboard.
	TfrxMemoControl	This control is used for entering several text lines with the help of the keyboard.
	TfrxButtonControl	The control represents a button.
	TfrxCheckBoxControl	The control represents a flag, which can perform two statuses: enabled and disabled. Near the flag, the explicative inscription is displayed.
	TfrxRadioButtonControl	The control represents a switch key counterpart with radio button. This is the reason why it cannot be used alone.
	TfrxListBoxControl	The control represents the list of lines with a possibility to select one of them.
	TfrxComboBoxControl	The control represents the drop-out list of lines with a possibility to select one of them.
	TfrxDateEditControl	The control represents a field with a drop-out calendar for date entering.
	TfrxGroupBoxControl	The control represents a bar with explicative inscription which is used for uniting several controls.
	TfrxPanelControl	The control represents a bar, which is designed for uniting several controls.
	TfrxBitBtnControl	The control represents a button with picture.
	TfrxSpeedButtonControl	The control represents a button with picture.
	TfrxMaskEditControl	The control represents a text box for entering information set in a template.
	TfrxCheckListBoxControl	The control represents a list of lines with flags.

	TfrxBevelControl	The control is used for the dialogue form design.
	TfrxImageControl	The control represents a picture in “BMP,” “ICO,” “WMF,” or “EMF” format.

As you can see, all the controls are similar to those used in Delphi. In the FastReport component help, you can obtain help about the properties, events and methods of each control.

10.2 “Hello, World!” report

In this example, we will create a report displaying a greeting window before outputting the report by using a dialogue form. Create a new project in Delphi, and then put the “TfrxReport” and “TfrxDialogControls” components on the form. Call FastReport designer by double-clicking on the “TfrxReport” component and add a dialogue form into the report. Put the “TfrxLabelControl” and “TfrxButtonControl” objects on the form:



Set objects' properties:

TfrxLabelControl:

Caption = 'Hello, World!'

TfrxButtonControl:

Caption = 'OK'

Default = True

ModalResult = mrOk

Set the “BorderStyle = bsDialog” property in the form. As we can see, both the controls and the form have the same set of properties as those of the corresponding Delphi controls.

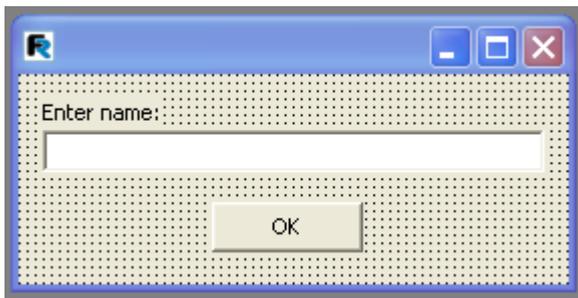
As soon as design of the dialogue form is finished, return to the report design page and place a “Text” object with some text in it there. Preview the report and you will see the dialogue form:



When clicking on the “OK” button, a report will be constructed and displayed. If closing a form via the “ ” button, the report will not be constructed. This is the mechanism of FastReport working: if there are dialogue forms in a report, it is constructed only when each form is closed with the “ ” button, i.e. it returns `ModalResult = mrOk`. That is why the “`ModalResult`” property of the button is set equal to “`mrOk`.”

10.3 Entering parameters and transferring them into a report

Let us make this example more complicated in order to show how to transfer the values entered in the dialogue form into a report. To perform this, modify the dialogue form in the following way:



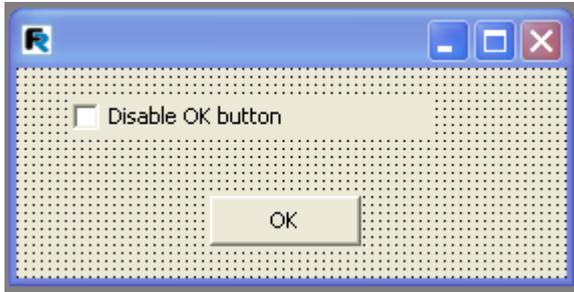
Place a “Text” object containing the following text on a page:

*You entered:
[Edit1.Text]*

Preview the report and make sure that the parameter you entered is successfully displayed in the report. You can address other objects of the dialogue form In the same way. Since each object has a name, which is unique within the whole report, it can be used anywhere within the report.

10.4 Interaction of controls

By using script, you can easily create logic for the dialogue form's work, for example, its controls' interaction. Let us illustrate this by a simple example. Modify the form in the following way:



Double click on the “CheckBox” object, so that the “OnClick” event handle would be created, and then write the following script:

PascalScript:

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

C++ Script:

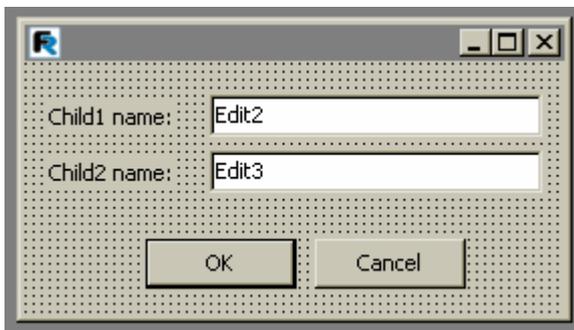
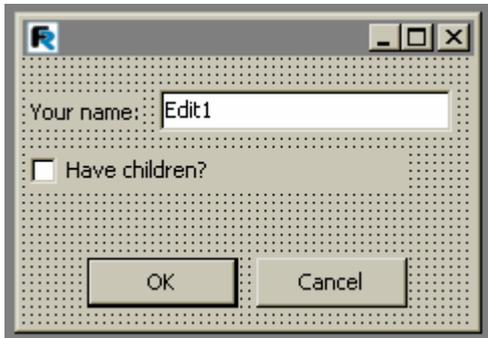
```
void CheckBox1OnClick(TfrxComponent Sender)  
{  
    Button1.Enabled = !CheckBox1.Checked;  
}
```

As you can see, the code does not differ much from what we use in Delphi. When running the report, you would see that the button responds to the flag condition's modification.

10.5 Several dialogue forms

Let us examine how report with two dialogue forms works. Create a report with two dialogues and one design page:

Name:	[Edit1.Text]
Child1 name:	[Edit2.Text]
Child2 name:	[Edit3.Text]



Set ModalResult properties of OK and Cancel buttons (mrOk and mrCancel accordingly). Now run the report. First of all we will be offered to answer questions from the first dialogue (name, are there any children), then, on clicking – from the second one (children's names). After clicking in the second dialogue the report will be built. In such a way works FastReport kernel – involving several dialogue boxes they appear in the order of their creation, moreover, every further dialogue will be displayed after clicking OK in the previous one (with ModalResult property = mrOk). If any dialogue were denied (via Cancel or cross on window heading), report building would stop.

10.6 Dialogue forms managing

In the previous example both dialogue forms are displayed irrespective of the fact whether we ticked "Have children" or not. Let us show how to hide the second dialogue in case when this flag is disabled. To perform this create OnClick handler of button on the first dialogue form (double-click on the button to create handler):

PascalScript:

```

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

```

C++Script:

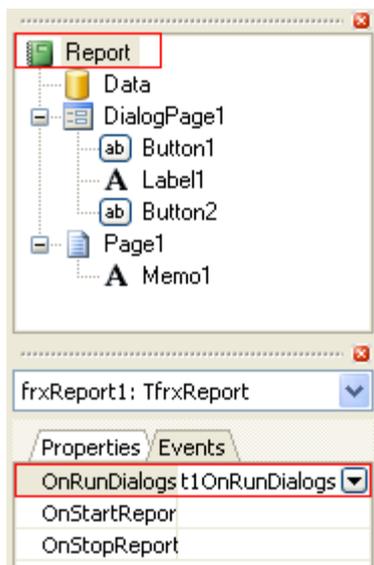
```

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

```

This code hides the second dialogue form (DialogPage2), if flag is not marked. Preview the report, to see that everything works correctly.

Another way of form managing is to use the OnRunDialogs report event. In order to create this event handler select Report object in report tree or object inspector and switch to “Events” tab in the inspector. Double-click on OnRunDialogs event to create necessary handler:



Write the following code in handler:

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;

```

```
end;  
end;
```

C++Script:

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

How the handler works we show the first dialogue. If it was closed via , look at CheckBox1 flag status and show the second dialogue, if it is necessary. If handler returns Result = True, report is building; if Result = False, report stops.

Chapter



**Data access
components**

Most reports, as a rule, are based on data from a DB. For accessing such data, Delphi offers effective mechanisms, which are used in FastReport. This matter concerns the “TTable” and “TQuery” components, which can act as data sources for the report. Generally, for this aim you can use any components, i.e. TDataSet successors.

In addition to accessing data defined in the Delphi project as we have done in our examples using the TfrxDBDataset, FastReport has available, several DB engine specific components for use within reports, which ones are dependant upon choices made during installation. In FastReport the principles for data access are much the same as those used in the Delphi environment. The same as in Delphi, a component is put on a dialogue form and its properties are set in the object inspector. Component ideology is very flexible: you can also create new components to support different data access engines easily (see the developers manual). They also, with the additional use of the TfrxDesigner , give the end user of the application the ability to design reports in runtime.

The screenshot shows the FastReport Designer interface for a report named "100.fr3". The interface includes a menu bar (File, Edit, Report, View, Help), a toolbar with various icons, and a main design area. On the left, a tree view shows the report structure: Report, Data, ADOTable1, ADOTable2, Page1, and Memo1. Below the tree view, the Properties window is open for ADOTable1, showing properties such as CloseDataSou (True), Database (ADOCConnection1), Description, FieldAliases (TStrings), Filter, Filtered (False), IndexFieldNanCompany, IndexName, Master (Not assigned), MasterFields, and Name (ADOTable1).

The main design area contains a diagram illustrating the relationship between two data tables. At the top, a text box reads: "Use the 'Text' object and 'Draw' category objects to draw a diagram." Below this, two boxes represent the tables:

- The "Customers" table** (ADOTable1): This table contains all information about the customer, such as company name, contact, phone, fax.
- The "Orders" table** (ADOTable2): This table contains all orders made by customers. Table is linked to the "Customers" table by master-detail relationship.

A yellow box highlights the relationship: "Orders.CustNo = Customers.CustNo". A yellow arrow points from the "Orders" table box up to the "Customers" table box, indicating the master-detail relationship.

11.1 Components' description

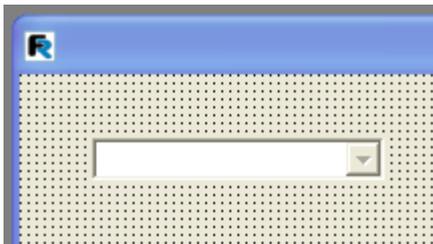
Let us examine usage of components for data access via ADO. They are available by adding the "TfrxADOCComponents"  component from the FastReport palette to the project. The following objects appear in the object toolbar when you switch to the "Data" page in the designer: "TfrxADOTable," "TfrxADOQuery," and "TfrxADODataBase." These components are similar to the corresponding Delphi components ("TADOTable," "TADOQuery," and "TADOCConnection") in terms of their functioning. Also you will be able to use the "TfrxDBLookupComboBox" control on a dialogue form.

Icon	Name	Description
	TfrxDBLookupComboBox	The control is used for selecting a value from a directory.
	TfrxBDETable	The control is used for access to DB table.
	TfrxBDEQuery	The control is used for performing SQL-query.
	TfrxBDEDataBase	The control is used for connecting to DB.

Let us examine each component.

11.1.1 TfrxDBLookupComboBox

This element is used for selecting a value in the directory table. It substitutes the directory identifier of the selected value.



The element has the following properties:

Property	Description
DataSet	Data source, which a control is connected to.
ListField	Name of the DB field, which will be displayed in a control.
KeyField	Name of the DB key field, which will identify the selected record.

KeyValue	Value of the DB key field, which was selected in the list.
Text	Value of the DB field displayed in the list.

For connecting of a control to the directory, you should fill values of the three properties: “DataSet,” “ListField,” and “KeyField.” The selected value is available via either the “Text” or “KeyValue” properties. You can set the initial position of a cursor in the list with the help of the “KeyValue.”

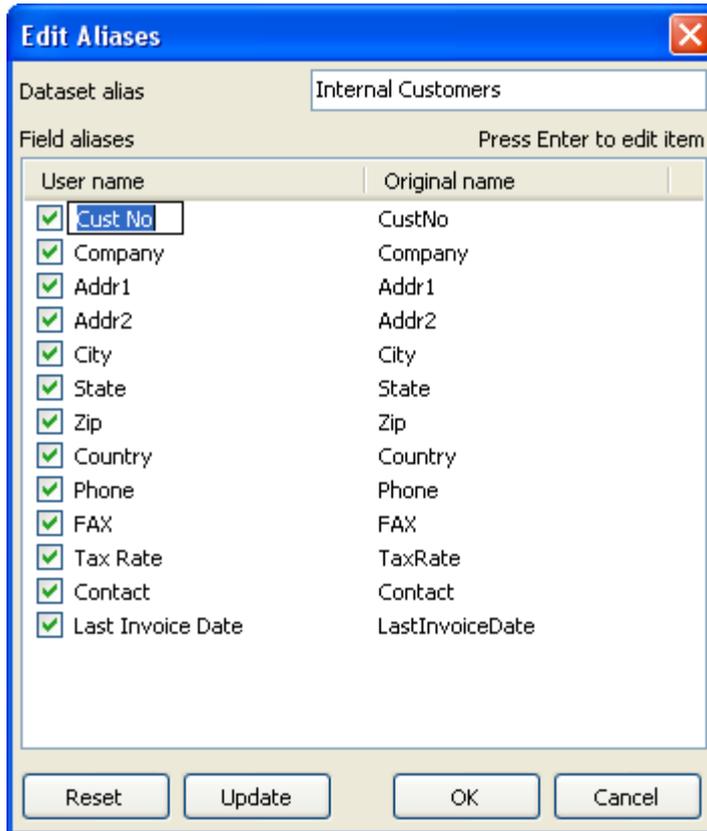
11.1.2 TfrxADOTable

The component is used for organization of DB table access. The component has the following properties:

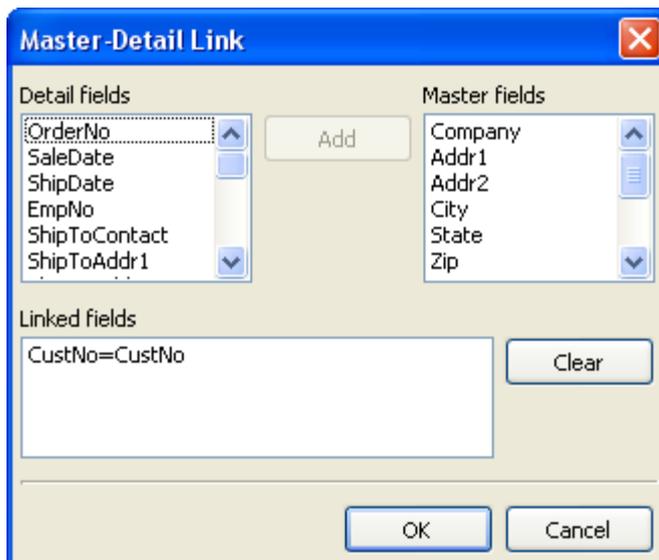
Property	Description
Active	Defines whether a table is active.
DatabaseName	Connection name (name of the TfrxADODatabase component).
FieldAliases	Enables to set fields aliases.
Filter	Expression for records' filtering.
Filtered	Defines whether it is necessary to use filter.
IndexFieldNames	Names of index fields.
IndexName	Secondary index name.
MasterFields	Fields connected with master dataset.
Master	Master dataset.
TableName	DB table name.
UserName	User name (alias) of the dataset.

Component properties' functions are similar to the “TADOTable” Delphi properties. To connect a component to the DB table, it is enough to fill the “DatabaseName” and “TableName” properties. Table opening is performed either via the “Active: = True” setting, or with the help of the “Open” method.

The “FieldAliases” property editor allows to select fields, which will be available upon addressing the table, and to set aliases for the whole table and for each field.



The “MasterFields” property editor is used for creation of master-detail connections between two tables. To connect two tables with the master-detail relation, a user should specify a general table in the “Master” property and call the “MasterFields” property editor for the subordinate table. If the table has secondary indexes, which are necessary to be used, set the “IndexName” property beforehand.



Here you can visually bind the “master” and the “detail” fields of data sets. When the sets’ connection is of “Master-Detail” type, then when moving within the master set, the contents of the detail set is filtered in a way that it contains only records belonging to the current record of the master set.

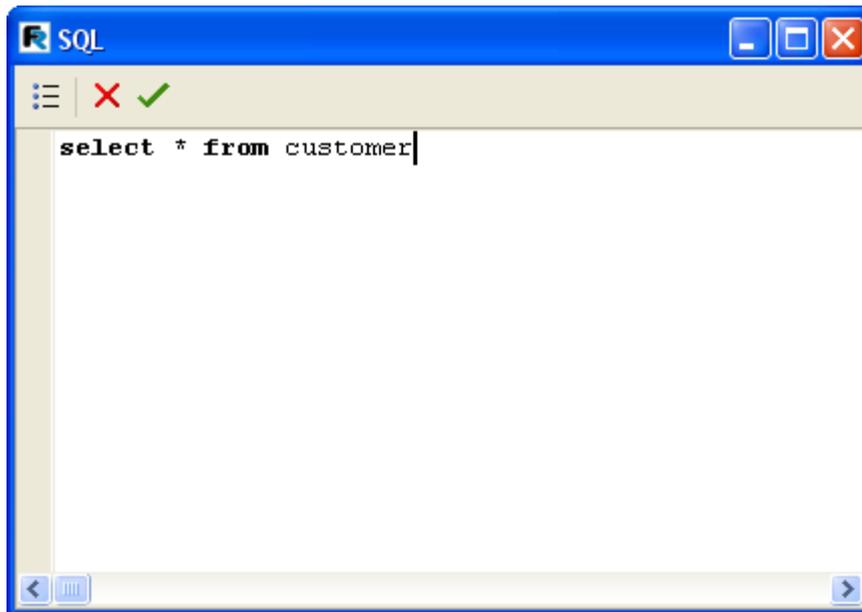
To connect the sets’ fields, select a field from the list on the left (detail set), then a field from the list on the right (master set), and click on the “Add” button. Thus, the fields’ bond would be transferred to the bottom list. To clear the bottom list, use the “Clear” button. The bound fields must be of an equal type and be the key ones.

11.1.3 TfrxADOQuery

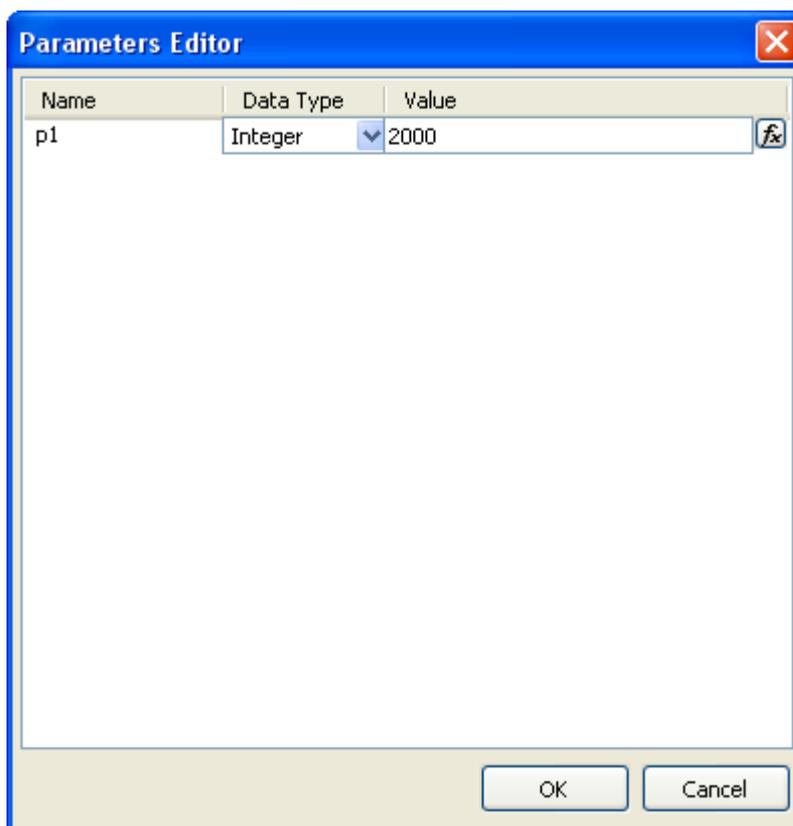
The component is used for performing SQL-queries to DB. The component has the following properties:

Property	Description
Active	Defines whether a query is active.
DatabaseName	Connection name (name of the TfrxADODatabase component).
FieldAliases	Allows to set user’s field aliases.
Filter	Expression for records’ filtering.
Filtered	Defines whether it is necessary to use the filter.
Master	Master dataset.
Params	The list of query parameters.
SQL	Query text.
UserName	User name (alias) of the dataset.

The “Active,” “DatabaseName,” “FieldAliases,” “Filter,” “Filtered,” and “Master” properties are similar to the properties of the “TfrxADOTable” component described above. The “SQL” property has its own editor for filling the SQL-query.



The “Params” property also has its editor. It becomes available as soon as a query text contains parameters.



A parameter can be of two types: either one assigned from the master-source or one having a concrete value (either an absolute symbol or a link to the variable or object's

property, as it is shown in the illustration above, can act as a value).

In case when a parameter is taken from the data master-set, it is necessary to adjust the “TfrxADOQuery.Master” property. The data set must contain a field with the name coinciding with the name of the parameter. At the same time, it is not necessary to specify either a parameter type, or its value.

11.1.4 TfrxADODataBase

This component is used to connect to a database. Its function is similar to the “TADOConnection” Delphi component. The component has the following properties:

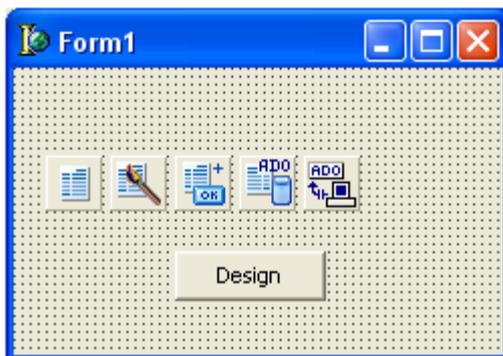
Property	Description
Connected	If “True,” it activates the connection.
DatabaseName	The ADO connection string.
LoginPrompt	Defines whether it is necessary to request a password upon connection to DB.

The LoginPrompt property defines whether it is necessary to request a password when connecting to DB. If “LoginPrompt” = “False,” a user name and a password must be specified in connection parameters.

11.2 Report constructing

Let us examine design of a simple report using data access components at runtime. We will use the demo database comes with FastReport - {FR}\Demos\Main\demo.mdb - as data for the example.

Create a new Delphi project which we will use for experimentation. Add the “TfrxReport,” “TfrxDesigner,” “TfrxDialogControls,” “TfrxADOCcomponents”, “TADOConnection” and “TButton” components on the form.



Setup the database connection. To do this, doubleclick on TADOConnection, choose "Build connection string", then choose the provider ("Microsoft Jet 4.0 OLE DB Provider") and choose our database (demo.mdb). Close connection dialog with OK button and set the components' properties:

ADOConnection1:
LoginPrompt = False

frxADOComponents1:
DefaultDatabase = ADOConnection1

Define the following handler for the "Design" button:

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

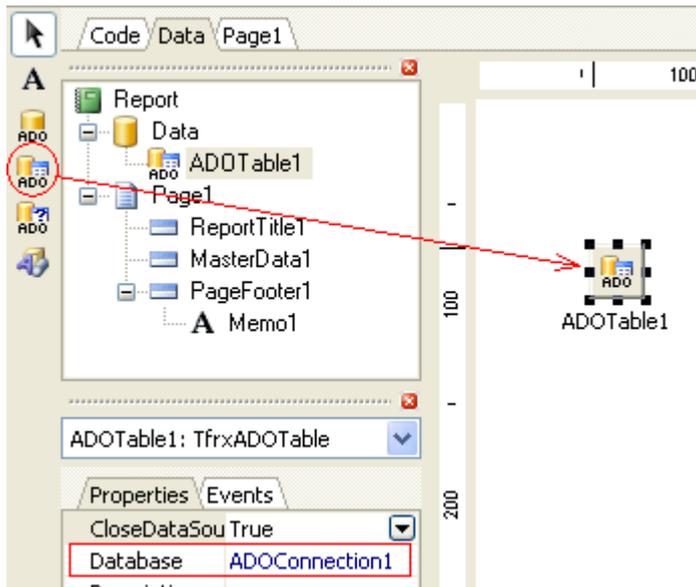
After that, compile and run the project. This is all you need for creation of the end-user runtime reports designer.

On clicking on the "Design" button, the designer, which contains a blank report, opens. Let us examine designing of simple reports in this environment.

11.3 Simple report of the "List" type

This report will contain data from one DB table. To construct a report, perform the following steps.

Click on the "New report" button  in the designer toolbar. FastReport will create an empty report containing "Code", "Data" and "Page1" pages. Switch to the "Data" page and put "ADO Table" component on page:

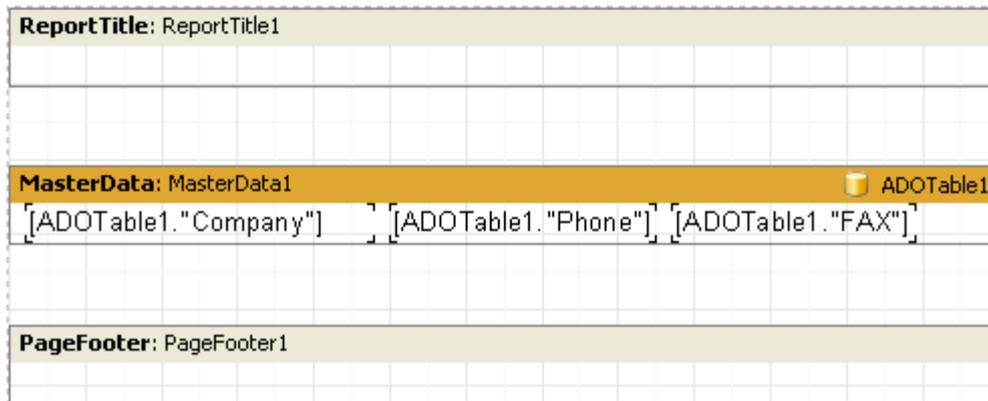


Pay attention to the "Database" property - it is already connected to our database. It happens because we have connected our database to the TfrxADOComponents.DefaultDatabase property. We have to choose the table name now:

TableName = 'Customer'

Go to the page with the report form. To connect the "Master data" band to the table, double-click on it, and then select the required table in the opened window.

Drag the required fields from the "Data tree" window to the report page. After that, the report will look roughly like this:



To preview the report, click on the "Preview" button  in the toolbar.

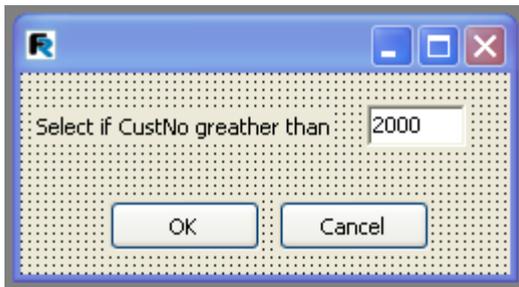
11.4 Report with parameters' query

Let us design a more complicated report, in which parameters would be requested in the dialogue window before the report begins to be output. To do this, using the same project in the report design window, click the new report button to clear the old one.

Switch to the "Data" page and put the "ADO Query" component on a page. Doubleclick it to call its editor and write the following SQL text:

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

Add a dialogue form into the report. Put the "Label," "Edit," 2 "Button," components on the report dialog form:



Set the components' properties:

Label1:

Caption = 'Select if CustNo greater than'

Edit1:

Text = '2000'

Button1:

Caption = 'OK'

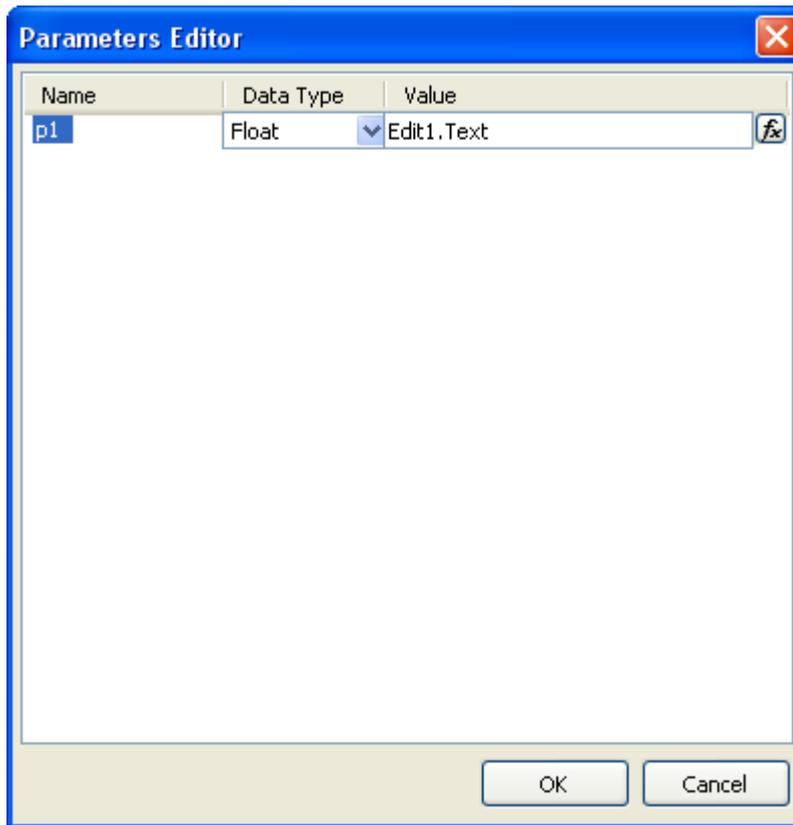
ModalResult = mrOk

Button2:

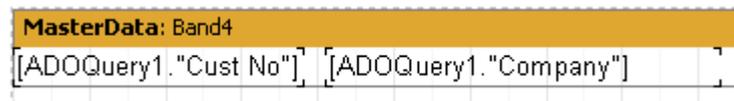
Caption = 'Cancel'

ModalResult = mrCancel

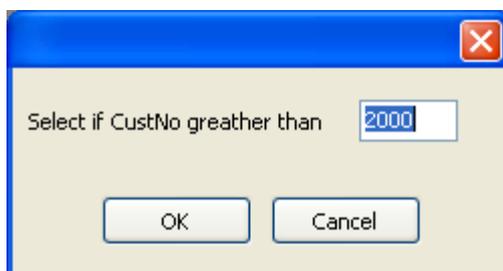
Open the "Params" property's editor of the "Query" component, and then set the parameter:



After that, go to the report design page and create the report as we did in the previous example:

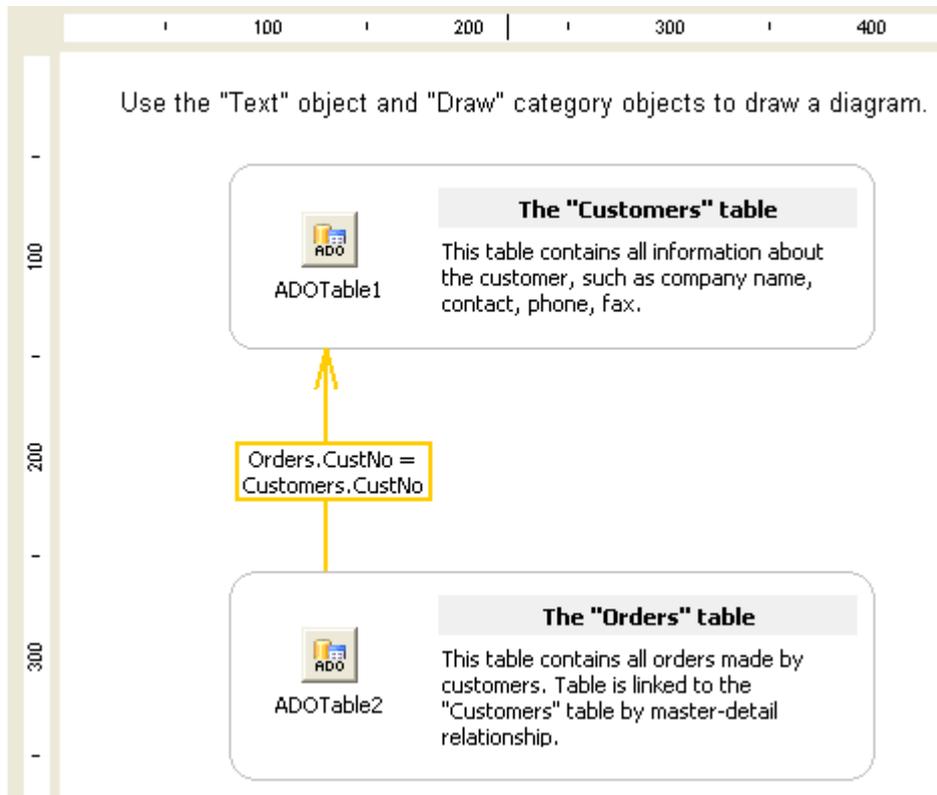


Upon previewing the report, the dialogue, in which a user will be prompted to enter a customer number, will be displayed. After entering a requested value and clicking on the "OK" button, the report's building is completed. The customers with numbers larger than the entered one will be displayed.



11.5 Other useful settings

You can put the "Text" and "Draw" elements on the "Data" page. Using such elements, you can draw simple diagrams like this:



Chapter

XII

**Report
inheritance**

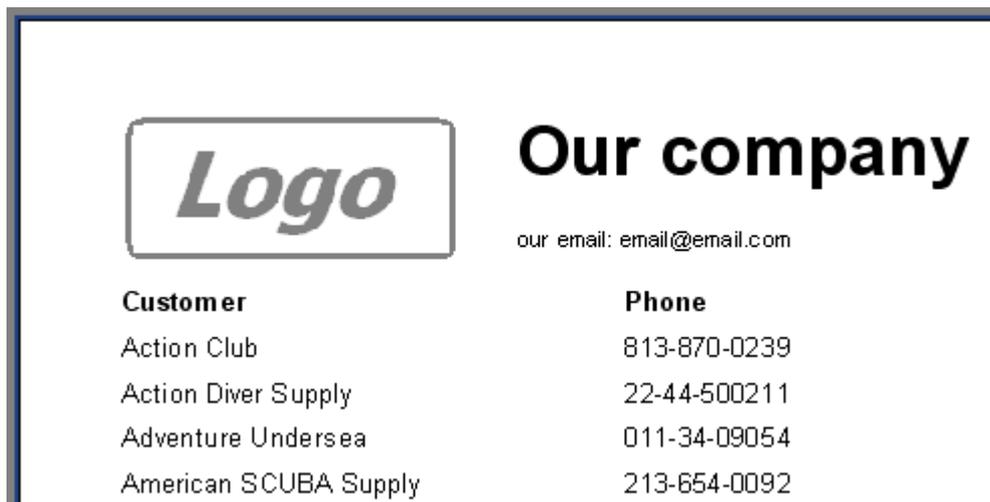
Often we have many reports with the same data in it - for example, same header/footer with company logo and some data - email, address etc. Now imagine the situation that you need to change some company data - for example, email. You have to do this in each report! To avoid this, you can use report inheritance. What is it?

For example, you have some common elements in each report (logo, company name, email etc). These elements are typically placed on the report title and/or page header. You can create a *base report* that contains only common elements. All other reports will use base report and thus will contain such common elements plus own elements defined in a report.

In case you need to change something (logo, email) you just open the base report and make necessary changes. All other reports that inherit from a base, will be changed automatically. In fact, when you open a report that is inherited one, the base report is opened first, then the inherited one.

12.1 Creating a report

Let's create a simple report that uses inheritance. Our report should look like this:



The image shows a report layout with a blue border. On the left, there is a rounded rectangle containing the word "Logo" in a stylized font. To the right of the logo, the text "Our company" is displayed in a large, bold font. Below this, the email address "our email: email@email.com" is shown. At the bottom, there is a table with two columns: "Customer" and "Phone".

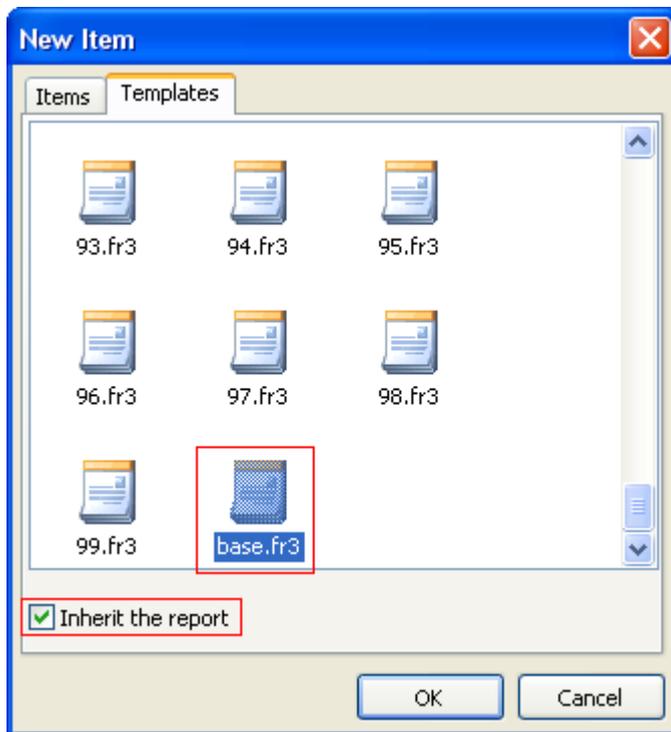
Customer	Phone
Action Club	813-870-0239
Action Diver Supply	22-44-500211
Adventure Undersea	011-34-09054
American SCUBA Supply	213-654-0092

At first you have to create a base report. Which elements must it contain? It's a logo bitmap, "Our company" title and email. Let's create a new report and place necessary objects into it:

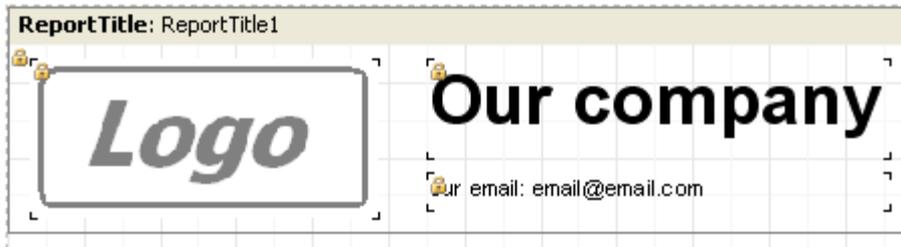


Save our report with "base.fr3" name. Into which folder? It depends on how you setup the TfrxDesigner component. By default FastReport will search base reports in the folder that contains your application's .exe file. You can setup a folder name for templates in the TfrxDesigner.TemplateDir property.

Now create inherited report. To do this, go "File" menu and choose "New...". In the dialogue select the "Templates" tab, search for our base report ("base.fr3") and click "Inherit the report" checkbox:



FastReport will create a report that contains all objects from a base report. They are marked by "lock" sign:



What this means. You cannot rename or delete such objects. You cannot move them to another band. All other settings (such as text, color, frame) can be made. Remember if you change some property (for example, color) of the base object, it will be stored in the inherited report. If you then try to change the color of this object in a base report, this setting will be ignored in the inherited one. For example: open the inherited report, change the "our company" object's color to red. Save the report. Now open the base report and set the color to green for "company". If you open the inherited report now, you will see the color is still red. So if you want to change some property of object with "lock" sign, it is preferred to do in the base report.

Let's finish with our report. All we have to do is add a page header and master data:



The report is ready.

12.2 Changing a base report

Let's look at situation when you need to change a base report. Open the base report ("base.fr3" in our example) and change necessary fields. Let's modify the email:



Save the report. Now open the inherited report and see that email is changed in this report as well:

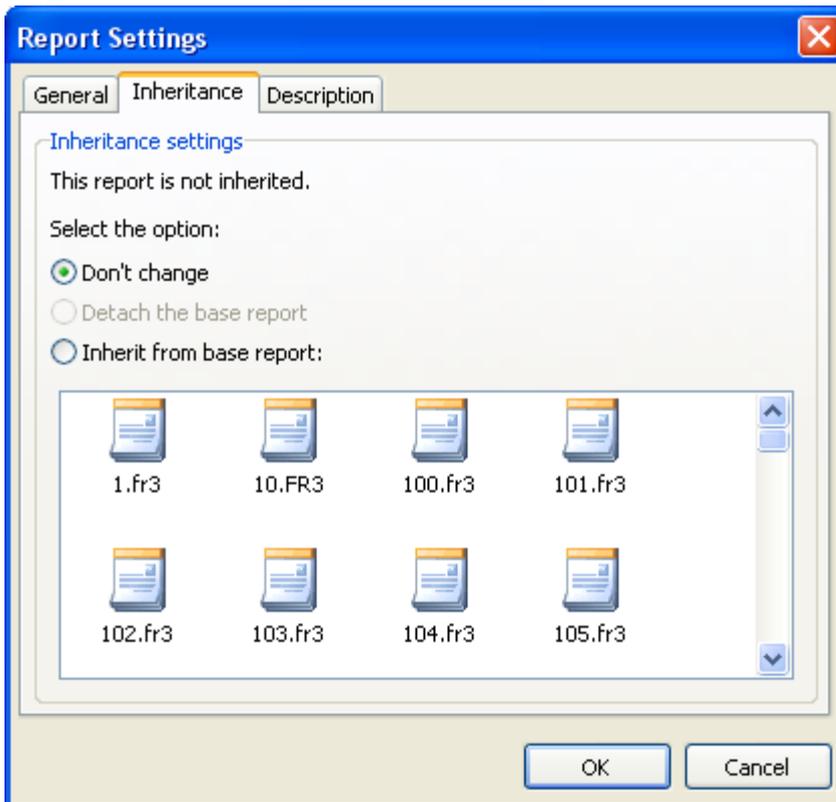


What if we need to add some objects in a base report? It's easy but remember: **the base and inherited reports can't contain objects with same names**. While changing the base report, we can't know how many reports already use this base one and what object names they have. So the common rule is: **if you add an object to the base report, give the following name to the object: ReportName_ObjectName**. For example, add a "Text" object to our report and set its name to BaseMemo3.

There is no restrictions on deleting objects from a base report or moving them.

12.3 Inheritance control

We have observed the inherited report creation from scratch. What if we have a report already which we need to make inherited? To do this, open the report and go "Report|Options..." menu. Choose the "Inheritance" tab:



We have to choose the "Inherit from base report" option and select the base report from a list. After this, press OK button. FastReport will join two reports. You may get the following error message:



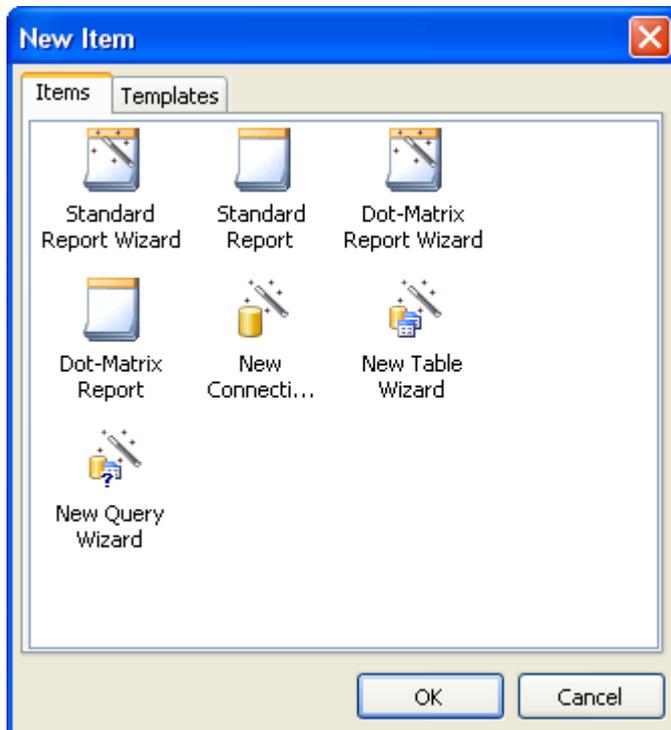
This may happen if two reports have objects with the same name. You can delete duplicate objects from a report or rename them.

Chapter



Wizards

FastReport contains several wizards that simplify the report creation process. Wizards can be found in the "File|New..." menu item.



13.1 New report wizard

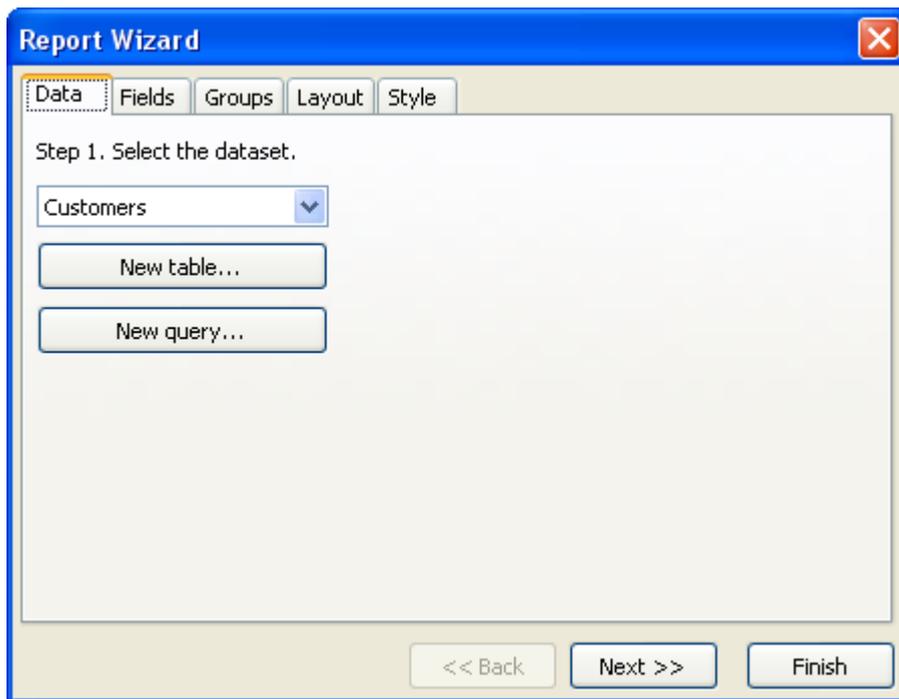
There are four wizards intended for creation of new report:

- Standard report wizard
- Dot-matrix report wizard
- Standard report
- Dot-matrix report

Wizards of type "Standard report" and "Dot-matrix report" will create the empty standard or dot-matrix report (you can read more about dot-matrix reports in the next chapter). The report will contain one empty page.

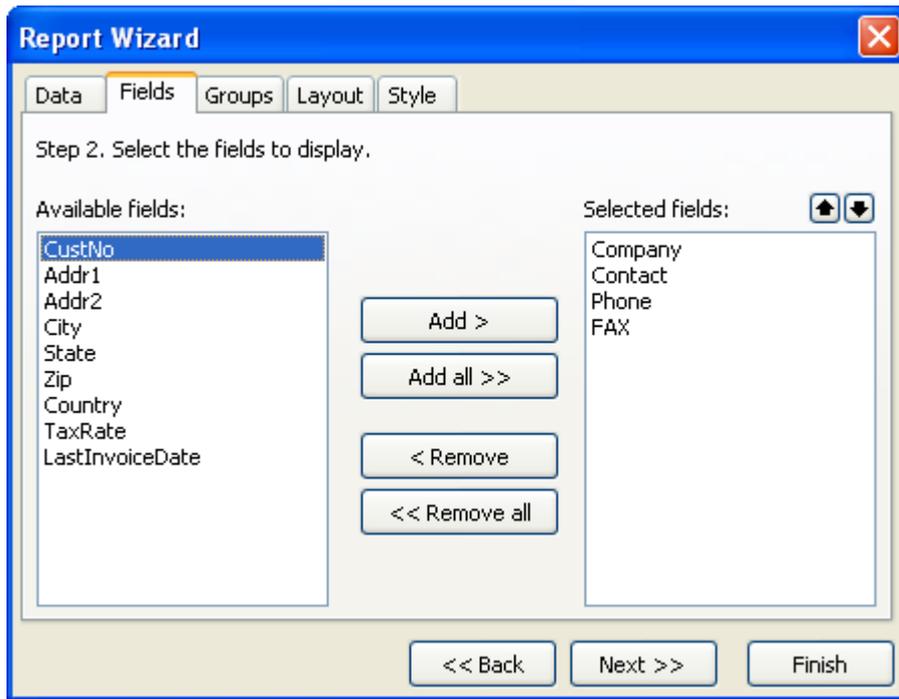
Wizards of type "Standard report wizard" and "Dot-matrix report wizard" allow you to choose fields you want to show in the report, create optional groups and select the data layout. Let's look at report creation process using "Standard report wizard".

Choose the "File|New..." menu item, then choose "Standard report wizard" item. We will see the report wizard dialog:



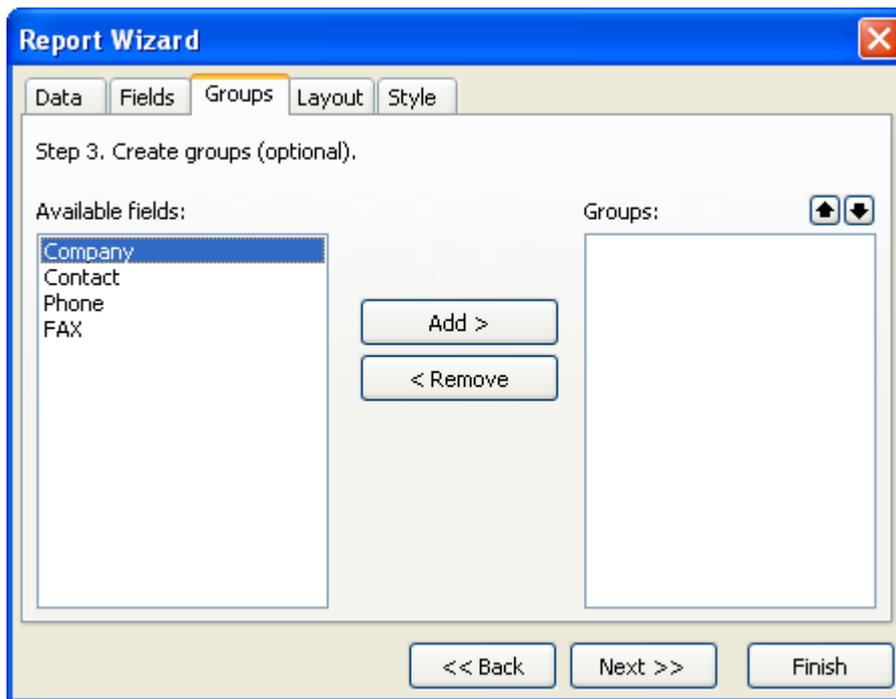
As you can see, there are several tabs in this window. On the first tab we need to choose the data source for our report. All data sources available in your application are listed here (TfrxDBDataSet components). You can also create a new data source - either table or a query - using the "New table" or "New query" buttons. In this case the "New table/query" wizard will be displayed (it is described later in this chapter). Let's choose the Customers table and press the "Next >>" button.

On the next tab we need to choose the fields we want to display in our report:



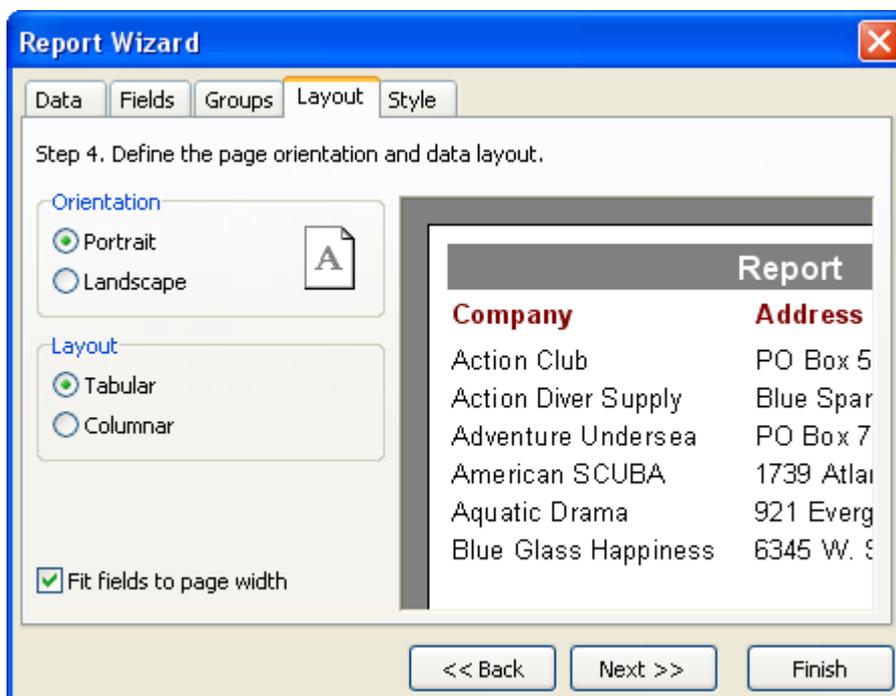
In the left side we can see a list of available fields; in the right side - list of selected fields, it will be displayed in the report. Use "Add", "Add all", "Remove", "Remove all" buttons to move necessary fields from one list to another. You can also use   buttons to move the selected field up or down. Let's add "Company", "Contact", "Phone", "FAX" fields to the selected fields list and press the "Next >>" button.

On the next tab we can create one or several groups. In this case FastReport will add the Group header, Group footer bands in our report.



This step is optional. We will skip it by pressing the "Next >>" button.

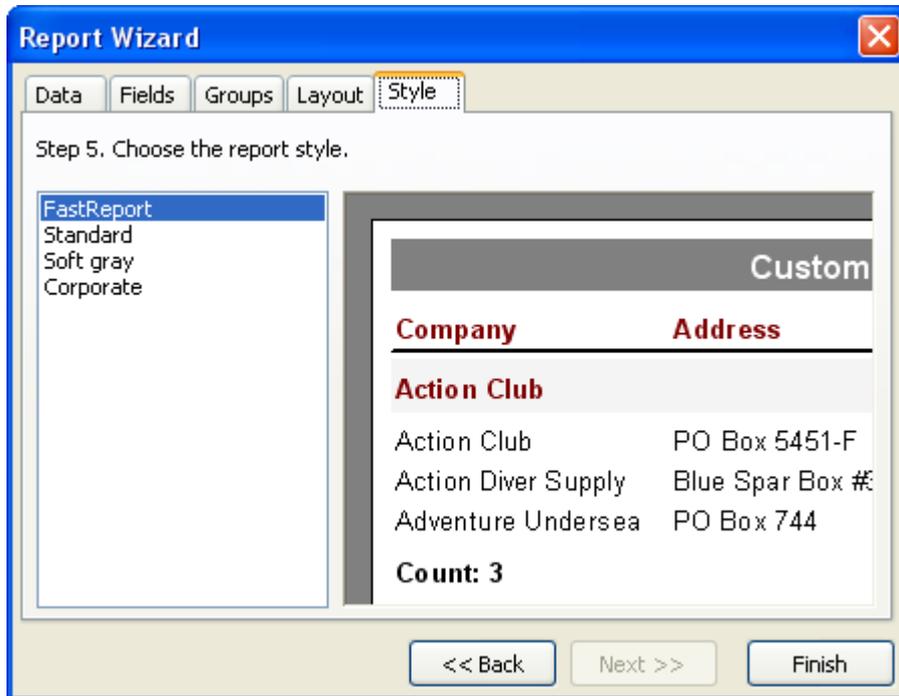
On the next tab we can set the page orientation and choose between two data layouts - tabular and columnar:



When choosing the layout we can see the report sample at right side of the

window.

Finally, the last tab displays the available color schemes for your report.



When we press the "Finish" button the wizard creates the following report:

ReportTitle: ReportTitle1			
Report			
PageHeader: PageHeader1			
Company	Contact	Phone	FAX
MasterData: MasterData1 Customers			
[Customers."Company"]	[Customers."Contact"]	[Customers."Phon	[Customers."FAX"]
PageFooter: PageFooter1			
			Page

We can run the preview immediately.

13.2 New connection wizard

This wizard allows you to add a new database connection into existing report. It may be necessary to have two or more connections if you want to display a data from two or more databases. The wizard will add the database component (like TfrxADODatabase) to your report.

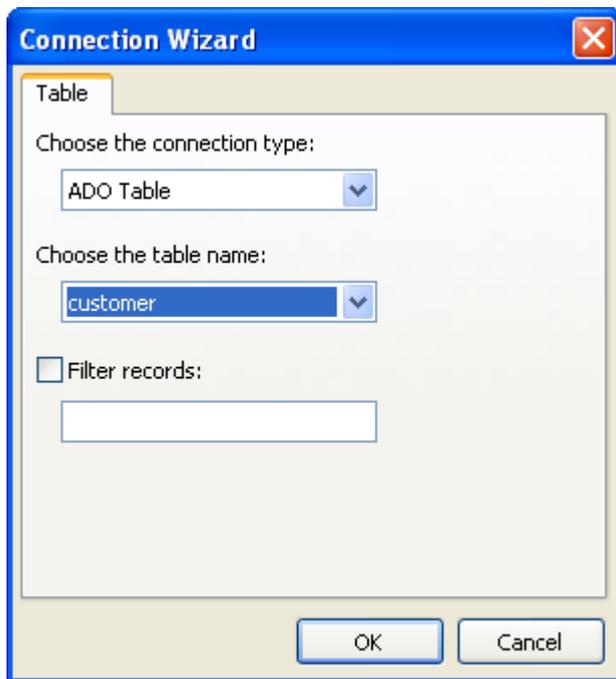


You need to construct the connection string (use the  button) - you will see the standard Windows connection window where you can choose the database and set the connection parameters. After this you may specify the user name/password.

NB: you can create a new connection manually - just put the TfrxADODatabase component into your report.

13.3 New table wizard

This wizard allows you to add a new database table into existing report.



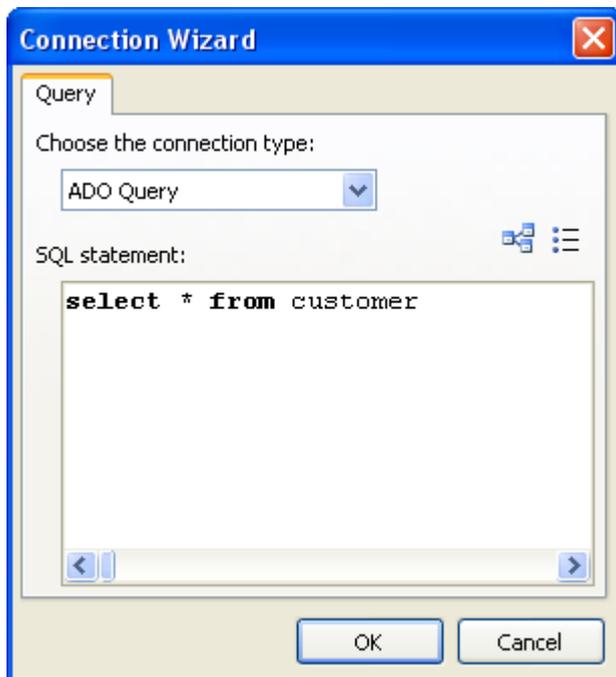
You need to select the table name. Also you can specify the filter if you want to filter a table records, for example:

(CustNo > 2000) and (CustNo < 3000)

NB: you can create a new table manually - just put the TfrxADOTable component into your report.

13.4 New query wizard

This wizard allows you to add a new SQL query into existing report.

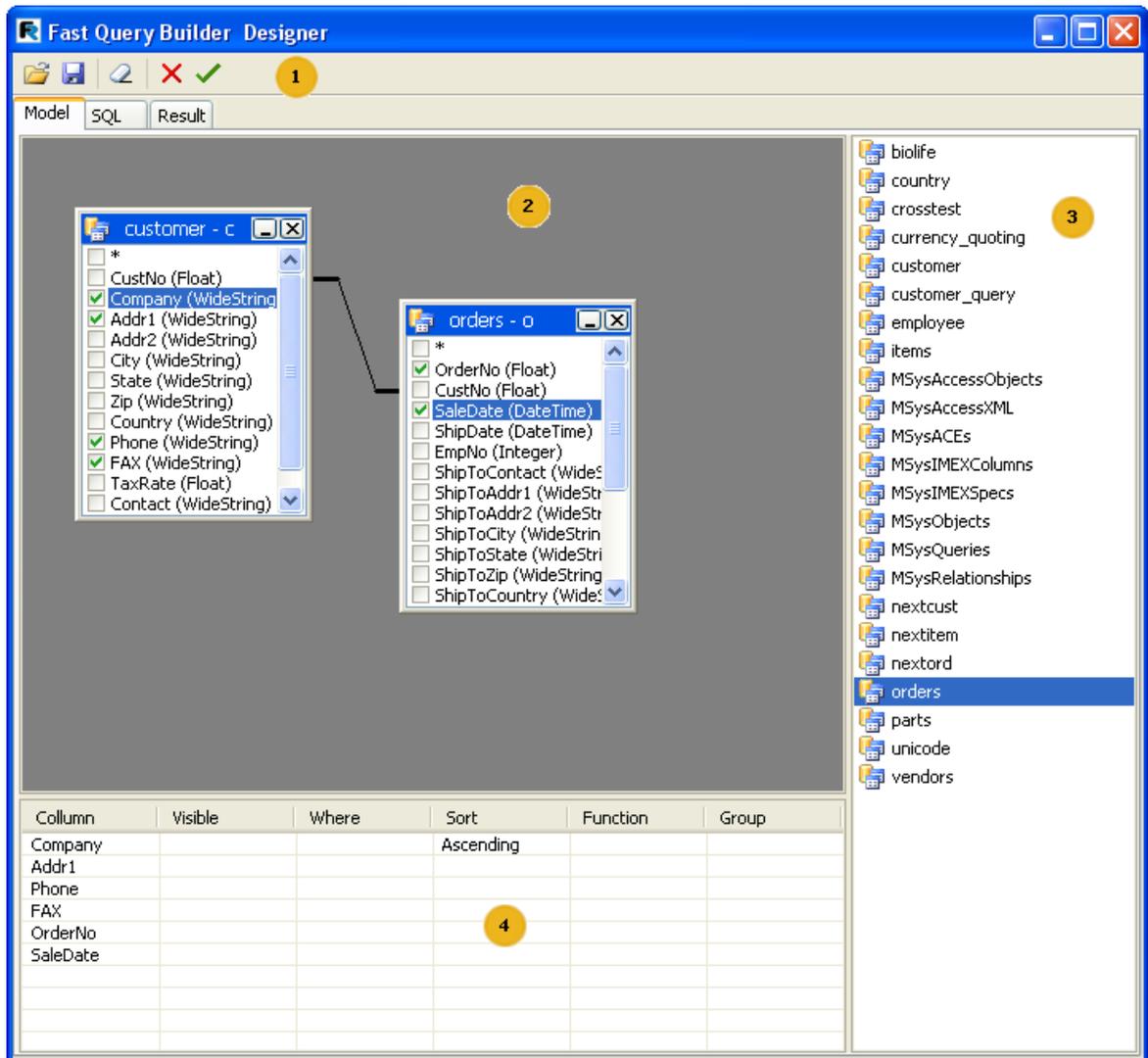


You need to specify the SQL here. You can use visual query builder to do this - push the  button. The query builder is described later in this chapter.

NB: you can create a new query manually - just put the TfrxADOQuery component into your report.

13.5 Query construction

For Visual query construction FastQueryBuilder is used. (It is available as an independent product for use in your applications). It is included in FastReport Professional and Enterprise versions. The query constructor is used for visual query building in SQL language. Constructor is shown in the illustration below:



- 1 - toolbar
- 2 – designer work area
- 3 – the list of available tables
- 4 – selected table fields parameters area

Toolbar:

- open SQL file
- save query into the file (query plan is also saved into the file)
- designer working area clearing
- button. Exiting designer with saving.
- Cancel button. Exiting designer without saving.

Constructor working area and the list of available tables support Drag&Drop technology, i.e. for placing table into working area drag it there with the mouse, or double-click on table title in the available list of tables.

To include any field from the table in the query mark it:



Marked fields appear in the fields parameter table area (4):

Column	Visible	Where	Sort	Function	Group
Company	<input checked="" type="checkbox"/>				
Phone	<input checked="" type="checkbox"/>		No		
FAX	<input checked="" type="checkbox"/>		Ascending		
OrderNo	<input type="checkbox"/>		Descending		
SaleDate	<input type="checkbox"/>				

- Visibility – defines whether field is included in output
- Where – field selecting condition. For example, '> 5'
- Sort – defines sorting according to field.
- Function – defines function applicable to field
- Group – grouping according to field.

By “dragging” fields between tables in the work area (2), “Join lines” will appear. On joining connecting field’s type compatibility is verified. It is impossible to create joins between incompatible fields. For link parameter settings, place the cursor on the “join line”, right click and select Options item. The join parameters window appears, where you can set the required values: See the illustration below.

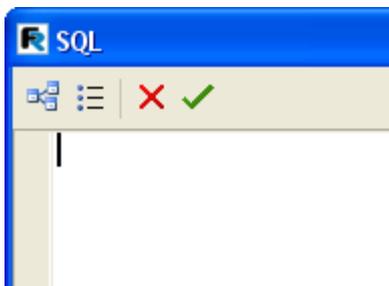


13.5.1 Query constructor usage

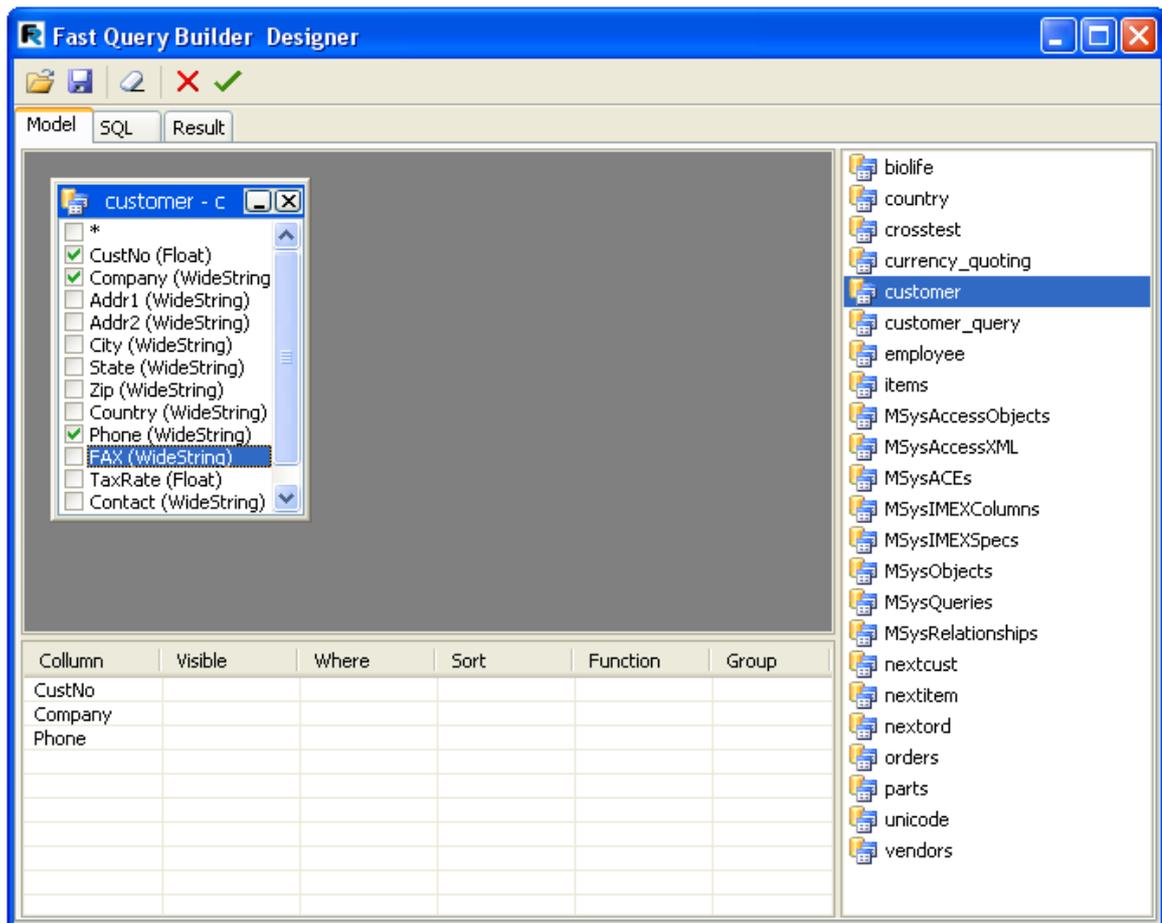
Build a simple report using the query constructor.

Click “New report”  on designer toolbar. A report page with “Report heading”, “First level data”, “Page footer” bands is created.

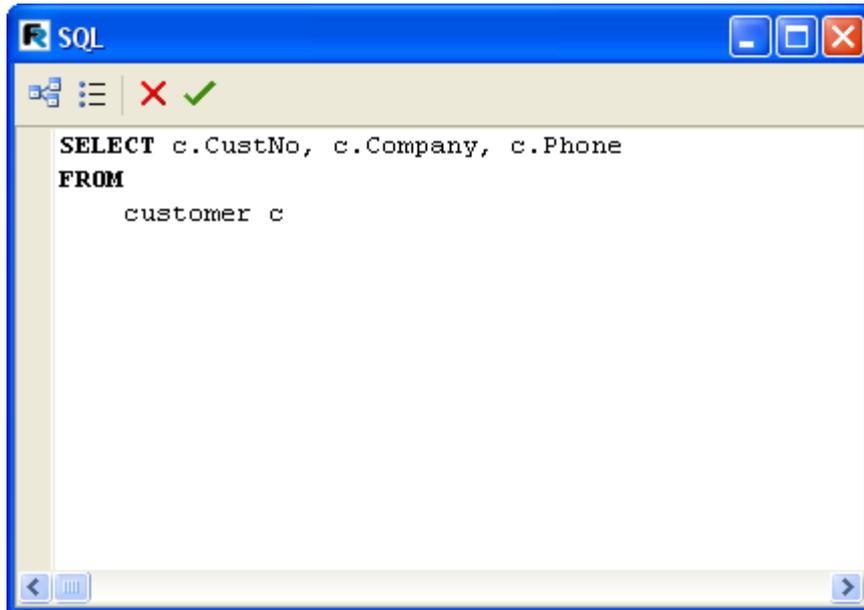
Put the "ADO Query" component on the "Data" page. Double-click on component and you will see query editor window.



Click  button in sql editor and you will see the query constructor window. Select Customer table in the table list (3) and drag it to work area (or can double-click on it). Mark CustNo, Company and Phone fields:



That is all that necessary for query building. You can see query code on SQL page tab, and on Result page tab you can see data which the query has returned. Click  to close constructor. At that we return to query editor window where the generated query code is now displayed:



Attention! If you modify query code, you will lose plan (tables placing in query constructor and their joins). Do not modify query code manually, you can always enter query constructor and correct the plan visually.

By clicking  in editor we return to the report designer. All that is left to do is to connect “Master data” band to data source and place fields on the band.

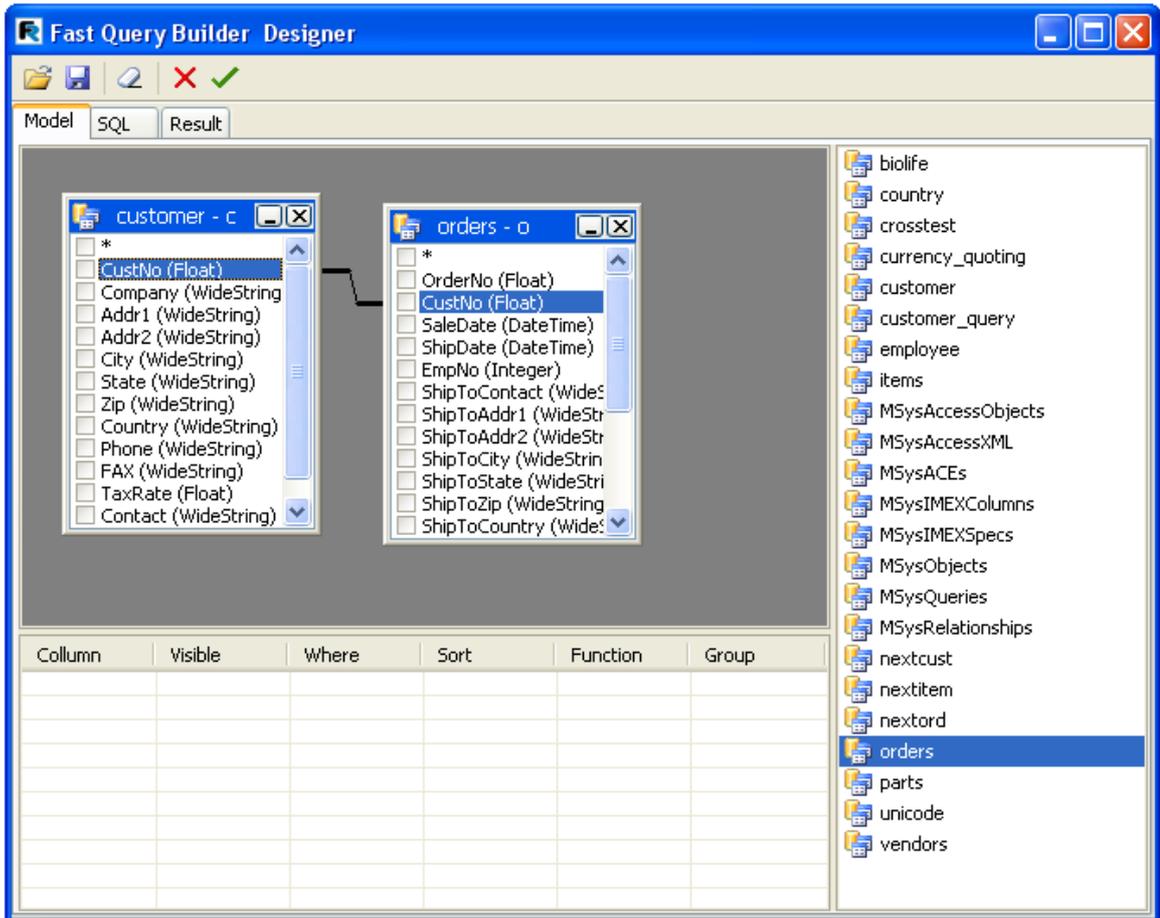
13.5.2 Complicated query building

In previous example we built reports on the basis of one table. Let us now examine query building including data from two tables.

Earlier we examined report working with groups. Let us build query for this report via query constructor. We need to make up a query in SQL language which will return data from both tables, and the data will be grouped according to a definite condition. In our case the condition will be CustNo fields in both tables.

As in the previous example, create new report and put ADO Query component on page. It query editor click button for query construction.

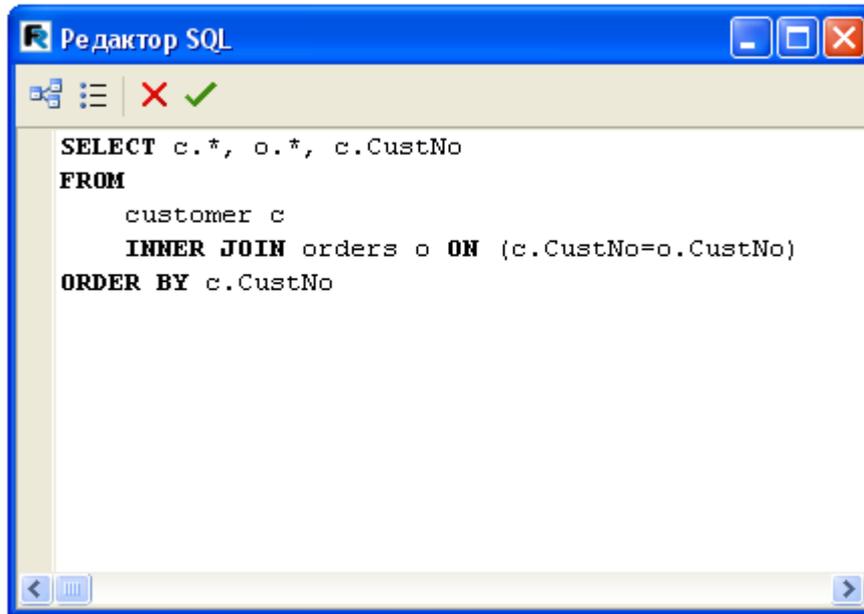
Drag two tables to work area – Customers and Orders. Both tables have CustNo field which we will use to join them. By dragging the CustNo field from one table to the other we create a join between the two tables:



Now it is necessary to mark the fields which are to be included and group it according to CustNo field. To perform this tick off "*" fields in both tables, and also CustNo field in Customer table. In the field parameters area selected fields appear. After that we need to select sorting for CustNo field:

Column	Visible	Where	Sort	Function	Group
*					
*					
CustNo	<input type="checkbox"/>		Ascending		

That is all. Query is ready. Its code looks like the following:

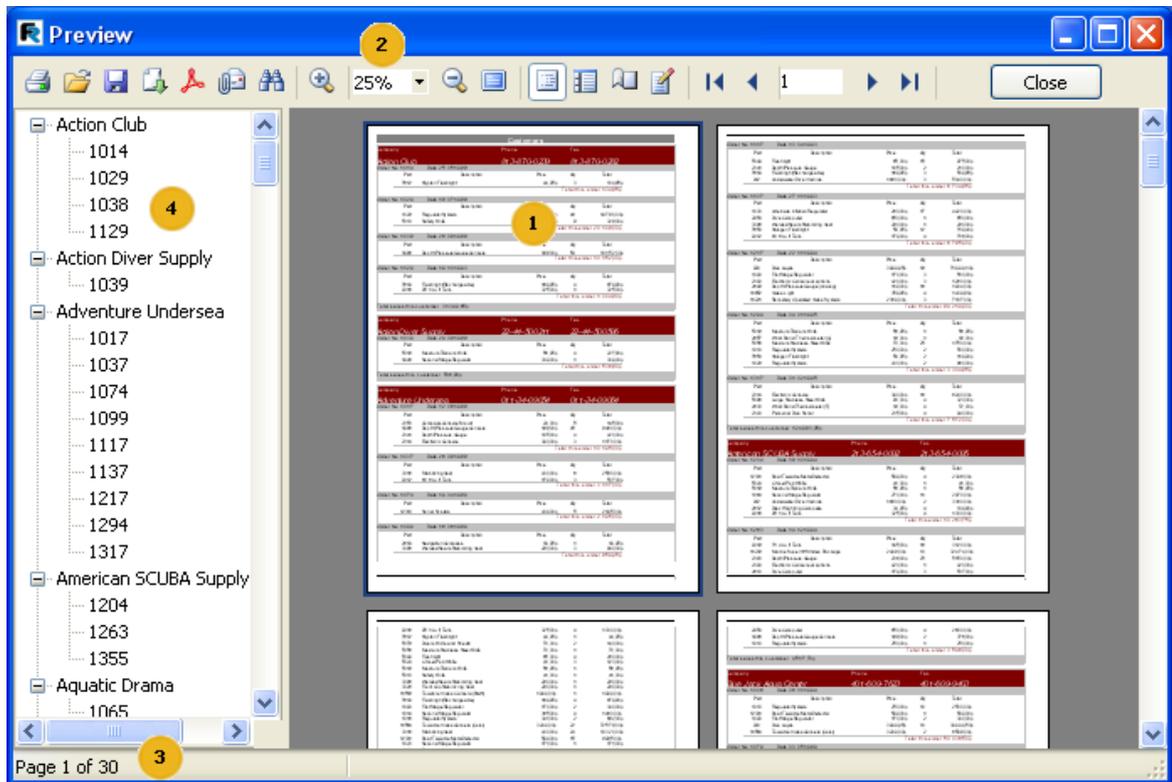


Chapter



**Report viewing,
printing and
export**

The built report can be displayed, printed or exported into one of the supported formats. Everything can be performed in preview window.



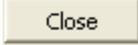
On the picture with figures the following is displayed:

- 1 – ready report pages;
- 2 – toolbar;
- 3 – status line;
- 4 – outline area. Either the outline tree (as shown on a picture) or thumbnails can be displayed here.

The following buttons are on the toolbar:



Icon	Name	Description
	Print report	Prints report. Hotkey analogue – Ctrl+P.
	Open report	Opens file with ready report (*.fp3).
	Save report	Saves report to file (*.fp3).
	Report export	Exports report to one of the supported formats.

	Export to PDF	Exports report to Adobe Acrobat file (*.pdf). This button is displayed if corresponding export filter is installed.
	Send via e-mail	Exports report to one of the supported formats and sends it via e-mail as enclosure. This button is displayed, if corresponding export filter is installed.
	Text search	Text search in report. Hotkey analogue – Ctrl+F.
	Zoom in	Zooms in the preview.
	Scale	Selects arbitrary scale.
	Zoom out	Zooms out the preview.
	Full screen	Displays report at full screen. For returning to normal conditions perform double-click on report.
	Outline	Shows or hides the report outline.
	Thumbnails	Shows or hides the thumbnail view.
	Page properties	Calls dialogue with page properties.
	Edit page	Edits current page.
	To beginning	Transfer to the first report page.
	Previous page	Transfer to previous report page.
<input type="text" value="1"/>	Page number	Transfer to report page with pointed number. Enter number and click Enter.
	Next page	Transfer to next report page.
	To end	Transfer to the last report page.
	Close window	Close preview window.

14.1 Control keys

Keys	Description
Ctrl+S	Save report to *.fp3 file.
Ctrl+P	Print report.
Ctrl+F	Text search.
F3	Continue search.
Arrows	Smooth document scrolling.

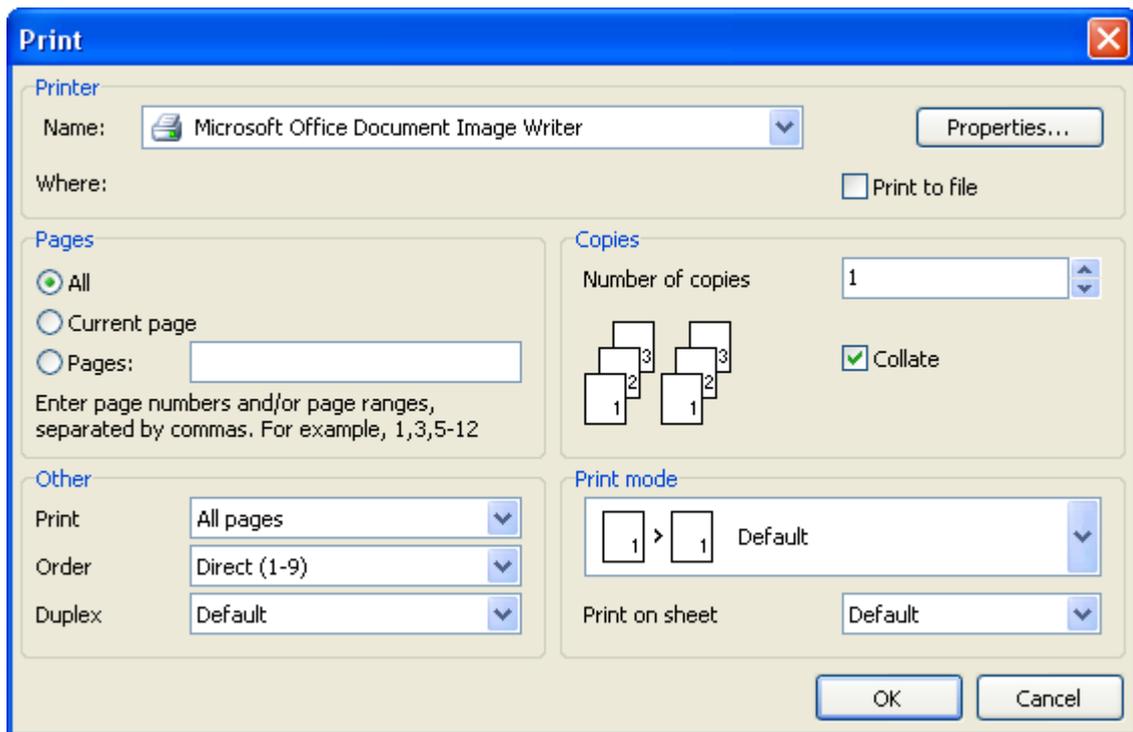
PageUp, PageDown	Up/down scrolling.
Ctrl+PageUp, PageDown	Next/previous page scrolling.
Home	Document beginning.
End	Document end.

14.2 Mouse control

Action	Description
Left button	Click on selected object (in interactive report); report scrolling in “hand” mode (move mouse with pressed button); zoom in is performed in “magnifier” mode.
Right button	Context menu; in “magnifier” mode zoom out is performed.
Double-click	In full screen mode it performs returning to normal conditions.
Mouse scroll	Report list scrolling.

14.3 Report printing

To print a report click on  button (or Ctrl+P hotkey). The window appears – it is printing dialogue.

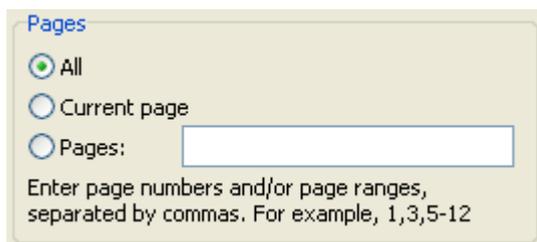


Let's look at options available in this dialogue.

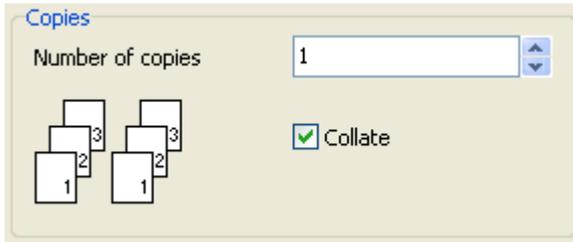
The "Printer" group: you can select a printer via which you want to print a report; set printer properties, for example, printing quality; and choose print to file.



The "Pages" group: you can select which pages to print (all, current, selected range).

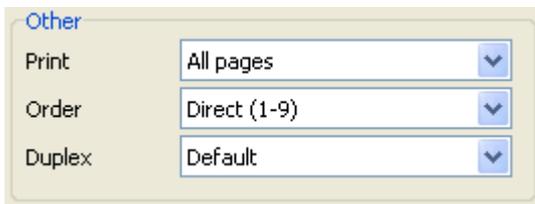


The "Copies" group: you can set how many copies to print. If Collate flag is set and you choose to print several copies, at first one report copy is printed, then – the next etc. If flag is disabled, several copies of the first page are printed, then – several copies of the second one etc.

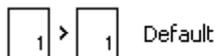
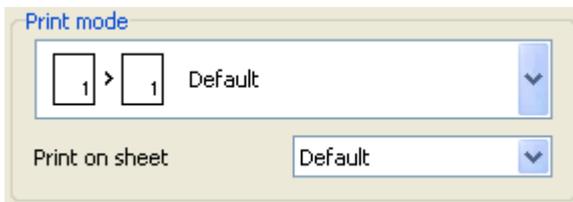


The "Other" group:

- Print - you can select which pages to print. Variants: All pages, Even pages, Odd pages.
- Order - print pages in direct or reverse order (from last to first page).
- Duplex - handle duplex by default (report settings are used) or choose one of duplex options: vertical, horizontal, simplex.



The "Print mode" group: you can select one of the printing modes.



- Default mode. The printer prints on the sheet defined in a report. One preview page is printed on one sheet.



- Split big pages. This mode is useful if you need to print A3 report on A4 sheet. One preview page is printed on several sheets. If you choose this mode, you have to choose the sheet size ("Print on sheet") as well.



- Join small pages. This mode is useful if you want to print A4 report on A3 sheet. Two or more preview pages are printed on one sheet. If you choose this mode, you have to choose the sheet size ("Print on sheet") as well.



- Scale mode. Report is printed on specified sheet. All report output is scaled. One preview page is printed on one sheet. If you choose this mode, you have to choose the sheet size ("Print on sheet") as well.

After clicking on  report printing begins. If the "Print to file" flag is selected, file name is called. And report is saved to this file (file with *.prn extension. It contains a copy of information sent to printer).

14.4 Text search in report

FastReport allows to search a set text line in a text in preview window. To perform that there is  button on toolbar (or its hotkey analogue - Ctrl+F). After that search dialogue appears:



Here you can set search line and options as well:

- Search from beginning – to search text from the beginning of document. Otherwise search will be performed from current page;
- Case sensitive – to distinguish letter cases (lower-case and capital types) on searching.

On clicking  text search is performed and the first found element is highlighted:

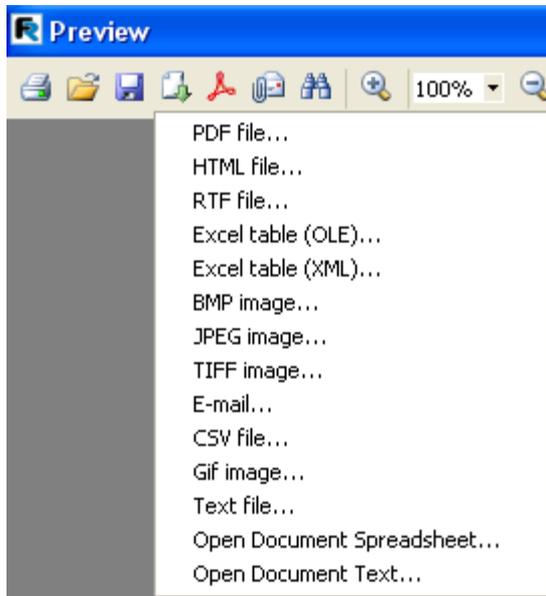
Company	Address
Action Club	PO Box 5451-F
Action Diver Supply	Blue Spar Box #3
Adventure Undersea	PO Box 744

To continue search click F3. The following element will be highlighted

14.5 Report Export

FastReport allows exporting a built (output) report to different formats for further editing, archiving, sending via e-mail, etc... To export you must add the desired FR export components to the Delphi form.

Export to 13 formats is supported. They are: PDF, Open Document Spreadsheet, Open Document Text, Excel, XML, RTF, HTML, text, CSV, BMP, Jpeg, Tiff, and Gif. There is the ability to send report via e-mail in any above-listed formats with FastReport means.



Exports in FastReport use one of the following three methods:

- Layer-by-layer – object transferring to resulting file is performed alternately. Export accuracy is approximated to original;
- Table – on object transferring transitional matrix of object allocation is used. There is high accuracy to original based on the assumption that rules of creating correct report sample were followed (“Report Design References” chapter);
- Enveloping – all report objects enveloping is performed on page image. There is full original accuracy. It is used on export to graphic formats.

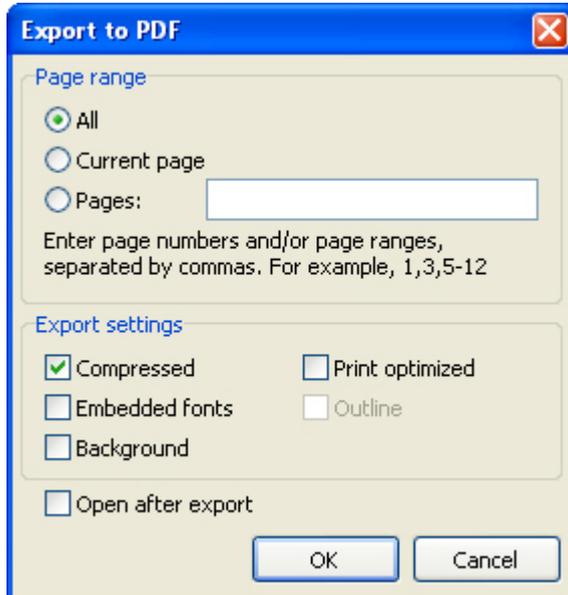
14.5.1 Export to PDF Format

PDF (Portable Document Format): a platform-stand alone format of electronic documents created by Adobe Systems. The free Acrobat Reader package is used for viewing. This format is rather flexible – it allows inclusion of necessary fonts, vector and

bitmapped images; it allows transferring and storage of documents intended for viewing and further printing.

Export method is a layered one.

On exporting to PDF format the dialogue box for output file parameter settings appears.



Export parameters:

- Compressed – output file compressing. It reduces file size but increases export time;
- Embedded fonts – all fonts used in report will be contained in the PDF output file for correct file displaying on computers where these fonts may be absent. Output file size increases considerably;
- Background – export of graphic image assigned to a page into PDF file. It considerably increases output file size;
- Print optimized – output of graphic images in high resolution for further correct printing. This option enabling is necessary only when the document contains graphics and its printing is necessary. It considerably increases output file size;
- Outline – option is enabled when report outline is used. It enables export of the outline to the PDF document;
- Open after export – resulting file is opened right after export via PDF files viewing program which must be installed in OS by default (for example, Adobe Acrobat Reader).

Export peculiarities: RichText objects are exported as a graphic.

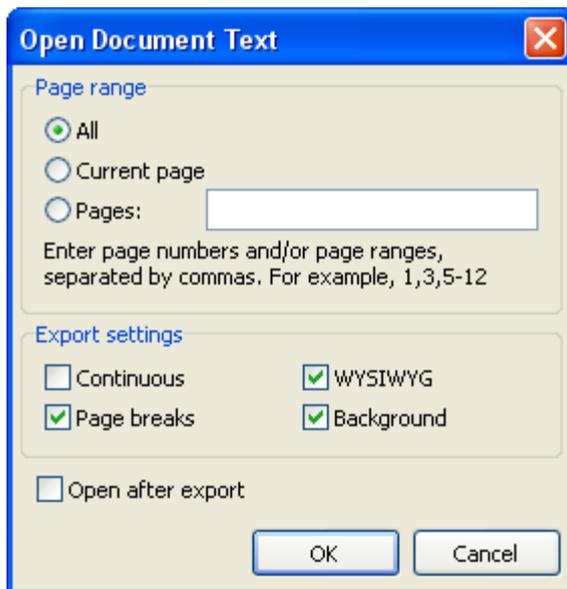
14.5.2 Export to Open Document

OpenDocument Format (ODF, OASIS Open Document Format for Office Application) was designed by OASIS and based on XML format used in OpenOffice.

FastReport supports export to table (.ods file) and text (.odt file). These files can be opened in OpenOffice.

Export method is a table one.

On exporting to ODF format the dialogue box for output file parameter settings appears.



Export parameters:

- Continuous - generate continuous document without page breaks and page headers/footers;
- Page breaks – enables page breaks in document file;
- WYSIWYG – full compliance to report appearance. Disabling the option allows optimization, reducing the number of lines and columns in the output file;
- Background – export of graphic image assigned to a page into ODF file. It considerably increases output file size;
- Open after export – output file will be opened right after export.

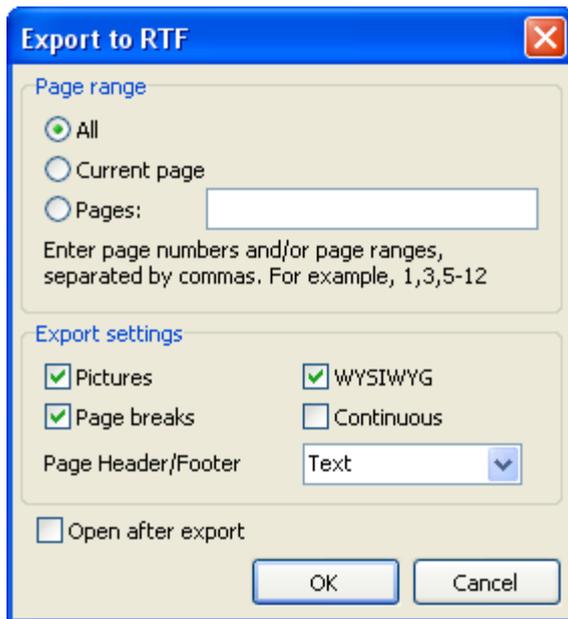
Export peculiarities: RichText objects are transferred as simple text, graphic images transference is supported.

14.5.3 Export to RTF Format

RTF (Rich Text Format) was developed by Microsoft as a standard for text documents interchange. Now RTF documents are supported by many modern text editors and operating systems.

Export method is a table one.

On exporting to RTF format the dialogue box for output file parameter settings appears.



Export parameters:

- Pictures – enables graphic images export to output file;
- Page breaks – enables page breaks in RTF file;
- WYSIWYG – full compliance to report appearance. Disabling the option allows optimization, reducing the number of lines and columns in the output file;
- Continuous - generate continuous document without page breaks and page headers/footers;
- Page header/footer - header/footer export mode. Variants are: Text (h/f exported as usual text), Header/Footer (h/f inserted in the document) and None (h/f are not exported);
- Open after export – output file will be opened right after export via RTF files viewing program which must be installed in OS (for example, Microsoft WordPad).

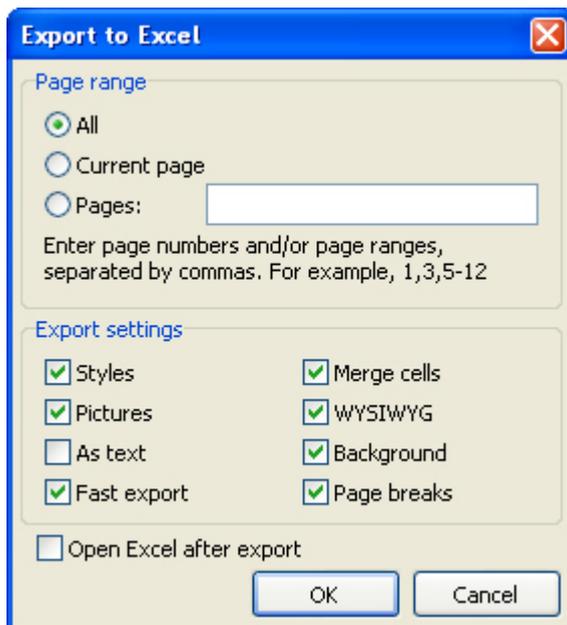
Export peculiarities: RichText objects are fully integrated into RTF format. File appearance and size depend on report sample accuracy (“Report Design References” chapter);

14.5.4 Export to Excel

Excel – application for working with electronic worksheets. It is included into Microsoft Office System.

Export method is a table/diagram one.

On exporting to Excel format the dialogue box for output file parameter settings appears.



Export parameters:

- Styles – transferring of text objects design styles into the table. Disabling increases exporting speed but worsens document appearance;
- Pictures – includes graphic images export into output table;
- As text – all objects are transferred into table/diagram as text ones. This option may be useful when transferring numeric fields with complicated formatting;
- Fast export – usage of optimized fast data transferring to Excel. This option disabling slows down data transferring but increases export compatibility on any errors during data transferring;
- Merge cells – cells integration in resulting table/diagram for achieving maximum correspondence to the original. Disabling increases exporting but reduces document appearance;
- WYSIWYG – full compliance to report appearance. On this option disabling the optimization for reducing the number of lines and columns in resulting table is performed;
- Background – export of filling color assigned to report page;
- Page breaks – includes page breaks in Excel;
- Open Excel after export – resulting file will be opened right after exporting into Excel.

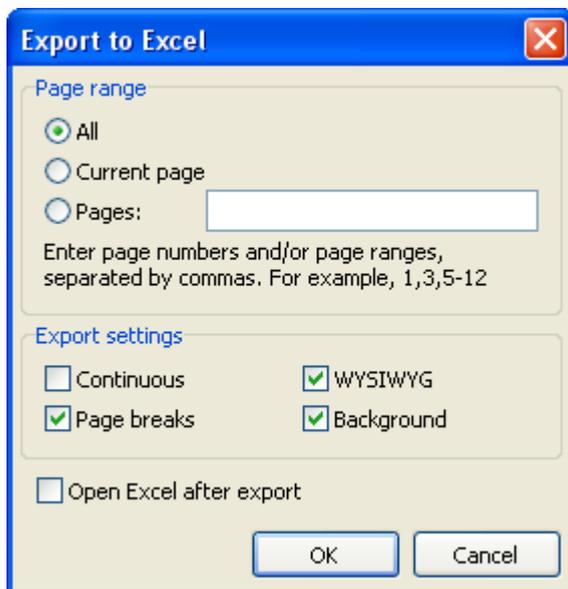
Export peculiarities: Excel program must be installed on your PC. RichText objects are transferred as simple text, graphic images transference is supported.

14.5.5 Export to XML Format

XML (Extensible Markup Language) is an expansible marking language. XML is intended for structured data storage and also for information interchange between different programs. FastReport uses XML format for data transferring into Excel table/diagram editor ver. 2003 and later.

Export method is a table/diagram one.

On exporting to XML format the dialogue box for output file parameter settings appears.



Export parameters:

- Continuous - generate continuous document without page breaks and page headers/footers;
- Page breaks – includes page breaks in resulting document;
- WYSIWYG – full compliance to report appearance. Disabling allows reducing the number of lines and columns in resulting table;
- Background – export of filling color assigned to report page;
- Open Excel after export – resulting file will be opened right after exporting into Excel.

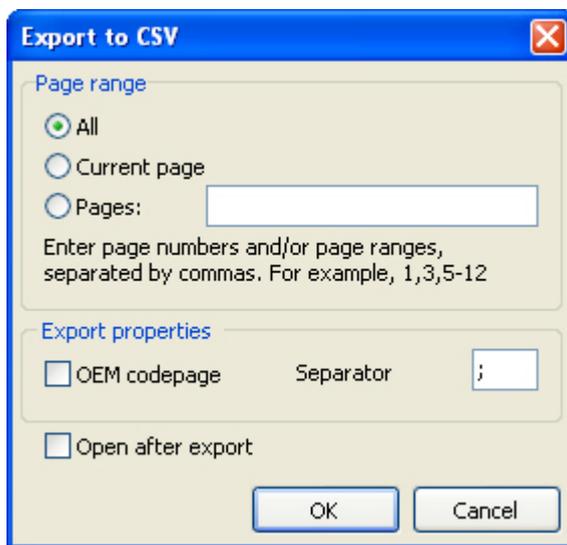
Export peculiarities: Excel program must be installed on your PC. RichText objects are transferred as a simple text; graphic images are not supported.

14.5.6 Export to CSV Format

CSV-file contains values formatted in the form of a table/diagram and adjusted in such a way that every value in column is divided from value in the next column by means of separator, and every new row begins with new line. This format may be imported into different table/diagram editors.

Export method is a table/diagram one.

On exporting to CSV format the dialogue box for output file parameter settings appears.



Export parameters:

- OEM codepage – resulting file OEM coding selecting;
- Separator – values separator in files;
- Open after export – resulting file will be opened right after exporting via CSV files viewing program which must be installed in OS.

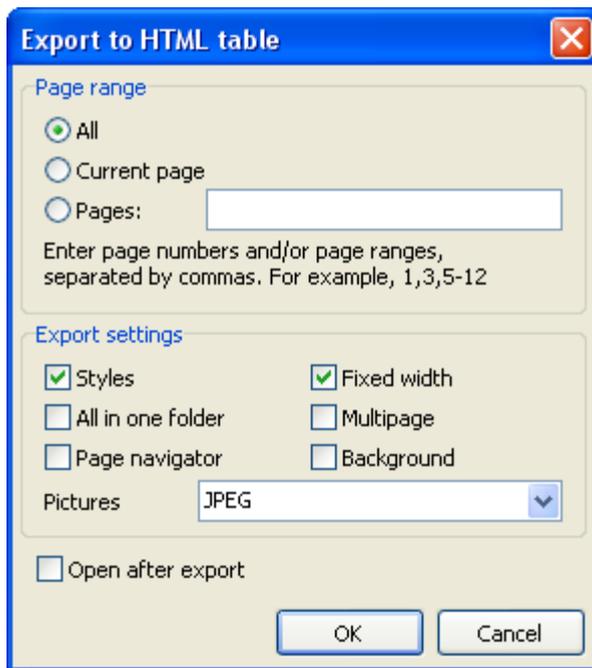
Export peculiarities: on transferring into this format report design is not saved. Graphic images are not supported.

14.5.7 Export into HTML Format

HTML (Hypertext Markup Language) is regarded as standard language for document marking in the Internet. HTML was created as a language for scientific and technical documentation interchange suitable for usage by people who are not specialists in nesting. It is used for creating relatively simple but nicely designed documents. Besides document structure simplification hypertext support is included into HTML.

Export method is a table/diagram one.

On exporting to HTML format the dialogue box for output file parameter settings appears.



Export parameters:

- Styles – transferring of text objects design styles. Disabling increases exporting but worsens document appearance;
- All in one folder – all additional files are saved in the same folder with main file;
- Page navigator – special navigator for fast shift between pages is created;
- Fixed width – blocking of automatic table/diagram width modifying on changing preview window size;
- Multipage – every page will be written to separate file;
- Background – export of graphic attributes assigned to report page;
- Pictures – includes graphic images exporting possibility;
- Open after export – resulting file will be opened right after exporting via HTML files viewing program which is allocated in OS by default.

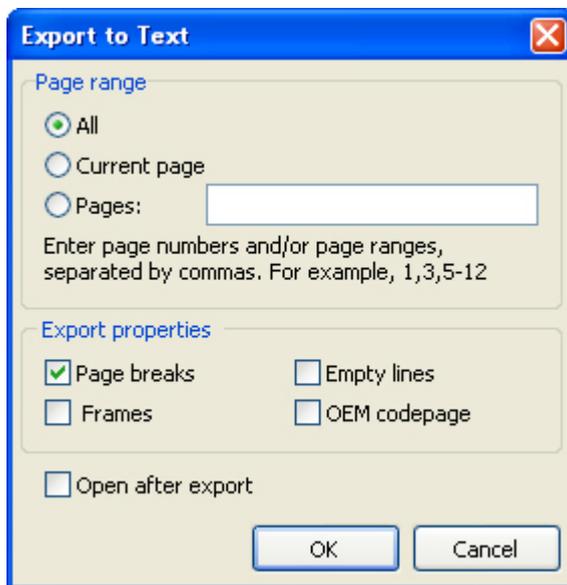
Export peculiarities: export may consist of several files. Each graphic image is supported and saved in their own separate file, RichText objects are transferred as simple text. Appearance and file volume depend greatly on report design (“Report Design References” chapter)

14.5.8 Export to Text Format

Usual text file. It contains information from report. This information is optimized to the limit and converted in accordance with the given format peculiarity.

Export method is a table/diagram one.

On exporting to text format the dialogue box for output file parameter settings appears.



Export parameters:

- Page breaks – export of page breaks to resulting file;
- Empty lines – export of empty lines;
- Frames – export of text objects frames;
- OEM codepage – resulting file OEM coding selecting;
- Open after export – resulting file will be opened right after exporting via default text files viewing program which is installed in OS.

Export peculiarities: report design is not saved on transferring to this format, graphic images are not supported, exported page width is figured automatically depending on type of text objects on report page.

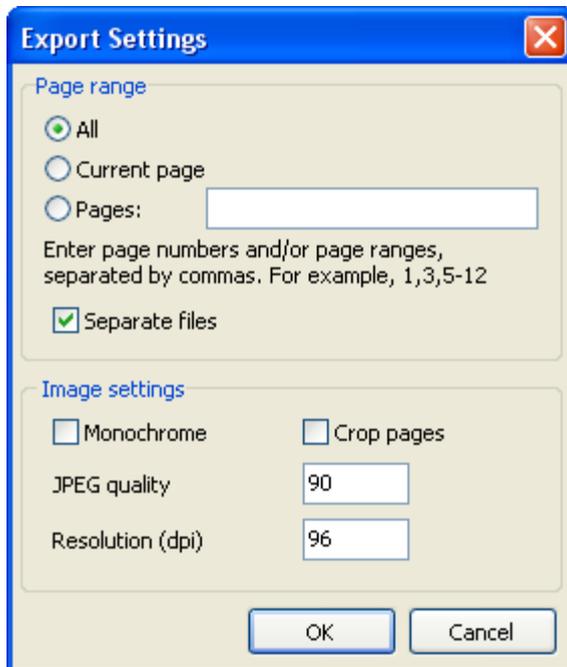
14.5.9 Export to Jpeg, BMP, Gif, Tiff Graphic Formats

FastReport allows exporting information to graphic formats.

- JPEG (Joint Photographic Experts Group) – is a format based on shrinking algorithm which is based not on the same elements search but on difference between pixels. It is characterized by high compression level at the expense of partial graphic information loss.
- BMP (Windows Device Independent Bitmap) – is used for storage of bitmap images assigned for usage in Windows. A standard file format for computers under Windows control.
- GIF (Graphics Interchange Format) – independent from hardware support the GIF format was developed for bitmap images transferring through networks. It allows compression of files containing many homogeneous fillings (logos, inscriptions, schemes) rather well.
- TIFF, TIF (Target Image File Format) – hardwarily stand-alone format. Today it is one of the most widespread and reliable in polygraphy and facsimile information transferring.

Export principle is enveloping.

On exporting to one of above-named graphic formats the dialogue box for image parameters setting appears.



Export parameters:

- Separate files – if option is enabled, every report page is exported to separate file. File name is given according to the selected one with addition of underlining and page number;
- Monochrome – monochrome picture creating;
- Crop pages – after exporting blank area cropping will be performed along edges;
- JPEG quality – JPEG file compression ratio. Option is enabled only on exporting to JPEG format;
- Resolution – output graphic presentation resolution.

Export peculiarities: on exporting several pages to one file (on disabled Separate files option) it is necessary to remember large resources capacity of export.

14.6 Sending a Report via E-mail

FastReport allows sending a ready report by e-mail in any format you need. You do not need any additional programs to send mail.

On selecting export by e-mail the dialogue box for setup of message and exporting format parameters appears. Before export forming and sending via e-mail, it is necessary to set parameters of mail box owner. All these parameters are on the “Account” page tab:

The screenshot shows a dialog box titled "Send by E-mail" with two tabs: "E-mail" and "Account". The "Account" tab is active. It is divided into two sections: "Mail" and "Connection".

Mail Section:

- From Name: Mike Smith
- From Address: mike@hotmail.com
- Organization: (empty)
- Signature: --
Best regards,
Mike
- Build button: (present)

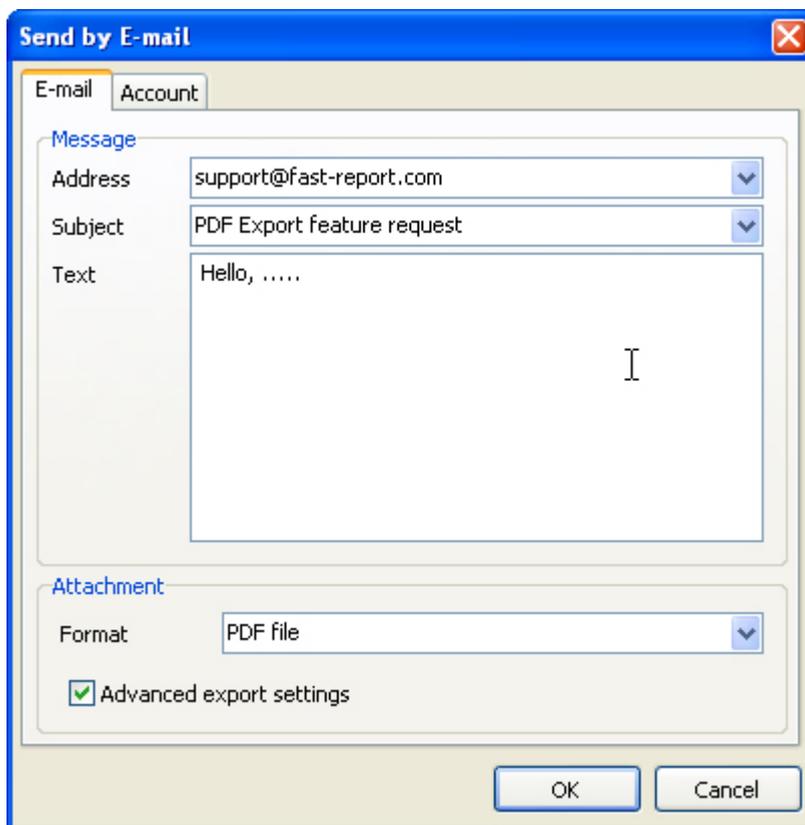
Connection Section:

- Host: smtp.hotmail.com
- Port: 25
- Login: mike_smith
- Password: *****
- Remember properties:

At the bottom of the dialog are "OK" and "Cancel" buttons.

- From Name – sender’s name;
- From Address – sender’s e-mail;
- Organization – sender’s organization;
- Signature – signature for mail. It may be automatically formed on clicking on “Build” button on condition that the earlier examined fields are filled;
- Host – SMTP server port;
- Port – SMTP server port;
- Login – access name for authorization on SMTP server, if its usage is necessary for mail sending via specified SMTP server;
- Password – authorization password;
- Remember properties – remember all parameters for further usage.

After filling in the necessary parameters for mail sending, you must fill in message parameters in "E-mail" page tab:



- Address – e-mail address of receiver. Earlier used addresses can be selected in drop-down menu;
- Subject – message subject. Earlier used topics can be selected in drop-down menu;
- Text – message text;
- Format – format of report attached to mail. One of the available export formats and also own format of FastReport (FR3) ready report may be selected;
- Advanced export settings – on this option enabling after clicking on “OK” the dialogue box for selected export format setting appears. Otherwise default export parameters will be

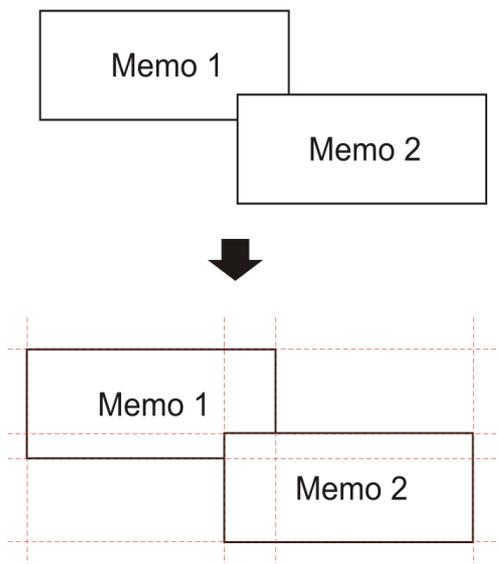
used.

Export via e-mail peculiarities: only plain authentication on SMTP servers is supported. If authentication is not required, it is not necessary to fill “Login” and “Password” fields in settings.

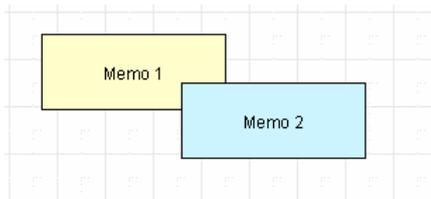
14.7 Report Design References

It is significant that the quality of the export into any other format depends greatly on competent design of initial report. FastReport allows a great number of ways to manipulate objects during report output creation. This gives the advantage of fast development of any reports and their further printing. Printed document will look just as on display. And this is the primary intent of FastReport report generator usage. The downside of such development freedom is the complexity of exporting the FastReport document to different data formats, which have their own limits and requirements for information presentation, and are sometimes rather complex. In this chapter, special design requirements of reports intended for export to other data formats will be discussed.

Many formats use table data presentation. Formats such as HTML, XLS, XML, RTF and CSV, do. Not allow cell crossing or arranging in layers when table marking, this concerns HTML and RTF. In contrast to freedom of report development in FastReport designer. Export filters, as a rule, take into account these requirements when objects are transferred from FastReport report to necessary format. This is carried out by special algorithms which takes object crossings into account and their optimal placing. At object crossing points new columns and lines in the resulting output table appear. That is necessary for saving of the FastReport transferable objects exact positioning and for getting maximum resemblance between the result and original report. A large number of cross objects in report design, leads to an increased number of columns and lines in the resulting table. This leads to the need to edit the resulting file in its own editor for further use.



For example, on report design a slight crossing of two objects placed one under another on the same band. The number of records on report forming was 150. On export to RTF format 450 lines will be created (150 lines for each object and 150 ones for crossing). If we remove crossing there will be already 300 lines. In large reports and on huge number of objects the difference will be really tremendous. That, of course, will affect output file size.



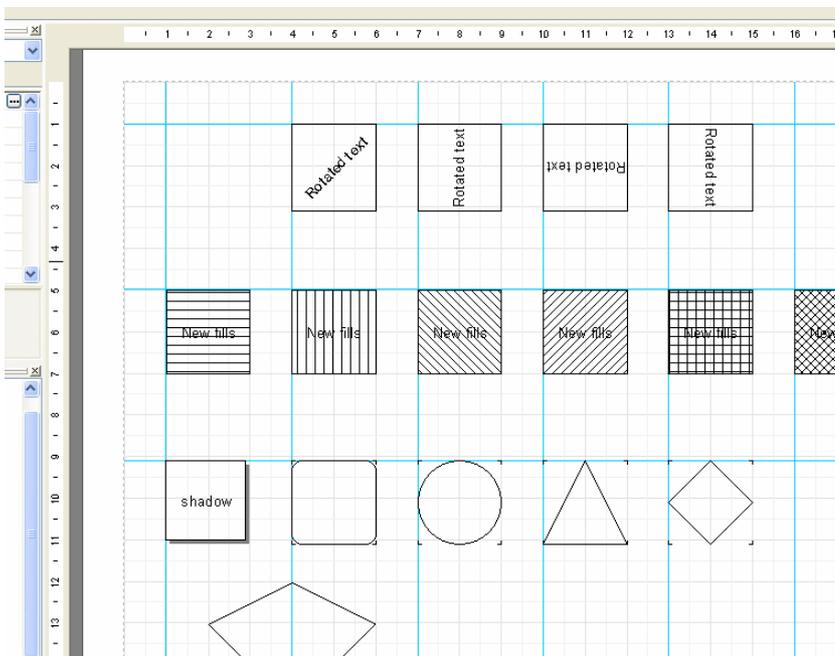
Objects in the report

	A	B	C	D
1	Memo 1			
2		Memo 2		
3				
4				

Export to Excel - result

Remember that during designing, if you want to export your reports in any table format.

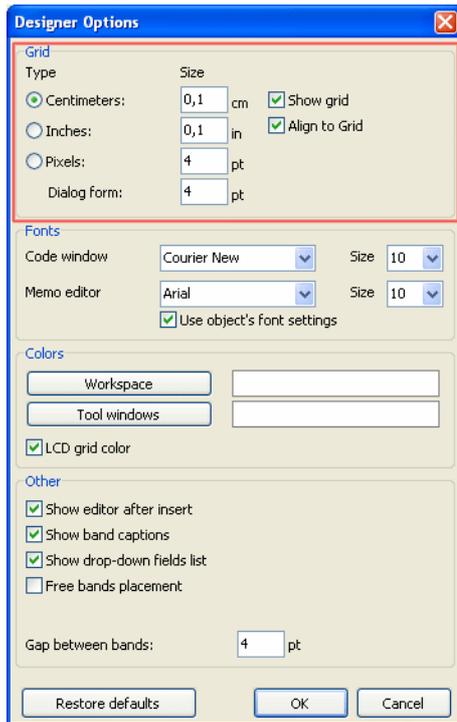
On creating tables in report keep an eye on neighboring cell's borders to adjoin each other. It is important that cells do not cross and arrange in layers. Export filter algorithm will cut off cells but export result may be far from desirable (you will see not exactly what you wanted to). Arrange objects in such a way that they are placed in line vertically as well as horizontally. Guidelines can help to perform this.



Using guidelines in the designer

To use guidelines in FastReport designer just click on the horizontal or vertical ruler limiting report page from the left and the top. Then, holding the mouse button down, drag the guideline to the required position on the page. You will be able to place objects immediately along guidelines horizontally and vertically.

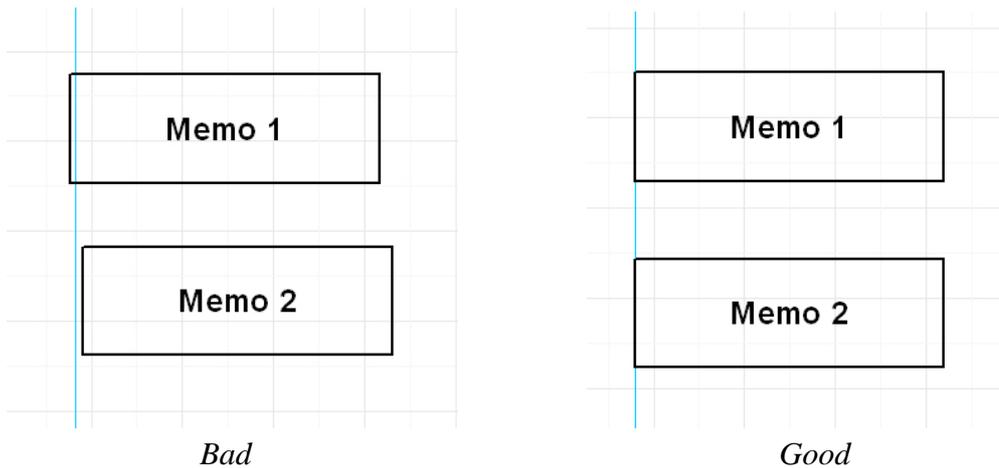
Text objects grid alignment can also be helpful in case of cells overlapping. Keep an eye on enabling grid alignment in designer options. In order to simplify alignment you can extend grid pitch. Setting of grid pitch and alignment can be found in designer menu “View” – “Options” – “Grid”.



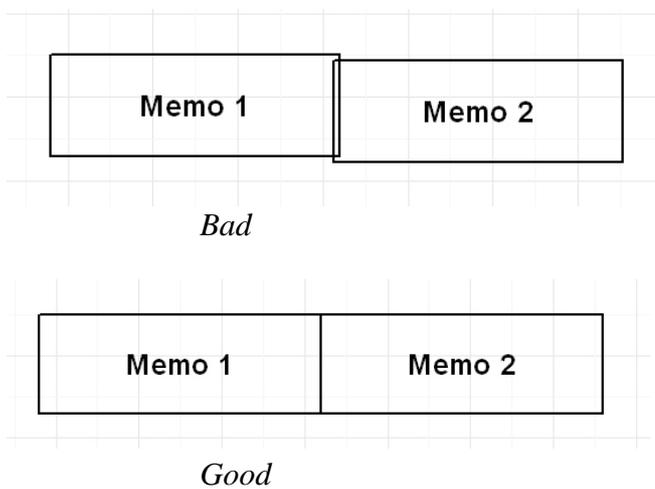
For text framing it is better to use text object embedded properties instead of single graphic objects – lines, rectangles, etc. try not to use background objects under transparent text objects.

Keeping these simple rules in mind will help you to create a report which will look perfect after export to any format using table (or table-based) marking for data presentation.

Below there are some examples of correct and undesirable object arrangement on report design creation.



Objects are displaced horizontally. It is necessary to use alignment according to extension lines as far as possible for objects to have the same horizontal coordinate.



Objects are overlapping. In such a case on export to table/diagram format additional useless lines and columns and also 3 additional cells in crossing zone are created.

It is recommended to get acquainted with demo reports included with the FastReport installation for mastering basic methods of optimum report development.

