

# **FastReport v2.4**

## **Report generator**

### **Developer's manual**

## Table of contents

<b>INTRODUCTION .....</b>	<b>5</b>
PREFACE.....	6
COMMENTS ABOUT FASTREPORT .....	7
THE HISTORY OF FASTREPORT .....	8
BUILDING REPORTS.....	9
Data.....	9
Parameters.....	10
Forms .....	10
Processing.....	11
Preparing reports.....	12
EXAMPLE: BUILDING A SIMPLE REPORT .....	13
<b>THE FASTREPORT KERNEL .....</b>	<b>15</b>
DELPHI COMPONENT PALETTE .....	16
"FastReport" tab .....	16
"FR Tools" tab .....	18
TfrReport component .....	19
TfrDBDataSet component.....	22
TfrUserDataSet component.....	23
FASTRPORT OBJECTS .....	24
The "Text" object.....	25
The "Band" object.....	28
The Picture object.....	30
The SubReport object.....	31
The Line object.....	31
The CheckBox object.....	31
The RichText object.....	32
The OLE object.....	32
The Chart object.....	33
The Shape object.....	35
The Barcode object.....	35
The RichText 2.0 object.....	36
BUILDING THE REPORT.....	37
Bands in the FastReport.....	38
Simple report (list) .....	42
Master-detail report.....	42
Master-detail-subdetail report.....	42
Cross-tab report.....	42
Dynamic reports.....	43
Broken bands .....	44
Multicolumn report .....	45
Report with title page.....	45
Nested reports (subreports).....	45
Master-Detail-Detail report.....	45
Composite report.....	46
Report with BLOb fields.....	46
Report without bands .....	46
Report with groups.....	46
Report with charts.....	47
<b>THE DESIGNER.....</b>	<b>49</b>
THE DESIGNER .....	50
Using the keyboard .....	51
Using the mouse.....	51
Report options.....	52
Page options.....	53
The "Paper" tab .....	53
The "Paper source" tab.....	53
The "Margins" tab .....	53
Designer options .....	55
The Object Inspector .....	57

<i>The "Insert data fields" window</i> .....	58
<i>The Data dictionary</i> .....	59
<i>The Expression builder</i> .....	64
<i>"Insert data field" dialog</i> .....	64
<i>"Insert variable" dialog</i> .....	64
<i>"Insert function" dialog</i> .....	65
<i>Toolbars</i> .....	67
<i>The "Standard" toolbar</i> .....	67
<i>The "Text" toolbar</i> .....	68
<i>The "Rectangle" toolbar</i> .....	69
<i>The "Alignment " toolbar</i> .....	69
<b>END-USER FEATURES .....</b>	<b>71</b>
INTRODUCTION .....	72
THE DIALOGUE FORMS.....	73
<i>Dialogue Form Controls</i> .....	74
<i>Label</i> .....	74
<i>Edit</i> .....	75
<i>Memo</i> .....	75
<i>Button</i> .....	76
<i>CheckBox</i> .....	76
<i>RadioButton</i> .....	77
<i>ListBox</i> .....	77
<i>ComboBox</i> .....	78
<i>Passing the information to the Report</i> .....	80
DATA ACCESS COMPONENTS.....	81
<i>Description of FastReport DB-aware components</i> .....	82
<i>TfrBDELookupComboBox</i> .....	83
<i>TfrBDETable</i> .....	83
<i>TfrBDEQuery</i> .....	86
<i>TfrBDEDataBase</i> .....	87
<i>Building Reports</i> .....	89
SIMPLE "TABULAR" TYPE REPORT.....	89
<i>Report with parameters</i> .....	90
THE TFRDATASTORAGE COMPONENT.....	92
CONNECTING TO A DATABASE.....	92
OPENING A TABLE.....	93
GENERATING A QUERY.....	94
FIELDS EDITOR.....	95
CREATING LOOKUP FIELDS.....	96
QUERY PARAMETERS EDITOR.....	97
JOINING DATA.....	97
PARAMETERS DIALOG.....	98
DESIGNER OF PARAMETERS DIALOG.....	98
BUILT-IN LANGUAGE.....	100
<i>Scripts and objects</i> .....	100
CODE WRITING.....	101
<i>Objects modification</i> .....	103
<i>Built-in functions</i> .....	104
<i>Aggregate functions</i> .....	104
<i>String functions</i> .....	104
<i>Properties and methods of objects</i> .....	107
<i>Using the interpreter</i> .....	118
<b>PROGRAMMING.....</b>	<b>119</b>
EVENT HANDLERS.....	120
<i>Other events of TfrReport object</i> .....	120
VARIABLES.....	122
EXTENDING FASTREPORT FUNCTIONALITY.....	124
<i>Making your own preview windows</i> .....	124
<i>Expanding the functions list</i> .....	125
<b>EXAMPLES OF REPORTS.....</b>	<b>127</b>

EXAMPLES OF REPORTS.....	128
<i>Insertion of graphs and diagrams in the report.....</i>	<i>129</i>
<i>Controlling the logic of report composition using the OnManualBuild event.....</i>	<i>132</i>
<i>Manual report composition at runtime using code.....</i>	<i>133</i>
<i>Printing of column reports with variable or unknown number of columns .....</i>	<i>134</i>
<i>Column reports with variable width of the columns .....</i>	<i>136</i>

# INTRODUCTION

Preface

Comments about FastReport

The history of FastReport

Stages of building report

Building simple report

## **PREFACE**

This manual assumes you are familiar with report writers and understand the basic concepts of report writers (i.e. bands, data sources, two-pass reports, etc.). The manual will help you get started creating reports with FastReport, but it won't help you with the basics of report writers.

If you are not familiar with report writers, we suggest you consult the QuickReports help system. A manual for QuickReports may also be included with your copy of Delphi. The basic concepts of QuickReports apply in most cases to FastReport, however, FastReport offers far more flexibility and end-user customisation.

## COMMENTS ABOUT FASTREPORT

FastReport is a highly flexible report designer, whereby data for the report can be obtained from just about any type of data source, including strings lists, BDE databases, ADO datasources (without using the BDE), Interbase (using IBO), Pascal arrays and records, to name just a few!

The entire FastReport system is written entirely in Delphi Pascal. The FastReport system does not require any Dll's to be installed and adds approximately 400kb (Delphi 5) to your *empty project*. If you want end-user designer capabilities, this will add an additional 500kb to your .EXE. Although this may seem large, this is a fraction of any other report writer. You also need to consider that FastReport includes the ability of the end-user to not only change the design of the report, but also the ability to change the queries and databases which the report draws its data from. FastReport even includes its own scripting language, to ensure any report can easily be changed by both the programmer and the end-user. If many of your applications use FastReports, you can simply deploy the FastReport BPL (around 1400kb) and all your programs will remain small.

FastReport has one of the most attractive user interfaces you will find, with all the latest user interface components, such as dockable toolbars. Your end-users will definitely approve of the designer, where most reports can be created using only the mouse.

FastReport has not named **FastReport** for nothing: compare it to any other Delphi report writer and you will find nothing comes close to it in terms of speed. The report preview window also leaves most report writers in the dust with its sleek, highly polished look, giving your application an extremely professional look.

FastReport is a proven report writer which has been around for over 3 years and has grown from strength to strength offering features no other Delphi report writer can match.

## THE HISTORY OF FASTREPORT

FastReport was borne out of necessity. While I was developing a salary calculation system in 1997, I looked around for a report writer which allowed me to create reports easily and which allowed me to be able to modify the reports at run-time. However, at the time no components were freely available which suited my needs. As such, I went about creating my own reporting tool and FastReport was born.

The basic concept for FastReport was taken from “1S-Bookkeeping” 6.0 for Windows, whereby the basic reporting element is a framed rectangle object with multi-line text inside it. This text can contain a mixture of standard text as well as **variables**. Variables, such as data fields, are denoted by enclosing them in square brackets. The first version of the report generator only supported one band, but it allowed the creation of multi-level reports. It was also not a component, but merely a set of units.

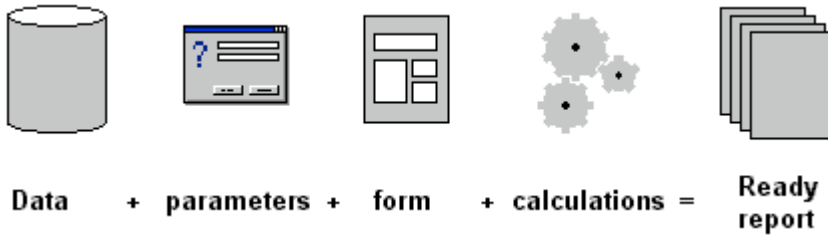
Later, in 1998, the report generator became a fully fledged Delphi component. At this point, its name became “FastReport” and it offered increased functionality. Today, FastReport is a fully-fledged, industrial-strength visual report generator. Some of its capabilities include:

- Built-in report designer, which is also available at run-time (great for end-user report modifications!).
- Preview similar to MS Word print preview.
- Compiles directly into the Delphi EXE, no DLL's required.
- JPEG (using Delphi library) and GIF (using RX library) support.
- Fast- performance comparable to QuickReport1 with far more features.
- Compact, pure Delphi code– without the designer its footprint is smaller than QR3!
- Powerful band-oriented report generator like QuickReports and ReportBuilder.
- Set of very useful components including: Text, Line, Picture, Shape, OLE object, RichText, RX Rich 2.0, Chart, Barcode, shadowed text.
- Unlimited number of pages in the prepared report.
- Multi-page reports; composite reports; sub-reports; groups; multi-column reports; master-detail-detail reports; cross-tab reports; two-pass reports.
- Full control over the printing process; supports all paper sizes.
- TXT, RTF (with graphics), CSV, HTML (with graphics) export filters.
- Text search in the report preview.
- Additional component TfrDataStorage allows for the creation of tables and queries at run-time. This is especially useful for end-user reports to give them total control over the reporting process.
- The pages of the prepared report can be edited.
- A built-in, Pascal-like interpreter is included for very flexible reporting! Syntax highlighting is supported using a freely available freeware library.
- The report data can be stored in the Delphi DFM file, an external file, a BLOB field of DB table or can be streamed.
- FastReport can easily be extended by creating your own report components, wizards and function libraries.
- IBOjects supported in reporter core (without the BDE).
- Interbase Express (IBX) support.
- ActiveX Data Objects (ADO) support in reporter core.
- Complete data manager functionality (without the BDE).



## Building reports

Building reports consist of the following stages:



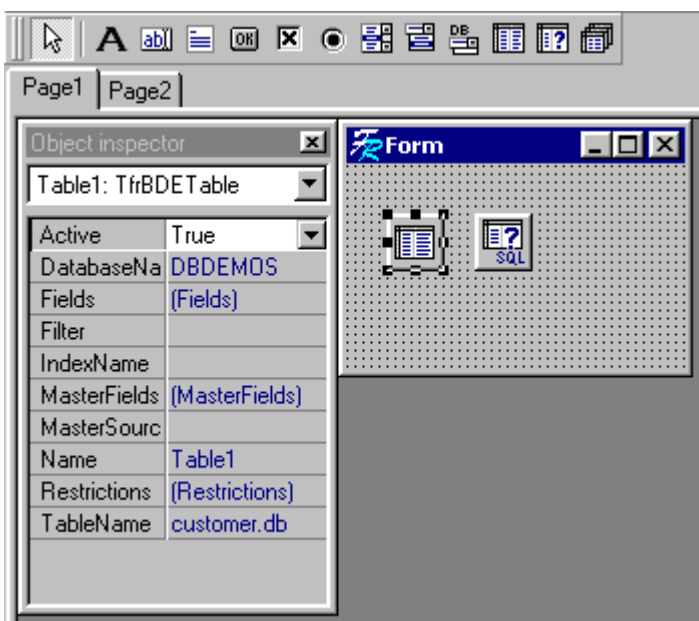
We will look into each of these stages and their realization in FastReport. Also we will compare these stages in Fastreport to some other report generators.

### Data

The majority of reports are founded on data from a database. Delphi itself gives efficient mechanisms for accessing databases. These mechanisms are also used in FastReport. The TTable and TQuery components can be used as sources for a report. In general it is possible to use any descendant from a TDataSet component. The access to data is realized inside the FastReport kernel without interference of a programmer.

Except data, stored in a database, FastReport can use practically any sources (arrays, files, StringGrids etc). In such cases a programmer must take care of accessing the non-database source itself. In Fastreport there is a set of events that allows a programmer to pass data to the FastReport kernel.

Accessing data is approximately alike in all report generators. All report generators can deal with DB-aware components, placed on the project forms. Except access to data, FastReport, ReportBuilder, and QR+QRDesigner allows creating new components in run-time. In FastReport the creation of DB-aware components are comparable to those that are used in the Delphi IDE. In the same way, just like in Delphi, you place components on a form and change its properties in the Objects Inspector. Component ideology is very flexible: it is possible to easily create new components for support of different database engines.

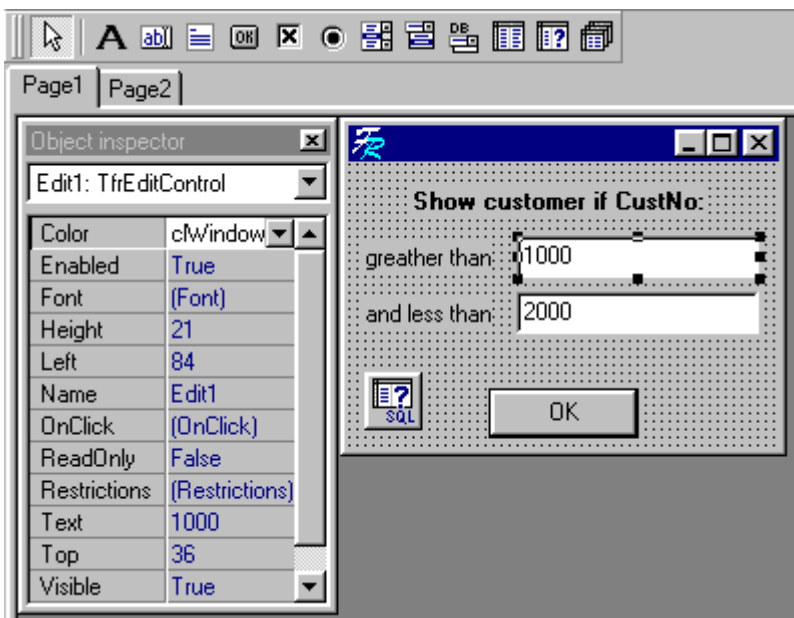


## Parameters

Users can be asked for input (parameters). Examples of user-input are: ranges of dates, a company's name, an invoice number, etc. Some reports don't use parameters at all or use fixed parameters (without requesting their values in a dialog).

The handling of parameters are differently realized in report generators. In ReportBuilder and QR+QRDesigner there is the possibility of requesting parameters if the report uses data from a query. For dialogue with a user a standard dialog window is used. Besides that, for requesting parameters it is also possible to use forms, created in the Delphi IDE. But this requires some changes to programming logic and you must recompile your project.

FastReport, on the contrary, allows end users to develop the dialog forms. This process is like the building forms in the Delphi IDE: there is set of standard controls, which can be dropped on the dialog form. If necessary you then can change their properties. Also, the built-in language in FastReport allows you to create dialogues by using pascal-like programming logic and pass the entered values to the kernel.

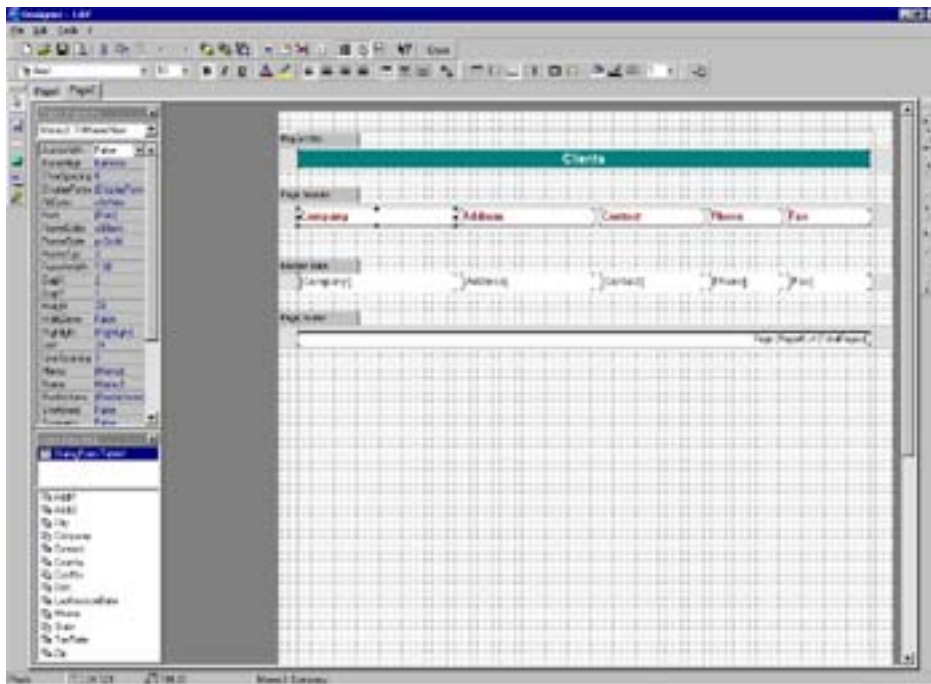


The possibility of creating your own dialogues is very useful. Other possibilities (creating data sources, use of the built-in language, etc.) allows you to create universal reports i.e. reports, independent from the application (compiled and build project). This allows you to create new reports and modify existing reports without rebuilding or recompiling the project.

## Forms

The report form itself presents a set of elements, describing how exactly the report must look. For grouping elements upon their location in reports, FR uses bands. There are two types of bands: service bands (report headers, page numbering, etc.) and bands that are forming multi line parts of reports (hereafter: data-bands). Data-bands are connected to the data sources, and the content is shown as long as there are records in the underlying data source.

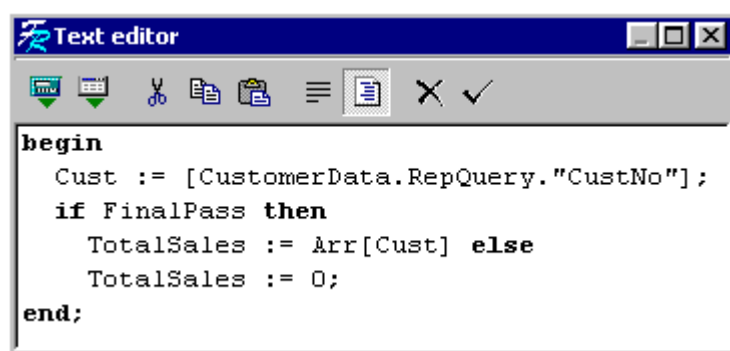
The visual environment of report development – the designer - is used for building reports. In the FastReport designer you can develop reports, combining power, simplicity and comfort of use. The interface of the designer consists of instrument panels (toolbars). Of course you can change the place of these toolbars, as you like. To manipulate object's properties an Objects Inspector can be used, similar to the one used in Delphi.



## Processing

Processing input data, modification of report forms or components are stages (processes) in the building of a report. An example of such a process is showing negative amounts in red. A more complex example of a process is printing the total amount of sold products to a customer, which is presented in the group footer (or a group header).

Realizing such a process is possible by writing event handlers in Delphi and exactly so it can be done in FastReport, QR, or ReportBuilder. This way is not universal since it does not allow creating new reports outside of Delphi without rebuilding the project. This is why in FastReport (and ReportBuilder) a built-in language is applied, analogue to Pascal, but simplified. Scripts written in this language are event handlers, fired before the processing of objects. This enables you to create complex processes without writing code in Delphi, and, accordingly, without linking the report to a project.

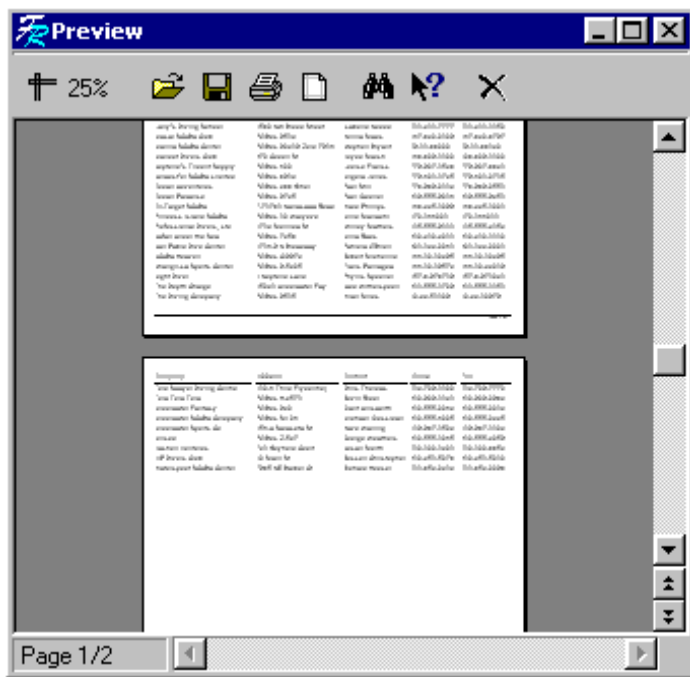


The possibilities of the built-in language of FastReport are very extensive. In scripts you can use all properties and methods of objects used in a report that are available, as well as variables and data fields. In scripts it is also possible to create variables and arrays, which then, on their turn, will be available in all report.

## Preparing reports

Prepared reports are products of activity of the FastReport kernel and can be previewed after clicking the "Preview" button. Unlike many report generators, which keep the content of report pages in metafiles (i.e. images in EMF format), in FastReport the prepared report presents a set of objects, describing the content of each page of the processed report. This allows you to modify the prepared report, by loading the necessary page in the designer. Besides, it is possible to describe the reaction on mouse clicks on objects in the preview window. This allows for easily organizing your work (clicking on a report object causes the generation of a new report with more detailed information).

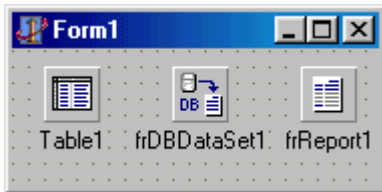
The preview window of FastReport differs from the ones used in other report generators. In particular, in FastReport viewing documents is used as in Microsoft Word: in a window it is possible to immediately see several pages. Besides that you can also search throughout the text in all the documents.



## Example: building a simple report


Let's build one of the simplest reports in FastReport:

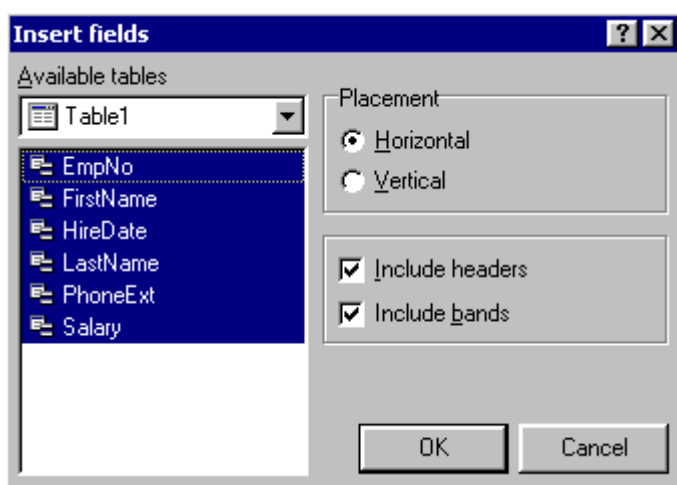
1. place three components on a form: TTable (or a TQuery), TfrDBDataSet and TfrReport. Your form should look like this:



2. link the TTable component with a database (set its DatabaseName and TableName properties).
3. link the DataSet property of the TfrDBDataSet component with the TTable object. At this stage the forms \*.dfm code will look like this:

```
object Form1: TForm1
  Caption = 'Form1'
  object frReport1: TfrReport
    Left = 16
    Top = 8
  end
  object frDBDataSet1: TfrDBDataSet
    DataSet = Table1
    Left = 56
    Top = 8
  end
  object Table1: TTable
    Active = True
    DatabaseName = 'DBDEMOS'
    TableName = 'employee.db'
    Left = 96
    Top = 8
  end
end
```


4. start the report designer (double-click on the TfrReport component);
5. click on the button "Insert data fields"  on the toolbar;
6. choose the necessary fields in the dialog and press OK button.



Now the report contents all the fields which you've choosed:

Page header				
Addr1	Company	Contact	FAX	Phone
Master data				
Form.Table1."Addr1"	DialogForm.Table1."Company"	logForm.Table1."Contact"	n.Table1."FAX"	rm.Table1."Phone"

It is possible to use automatically generated report as a template, which can also contain graphics, headlines or footers, numbers of pages, etc.

To run the report, press the "Preview" button  on the toolbar. FastReport now builds the report and shows its content in the preview window:

Preview					
60%					
EmpNo	FirstName	HireDate	LastName	PhoneExt	Salary
2	Roberto	26.12.88	Nelson	250	40000
4	Bruce	26.12.88	Young	233	55500
5	Kim	06.02.89	Lambert	22	25000
8	Leslie	05.04.89	Johnson	410	25050
9	Phil	17.04.89	Forest	229	25050
11	K. J.	17.01.90	Weston	34	33292,9375
12	Terri	01.05.90	Lee	256	45332
14	Stewart	04.08.90	Hall	227	34482,625
15	Katherine	14.06.90	Young	231	24400
20	Chris	01.01.90	Papadopoulos	887	25050
24	Pete	12.09.90	Fisher	888	23040
26	Ann	01.02.91	Bennet	5	34482,8
29	Roger	16.02.91	De Souza	288	25500
34	Janet	21.03.91	Baldwin	2	23300
36	Roger	25.04.91	Reeves	6	33620
37	Willie	25.04.91	Stansbury	7	38224
44	Leslie	03.08.91	Phong	216	40350
45	Ashok	01.08.91	Ramanathan	209	33292,94
46	Walter	09.08.91	Steadman	210	19599
52	Carol	02.10.91	Nordstrom	420	4500
61	Luke	16.02.92	Leung	3	34500
65	Sue Anne	23.03.92	O'Brien	877	31275
71	Jennifer M.	15.04.92	Burbank	289	45332
72	Claudia	20.04.92	Sutherland		35599
83	Dana	01.06.92	Bishop	290	45000
85	Mary S.	01.06.92	MacDonald	477	35599
94	Randy	06.06.92	Williams	892	28900
105	Oliver H.	06.10.92	Bender	255	35799
107	Kevin	01.02.93	Cook	894	35500
109	Kelly	04.02.93	Brown	202	27000

Page 1/1

# The FastReport kernel

Delphi component palette

FastReport objects

Report types

## Delphi component palette
















After installing FastReport, two additional tabs will be created in the Delphi component palette: the "FastReport" tab and the "FR Tools" tab. The first tab contains the main FastReport components, such as TfrReport, TfrDesigner, etc. The second tab contains additional components used by FastReport which can also be used in your applications.

### "FastReport" tab













Icon	Name	Description
	TfrReport	This is the main report generator component. To edit your reports, double click on this component at design-time to open the Report Designer window. <i>This component is required for all reports. It defines the design and layout of the report.</i>
	TfrCompositeReport	The composite report component. This is used when you need to combine several reports into a single report. For this type of report, you must provide a list of TfrReport objects that will be combined. <i>This component is only required if you are building composite reports.</i>
	TfrDBDataSet	This component provides a data source connection for the report. This data source can either be obtained from a Ttable (via its DataSet property) or a TDataSource (via its DataSource property). <i>This component is optional but is required to connect to a database. One instance of this component is required for each data-enabled band in the report.</i>
	TfrUserDataSet	This component provides a data source connection for the report. This differs from the TfrDBDataset in such that you must provide the data programmatically using events. This data source allows you to print virtually anything such as an array, a string grid or a text file. The OnFirst, OnNext and OnCheckEOF events are defined for record navigation of the data source. <i>This component is required only when you need to print reports where the data source is not a supported database format. One instance of this component is required for each data-enabled band in the report.</i>
	TfrOLEObject	Additional FastReport objects. The functions for each of these objects are discussed below. <i>These components must be dropped onto the form if your report uses any of their functionality.</i>
	TfrRichObject	
	TfrCheckBoxObject	



	TfrShapeObject	
	TfrBarcodeObject	
	TfrChartObject	
	TfrRoundRectObject	
	TfrTextExport	Export filters. These components are used to export your reports to one of the supported external file formats. (Currently: Text, RTF, CSV and HTML). <i>Only required if you need export functionality.</i>
	TfrRTFExport	
	TfrCSVExport	
	TfrHTMExport	
	TfrDesigner	This is the end-user, run-time report designer. <i>Only required if you need the report designer at run-time.</i>
	TfrDataStorage	This component allows the end-user to create tables and queries. <b>This component is obsolete. Use TfrDialogControls and TfrBDEComponents instead.</b> <i>Only required if you want to give the end-user the ability to create tables and queries.</i>
	TfrPreview	This component is required if you want to create your own preview windows.
	TfrPrintTable	This component prints a TTable or a TQuery's contents on-the-fly.
	TfrPrintGrid	This component prints a TDBGrid's contents on-the-fly.
	TfrDialogControls	This add-in component contains a set of dialog controls that can be used to create dialog boxes at run-time. <i>Only required if your reports contain internal dialog forms.</i>
	TfrBDEComponents	This add-in component contains a set of database access objects such as TTable, TQuery and TDataBase. <i>Only required if your reports contain internal database components.</i>

## "FR Tools" tab



Icon	Name	Description
	TfrSpeedButton	An enhanced TSpeedButton. Can have the "flat" look (Flat property) and you can optionally have it change its color when the mouse moves over it (GrayedInactive property).
	TfrDock	Used in creating dockable toolbars.
	TfrToolBar	A toolbar component that simulates the MS Office toolbar.
	TfrTBButton	Toolbar button.
	TfrTBSeparator	Toolbar separator.
	TfrTBPanel	A toolbar area that can contain other Delphi controls like TComboBox, TEdit, etc.
	TfrOpenDBDialog	"Open DB table" dialog box. Displays a modal dialog box for selecting a BDE alias.
	TfrComboBox	A flat TComboBox.
	TfrFontComboBox	A flat TComboBox that contains the list of installed fonts.
	TfrComboEdit	An edit control with a customizable button at the right.

Let's take a quick look at the main FastReport components.



## TfrReport component

This is the main FastReport component. It contains methods for loading, saving, previewing and printing reports. Each TfrReport component can only contain a single report.

### TfrReport properties.

Property	Default value	Description
DataSet	-	Points to a TfrxxxDataSet. The number of records in this data source defines how many times the report will be built and printed. (see also: ReportType property)
GrayedButtons	False	If True, the toolbar buttons of the designer and preview windows will be displayed in grayscale.
InitialZoom	pzDefault	Defines the initial zoom value of the preview.
MDIPreview	False	Displays the preview window as a MDI child window.
ModalPreview	True	If True, the preview window will be modal.
ModifyPrepared	True	If True, the prepared report can be modified under the preview window by double-clicking on the page.
Preview	-	Points to a TfrPreview. If this property is set, the prepared report will be shown in this component.
PreviewButtons	All	Defines the set of buttons that will be available in the preview window.
ReportType	RtSimple	Defines how to interpret the data source connected to the DataSet property. If ReportType = rtMultiple, the report will be built as many times as the number of records in the connected data source. This is useful for printing a report multiple times based on a list (i.e. the data source).
ShowProgress	True	If True, displays a progress window when preparing, printing or exporting reports.
StoreInDFM	False	If True, stores report in the DFM file. <i>Note: Reports saved in DFM files will not be modifiable since it will be stored in the executable file (EXE)!</i>
Title	-	The report title. This is the name displayed in the preview window and is the title assigned to the print job.

**Note: TfrReport.StoreInDFM property is False by default. It means that your report must be stored in a external file or alternately, in a database BLOB or binary field.** If you want to store your report in the DFM resource (like in other reporting tools such as QuickReport and

ReportBuilder), set this property to True. But remember – this will require you to recompile your application whenever a report is modified!

### TfrReport events.

Event	Description
OnBeforePrint	This event will be fired before printing the report. <i>Note: FastReport objects are not components, so they are not visible in Object Inspector. You will not be able to assign event handlers to each object individually. Instead, you can use common event handlers like OnBeforePrint, OnBeginBand and OnEndBand.</i>
OnBeginBand	This event is fired before printing a band.
OnBeginColumn	This event is fired before printing a cross-tab column.
OnBeginDoc	This event is fired at the beginning of a report.
OnBeginPage	This event is fired at the beginning of a page.
OnEndBand	This event will fire after printing a band.
OnEndDoc	This event will fire at the end of a report.
OnEndPage	This event will fire at the end of a page.
OnGetValue	This event is fired when FR finds an unknown variable in the expression. The assigned event handler must return the value for this variable.
OnManualBuild	Assigning a handler to this event will enable you to build the report manually. (i.e. show necessary bands programmatically in code). See demo: DEMOS\MANUAL.
OnMouseOverObject	This event fires every time the mouse moves over any object in the preview window. The event handler should return the cursor type (e.g. "pointing hand" cursor) for this object. This enables the user to see which objects are "clickable" and which are not.
OnObjectClick	This event will fire when you've clicked on an object in the preview window.
OnPrintColumn	This event will fire before printing a cross-tab column. The event handler can return the width of the column.
OnProgress	This event fires periodically during long time operations. The event handler can display the progress of the work done so far.
OnUserFunction	This event is fired when FR finds an unknown function in the expression. The event handler should return a value of this function.

**Basic TfrReport methods.**

Method	Description
LoadFromFile, LoadFromStream, LoadFromBlobField	Loads a report from an external file, a stream or a BLOB field.
SaveToFile, SaveToStream, SaveToBlobField	Saves the report to an external file, a stream or a BLOB field.
DesignReport	Runs the report designer. <i>You should include the designer component (TfrDesigner) in your application to have this functionality.</i>
ShowReport	Builds the report and displays it in the preview window. <i>Note: After closing the preview window, the prepared report will be destroyed and saving it to a file, exporting or printing will not be possible.</i>
PrepareReport	Builds the report without previewing. <i>This method should be called before calling ShowPreparedReport, PrintPreparedReport, PrintPreparedReportDlg, ExportTo or SavePreparedReport methods.</i>
LoadPreparedReport	Loads a previously prepared report (.FRP file) from an external file.
SavePreparedReport	Saves the prepared report to an external file.
ShowPreparedReport	Shows a previously prepared report.
PrintPreparedReport	Prints a previously prepared report.
PrintPreparedReportDlg	Displays the print dialog before printing a previously prepared report.
ExportTo	Exports a previously prepared report using one of export filters.



## **TfrDBDataSet component**

The non-visual component TfrDBDataSet is used to move through a dataset with methods like First, Next and Prior. It is similar to ReportBuilder's TppBDEPipeline component but is used only for navigation and does not provide field data to the report.

Property	Description
CloseDataSource	Closes related datasource after the report is built.
DataSet	A DataSet like TTable or TQuery.
OpenDataSource	Opens the datasource before building the report.
RangeBegin	First record in the dataset.
RangeEnd	Last record in the dataset.

The DataSet property of this component links to a TDataSet component like TTable or TQuery. Set OpenDataSource and CloseDataSource to True to automatically open and close the related dataset.

RangeBegin and RangeEnd are used to select the range of records.

RangeBegin can take the following values:

- rbFirst – start record selection from the first record;
- rbCurrent – start record selection from the current record.

RangeEnd can take the following values:

- reLast – stop record selection at the last record;
- reCurrent – stop record selection at the record that was the current record when you started the report;
- reCount – stop after N records are selected. Number of records to select is stored in the RangeEndCount property.



## **TfrUserDataSet component**

This component is also used to navigate a dataset but it uses data structures such as array, StringGrid, external file and so forth, instead of a database, as the source for the data.

Event	Description
OnCheckEOF	Event handler takes one parameter - EOF of Boolean type. Set this parameter to True to stop navigation.
OnFirst	Event handler must set pointer to the first record.
OnNext	Event handler must set pointer to the next record.
OnPrior	Event handler must set pointer to the prior record.














In this case the Words "pointer" and "record" are allegorical and not literal. If you are working with data from an array, "record" is the row of the array and "pointer" is the variable that stores the current row number.

Navigation methods are called in the following manner: OnFirst, OnCheckEOF, OnNext, OnCheckEOF, OnNext, ..., OnCheckEOF, OnNext. If the report has groups, then the method OnPrior is also called after the group ends.

If you know the number of "records" before building the report, you can use the RangeEnd and RangeEndCount properties. Just set RangeEnd := reCount and RangeEndCount := number of records (for instance, number of array's rows). In this case, you can leave the OnCheckEOF event handler empty. To find the current row's position, use the RecNo: Integer property. At the First position RecNo = 0.

## FastReport objects



Icon	Name	Description
	Text	Provides a framed rectangle with multi-line text inside. Text also can contain variables.
	Band	Band area. Defines where the band's contents will be placed in the final report.
	Picture	Shows BMP, ICO, WMF, EMF and JPG picture formats. Source can be a BLOB field. Uncomment the corresponding line in FR.inc to enable JPG format in FR.
	SubReport	Intended for creation of subreports. When you insert this object into a report, you'll see that a new page is added to your report.
	Line	Draws horizontal or vertical lines on the report.
	Shadowed text	Provides Multi-line text inside a framed rectangle with optional shadow area and/or gradient fill. Useful for printing labels.
	Barcode	Shows data in barcode format.
	Shape	Allows insertion of geometrical shapes in the report (rectangle, rounded rectangle, ellipse, triangle).
	CheckBox	Shows Boolean data as either a checkmark or an X.
	RichText	Designed for the insertion of RTF (Rich Text format) documents into the report. Source can be a BLOB field.
	OLE	Designed for the insertion of an OLE object into the report.
	Chart	Designed for the insertion of charts or diagrams into the report. TeeChart component must be installed in Delphi for this to work.
	RichText 2.0	Similar to the RichText component but will also allow pictures and OLE objects inside the RTF text. Source can be a BLOB field. TrxRichEdit component and RX library must be installed in Delphi to use this component. Uncomment the corresponding line in FR.inc to enable in FR.





## The "Text" object

This object is undoubtedly the most powerful and flexible object in FastReport. Basically, it provides you with a framed rectangle of multi-line text. You can set the frame type, color and width; as well as the font attributes, text alignment and font rotation (horizontal or vertical). To set the object's attributes, use the "Text" and "Rectangle" toolbars:



The contents of a text object basically consist of a memo type object, which may include: text, variables, data fields, or any combination of these. The font formatting will be applied to *all text* contained within the text object.

### Some examples of using text objects:

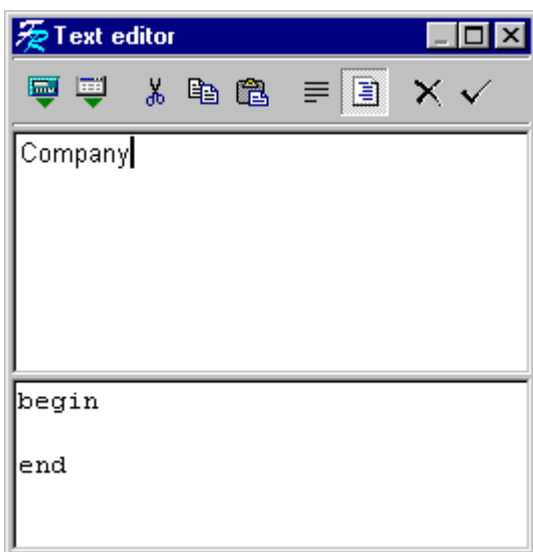
*Length, cm: [Length]* - This shows some **fixed text** (Length, CM) followed by a **variable** ([Length])

*Length, cm: [Table1."Length\_cm"]* - This shows some **fixed text** (Length, CM) followed by a **data field** ([Table1."Length\_cm"])

*Length, cm: [[Length inch] \* 2.54]* - This shows some **fixed text** (Length, CM) followed by a **variable used with a formula**, ( [[variable] \* value]). It is important to note the additional set of square brackets when using formulas with variables.

*Length, cm: [Table1."Length\_in" \* 2.54]* - This shows some **fixed text** (Length, CM) followed by a **data field used in a formula** ([datafield \* value]). Notice the single set of square brackets is required for data fields and formulas.









The fastest way to edit a text object is to select the desired rectangle (by clicking on it) and then double click on it in the report editor. This will bring up the memo editor dialog:



Here is a short description of the toolbar buttons:



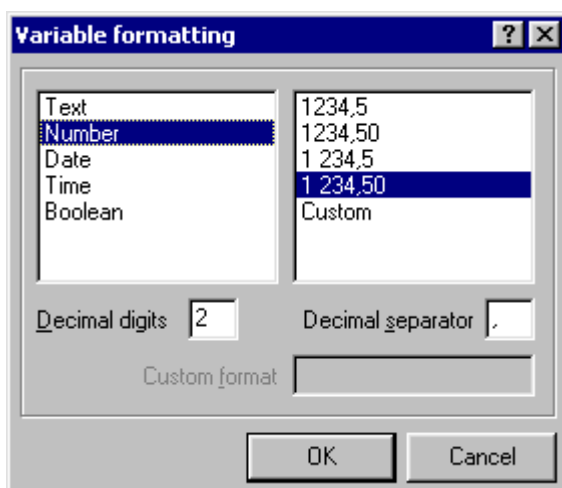
Inserts an expression;

-  Inserts a data field;
-    clipboard operations;
-  turns on/off word wrapping;
-  turns on/off the script editor;
-  Cancel button;
-  OK button.

From the editor you may type in text, insert variables, data fields or expressions. Clicking on the "Data field" or "Expression" buttons in the memo editor dialog will display the currently available data fields or variables. You may also use the following keys:

- Insert**            "Variable" button - brings up the variable dialog.
- Ctrl+Enter**      "OK" button - accepts and closes the dialog.
- Esc**                "Cancel" button - closes and cancels editing.

Each text object can also have its own formatting style. You can edit the format by clicking on the "DisplayFormat" property of the text object in the Object Inspector. Formatting options include: no formatting, displaying as a numeric value, date, time or boolean formatting.



For each formatting category, you can choose one of several predefined formatting options, or you may perform custom formatting (for example, #,##0.000 for a numeric value). Formatting is done the same way as Delphi formats strings (this is described in Delphi online help system, see the "Formatting strings" topic). Boolean values can be formatted by using the following formatting: **False\_string;True\_string**. (i.e. to show yes or no, use No;Yes as the format).

The formatting is applied to each and every variable in the memo. If a variable cannot be formatted, it will be shown as plain text. If you use several variables within a text object, but you want to use different formatting for each variable, you may override the default formatting by using the «#» tag followed by the formatting style. Put this formatting tag and the format string inside of the variable brackets, i.e.:

[Variable #format], where format is one of following values:


- x.x or Nx.x or Nyyyyy - numeric formatting. x.x - length of number/number of digits in fraction part; yyyyy - string like #,##0.00 (described in Delphi online help system, "Formatting strings" topic). If the x.x or yyyyy string contains any «.», «,», «-» characters, this character will be used as the decimal/fractional separator.

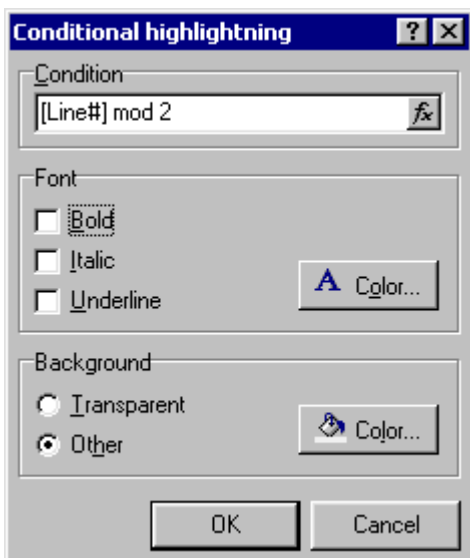
- Dxxxxx, Txxxxx - date and time. xxxxx – a string like dd.mm.yy.
- Bxxxxx;yyyyy - boolean formatting. If the value is False, the xxxxx string will be shown; otherwise the yyyyy string will be shown.

Here are some examples of using the «#» tag:

[Table1.«N1» #9.2] [Table1.«N2» #N9-2] [Table1.«N3» #N#,##0.00] - numeric formatting  
 [Table1.«Date1» #Ddd.mm.yyyy] [Table1.«Time» #Thh:mm:ss] - date/time formatting  
 [Table1.«Bool1» #BFalse;True] [Table1.«Bool2» #BNo;Yes] - boolean formatting

You cannot use the formatting tag in expressions that have been created in the variables editor (see below).

For reports where you would like to have objects change their font color, background color etc. based on a **specific condition or expression**, click the  button in the Text formatting toolbar. This will bring up the “Conditional highlighting” dialog where you may enter the conditions to be satisfied in the text box and select the various options available.



For example, assume it is necessary to pick out orders exceeding \$1,000.00, Demo report "3-level" is an example of this. To achieve this, select the rectangle containing sum order and click on the highlight button in the formatting toolbar. In the textbox of the dialog type the condition “Value > 1000” without quotes. Select an appropriate font & background color then click on the Ok button. Your report will now show any orders over \$1000.00 highlighted with the chosen color. Try various combinations of font styles and background colors to achieve the results you need.

By right clicking on the "Text" object you can set the following options:

- **Stretched** – the object has a variable height depending on the actual number of text strings in it. You should also turn on this option in the object’s parent band. When this band is printed, it will calculate the maximum height of all objects with the Stretched option and stretch itself.
- **Word wrap** – long strings are wrapped to make several lines of text.
- **AutoWidth** – before drawing, the object calculates its actual width.
- **TextOnly** – do not process variables in the object.
- **Suppress repeated values** – do not show repeated values.

Also you can find some additional properties in the Object Inspector:

- **CharSpacing** – extra space between chars;

- **GapX, GapY** – left and top gaps;
- **LineSpacing** – extra space between lines of a text.



## The "Band" object

Like all other visual report designers, FastReport is a band-based report generator. This means you place bands on a report and then place data into the band. The table below lists the types of bands which FastReport currently supports and where they will appear in the final report.

**NOTE:** The bands do not have to appear in the correct order in the Report designer. The *band type* governs where they will be located on the final report. However, placing bands in the correct order makes it much easier to modify them later.

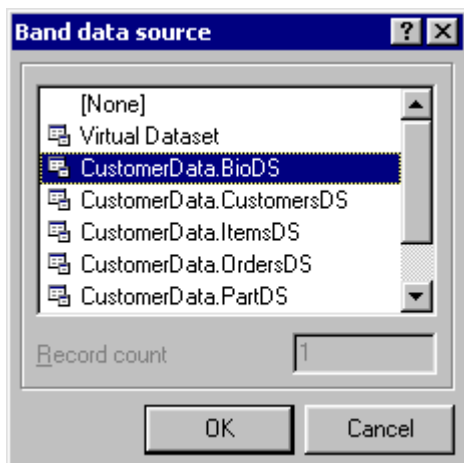
Name	Where and when
Report title	Printed at the beginning of the report
Report summary	Printed at the end of the report
Page header	Printed at the top of each page
Page footer	Printed at the bottom of each page
Master header	Printed at the beginning of the first data level
Master data	Data of the 1 <sup>st</sup> data level– repeated for each master data record
Master footer	Printed at the end of the first data level
Detail header	Printed at the beginning of 2 <sup>nd</sup> data level
Detail data	Data of the 2 <sup>nd</sup> data level– repeated for each detail record
Detail footer	Printed at the end of 2 <sup>nd</sup> data level
Subdetail header	Printed at the beginning of the 3 <sup>rd</sup> data level
Subdetail data	Data of the 3 <sup>rd</sup> data level– repeats once for each sub-detail record
Subdetail footer	Printed at the end of 3 <sup>rd</sup> data level
Overlay	Printed on each page using a lower page layer (useful for printing watermarks)
Column header	Printed at the beginning of the column
Column footer	Printed at the end of the column
Group header	Group title printed at the beginning of group
Group footer	Printed after group
Cross header	This group of bands is designed for creating cross-tab reports

Cross data Cross footer	which have a variable amount of columns on the page.
Child	This band can be added to another band of any type (except Cross and Page footer). Child band is showed after parent band.

As you can see, the set of bands is different from those generally accepted. In the classical scheme (in particular, in ReportBuilder) there is only one data-band Detail.

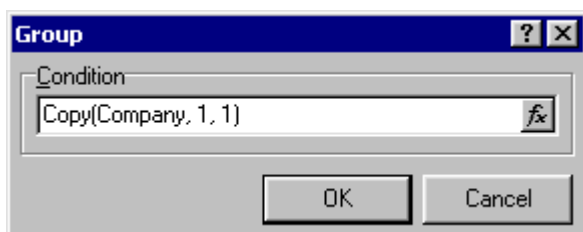
Depending on the band type, FastReport shows appropriate editor.

For bands which display data from a data source (the detail data band, master data band, etc.), you will be asked where this band will obtain its data from. You will be presented with a list of all available datasets or you can select the "virtual" dataset option.



When you select the virtual dataset option, you must specify the number of “records” contained within the “virtual dataset”. The band then behaves as if the specified number of records actually exists. Virtual datasets can help you print forms, where multiple lines need to be printed, but you don’t want to manually insert each one. You can simply design one line and repeat it the required number of times using a virtual dataset.

If you select the group header band type, you will be presented with the group header editor. This editor allows you to define the grouping condition, based on the fields in the table or on any other expression (i.e. group the data based on the first character of the surname, as used in telephone directories).



If you right-click on a band, you will also see a number of options in the context menu. These include:

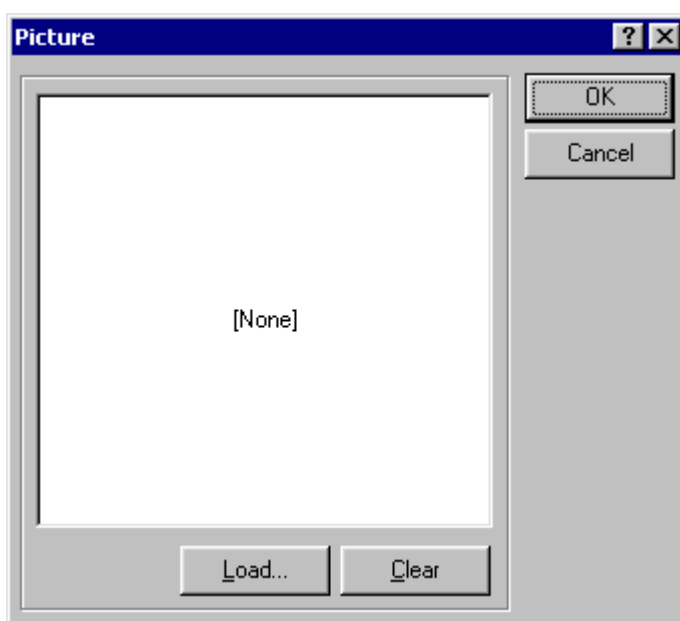
- **Stretch** – This means the height of the band is determined by the tallest object within the band. “Text” objects will be stretched as required and the word-wrap option should also be enabled. This allows for automatic adjustment of the band (row) height as required so that all data fits into the row.
- **Breaked** – This will try to fill any unused space on the page before creating a new page.

- **Force new Page** – This will force the band's contents to be printed on a new page.
- **On first page** – The band should print on the first page (applicable to page header & page footer bands only).
- **On last page** – The band should print on last page (applicable to page footer bands only).
- **Repeat on all pages** – This option is only available for Master header, Detail header, Sub-detail header, Group header and Cross header bands. If the amount of data below these headers forces a new page, these headers will also be included on that page.



## The Picture object

The Picture object is used for inserting graphics into a report. Image formats supported include BMP/WMF/ICO. JPG and GIF images can also be inserted but are not directly supported as they require additional Delphi graphic support libraries. These libraries are freely available elsewhere.



The Picture object Editor is used to choose an image, clear an image, or select an image from a BLOB field of a database table. To insert an image from a file, double-click on the Picture Object. To insert an image from a BLOB field, select the Picture object, press Ctrl + Enter to invoke the Text Editor, then reference the fieldname of the BLOB in the memo editor, for example: [Table1.GraphicField]. This operation can also be performed by selecting from the Insert Data Field in the Text Editor.

If encrypted images in a database are to be printed directly, use the OnBeforePrint event to decrypt the image and load it into the report at run-time. An example using ADO and SQL Server is given later in the manual.

If images to be printed are stored on disk, use the TfrReport.OnBeforePrint to load and the images before printing. An example of printing thumbnails for a given directory can be found at the end of the manual.

The context menu of the picture object allows the following options to be set:

- **Stretch** - image will stretch to fill the rectangle.
- **Maintain Aspect ratio** - if stretch is selected image proportions are preserved.
- **Center** - centers the image in the rectangle.

- **BlobType** – type of BLOB stream: BMP, WMF, ICO or JPG. This is necessary because Delphi can not handle stream type automatically. Use the Object Inspector to set this property.



## The SubReport object

A Subreport object is used as a placeholder for inserting additional reports at a particular point in a report. The report which will be printed within the Subreport must be placed on a **separate page** within the main report. An example of when to use a Subreport object is inserting a chart below or next to some sales figures.

When inserting a Subreport object into a report, an additional page is automatically inserted into the main report. The Subreport object automatically points to this newly created page. Anything added to the new page will be printed within the Subreport object instead of being shown on the main report.

Subreport objects can be placed next to or below the main report. If multiple Subreport objects are to be placed below one another, insert them into separate databands.

There are some limitations when using Subreports:

- Columns can not be used
- The following bands can be used in a Subreport (but not in the associated main report): Report Title, Report Summary, Page Header, Page Footer, ColumnXXX bands.
- Broken bands can not be used
- Groups can not be used within a Subreport.



## The Line object

The Line object can be used to insert horizontal or vertical lines into a report. Lines can be used when separating details within a report, making it easier to read. The thickness and color of the line can be adjusted using the drawing toolbar.

To draw a line, click on the Line toolbar button, move the mouse over the active page and the cursor will change to a pencil indicating a line is about to be drawn. Position the mouse at the point where the line is to start. Press and hold down the left mouse button and move drag to the point where the line is to end. Release the mouse button and the line is drawn. The line can then be selected for editing it as required.



## The CheckBox object

The Checkbox object is used to graphically show Boolean data and should only be linked to a Boolean variable or data field. When the value of the variable or field is True, the cross will be displayed. If the value is False, nothing will be shown.

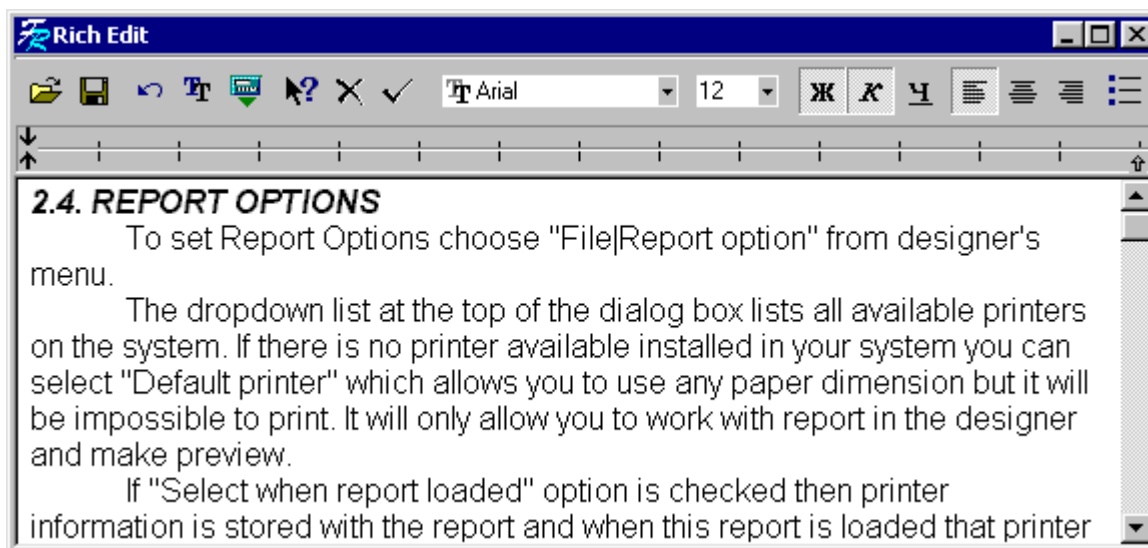
**Note:** If a Checkbox object is inserted into a report, a TfrCheckBoxObject component must be inserted into the same form or an error message will be generated at run-time.



## The RichText object

The RichText object is used to print RTF (Rich Text format) documents. The RichText object should be used when formatted text is required on a report and a Text object can not be used.

The RichText object will provide a high level of text formatting control which the FastReport Text object cannot provide. RTF files can easily be created in programs such as Microsoft Word or the included RTF editor, which is based on the Delphi RTF editor, can be used. A RichText object will retain and render all the formatting in an RTF file.



The RTF editor included with FastReport provides for all the basic RTF operations on text. Variables can be inserted into the formatted text. These variables must be enclosed in square brackets (as in Text object).

The RichText object can also be used to print formatted text contained within a database BLOB field. To select the required field, press Ctrl+Enter to invoke the text editor then either enter the table field in square brackets , for example: [Table1."RichField"], or click on Insert Data Field and select the required datasource and field.

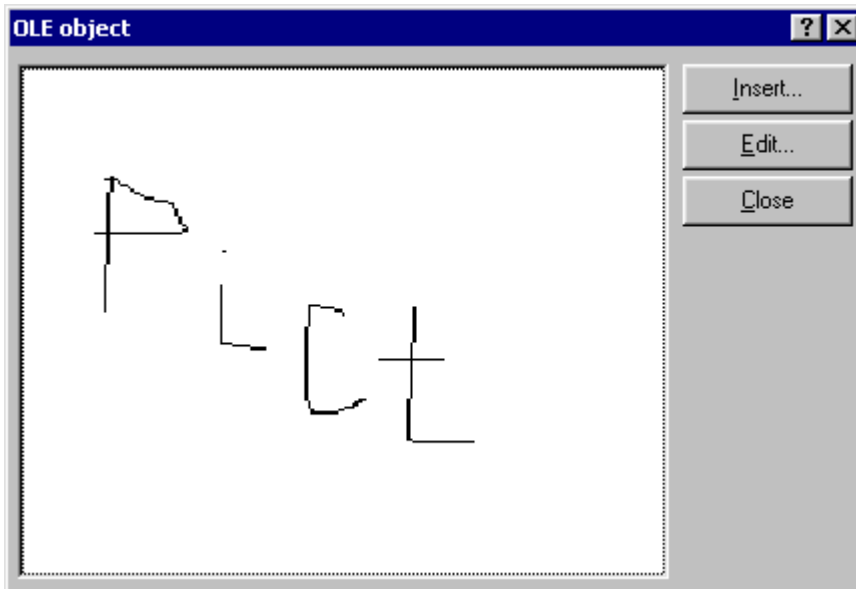
**Note:** A TfrRichObject object must be inserted onto the same form. If not included, an error message will be generated.



## The OLE object

The OLE object is used to insert an OLE object into a report. The OLE object's editor allows insertion of a new OLE object. Click on insert in the editor to invoke the standard OLE insert dialog window to be presented with a list of available OLE objects.





The OLE object can be used to utilise OLE objects contained with a BLOb field. To do this, press Ctrl+Enter to open the OLE Object inspector, select Insert, and find the OLE object required.

**Note 1.** The Stretched option in the context menu sometimes corrects the viewing of Excel data.

**Note 2:** a TfrOLEObject component must be used on the same the form. If not, an error message will be generated at run-time.

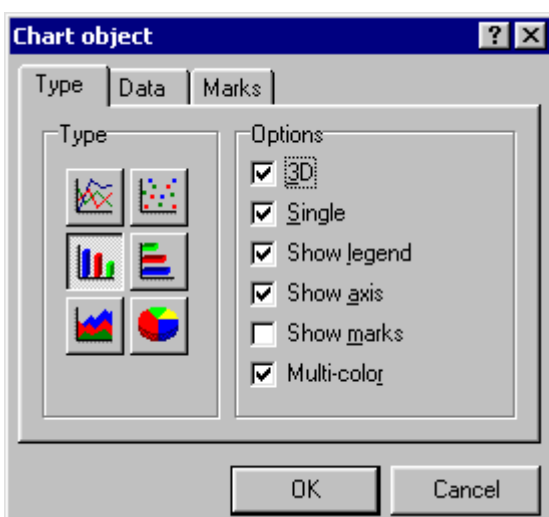


## The Chart object

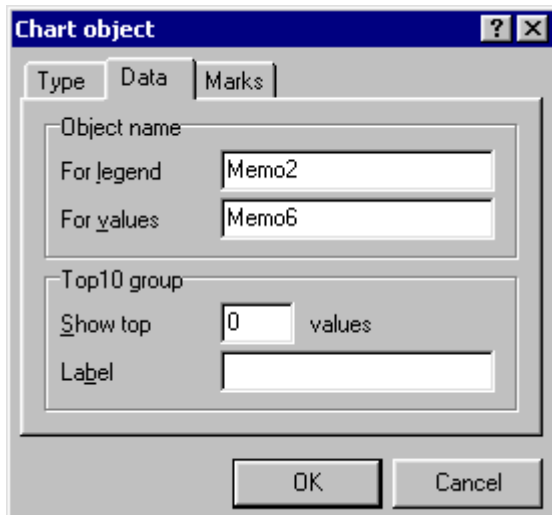
The Chart object is used to insert charts or diagrams into a report. This is especially useful for printing sales figures.

The Type tab of the chart editor is used to pick from six types of charts and their display options as follows:

- 3D – shows chart in 3-dimension;
- Single – should be turned on (reserved for further use);
- Show legend – shows legend near by chart;
- Show axis – shows axis (should be turned off if chart type is pie-chart);
- Show marks – shows marks;
- Multi-color – shows chart values with several colors.

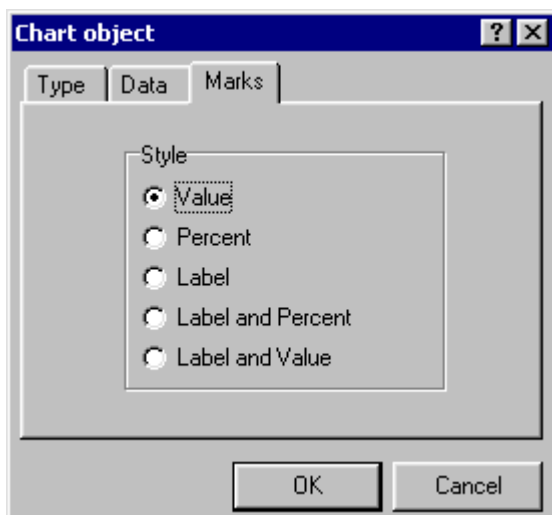


To link the chart to the data fields, you should set the names of two Text objects i.e. two data fields, already on the report. The contents of these fields will be used as the chart value and legend strings. The Legend will normally be in the X-axis, the Value in the Y-axis of the chart. When building the report, the contents of the select Text objects will be accumulated in the memo of chart object. Right Click on the Chart object, select Edit then Data, and the Legend/Data can be selected i.e. legend might be “memo2”, values might be “memo6”. (See demo for further example).



The Chart object allows the creation of ‘Top-10’ charts. This will chart represent several of the biggest values and summary of other values not included in chart. To do this, set the number of top values on Data tab of Chart editor and set the Legend name for non-included values (‘Others’ word is usual used).

The Marks tab of the chart editor allows the type of marker shown to be selected. ‘Value’ is the default. If any other selection is made the Marks option of the display tab must be set.



If the Text object being used as a Chart value contains a formatted value (for example, 10 000.00 or \$100.00), the Chart object attempts to expand the numeric value from this string. It skips all non-digit symbols at the beginning and end of the string, then skips all symbols – e.g. digit separators. If more advanced formatting is set (for example, 10000km2), these values can not be used as Chart values. Non-visible, non-formatted objects should be created for these values, and non-visible objects created used as the Chart values. Objects can be hidden by setting the Visible property of the object to False in the Object Inspector.

Chart values are accumulated in the Memo of the Chart object:

*Header1;Header2;Header3*  
*Value1;Value2;Value3.*

A Chart can also be built by using the Text Editor (Ctrl+Enter). Insert the appropriate values into the Memo of the object.

**Note 1:** The TeeChart component is available in Delphi 3 and above. To use in Delphi 2, install the TeeChart component first, then correct the FR.INC file and recompile the FR library.

**Note 2:** A TfrChartObject component must be on the the form. If not, an error message will be generated at run-time.



### **The Shape object**

The Shape object is used to insert geometrical shapes in a report. i.e. (rectangle, rounded rectangle, ellipse, triangle).

**Note 1:** When using the triangle shape, the background and fill colors default to white and these cannot be changed.

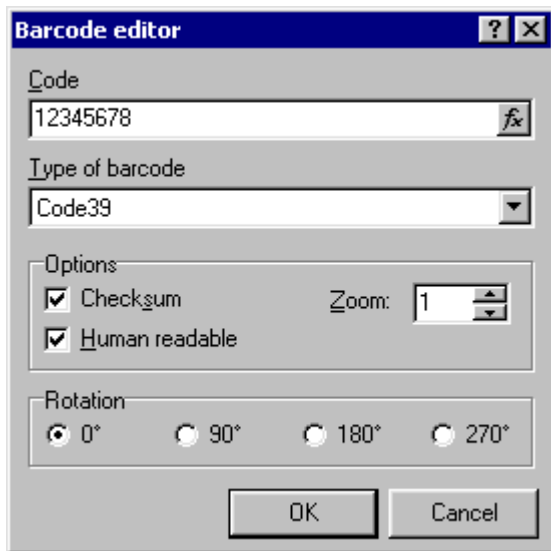
**Note 2:** A TfrShapeObject component must be on the form. If not, an error message will be generated at run-time.



### **The Barcode object**

The Barcode object is used for printing barcodes. The width of the barcode is determined automatically by the amount of data it represents. The following barcode formats are supported:

- 2 of 5 interleaved
- Code39
- Code39 Extended
- Code128A-C
- Code93
- Code93 Extended
- MSI
- PostNet
- Codebar
- EAN8
- EAN13
- EAN128A-C
- UPC A, E0, E1, Supp2, Supp5



**Hint:** If whitespace is needed around the barcode, set the **FrameTyp** property to 15 (this will draw a frame around the entire object), the **FrameWidth** to 6.00 and **FrameColor** to clWhite. This can also be achieved this using the border toolbars buttons.

**Note :** A TfrBarcodeObject component must be on the form. If not, an error message will be generated at run-time.






### The RichText 2.0 object



The RichText 2.0 object is similar to the RichText object except that it is based on RX Rich control. It can contain OLE objects and images inside RTF text. This object can be used if the RX library installed (required RXLib ver.2.60 or above). To activate this object, remove the comment from line {\$DEFINE RX} of file FR.INC and recompile FR packages.

**Note:** A TfrRxRichObject component must be on the form. If not, an error message will be generated at run-time.




## Building the report

To start building report, use TfrReport  object from FR component palette. It can store one report, load and save it, run, preview and print. Double-click on this component at design-time and you will be presented with report designer.

The most of reports contains data from databases. To access this data, FR uses standard Delphi db-aware components TTable  and TQuery . Of course, you can use ANY other components based on TDataSet and TField.

FR, as all other report generators, uses bands to logical grouping report data. Bands are divided on two categories: data-bands (master data, detail data, ...) and other (report title, page header, ...). Data-bands presents the multi-line part of a report, for instance, the rows of the TTable. That's why you should assign a dataset to such band. FastReport has two components for this: TfrDBDataset  and TfrUserDataset  from FR component palette. Actually TfrDBDataSet component is the connector between db-aware component like TTable and between the data-band. The TfrUserDataset component is used when you deal with non-DB data such as array, grid etc.

Well, before building a report, you must:

- a) place TfrReport  component on the form;
- b) place db-aware components like TTable, TQuery on the form, if your report will use DB data;
- c) for each data-band level used in your report you must create connector (TfrDBDataset  or TfrUserDataset ) and tune its properties.



## Bands in the FastReport

FastReport has 22 bands. All they listed here.

Name	Where and when
Report title	Printed at the beginning of the report
Report summary	Printed at the end of the report
Page header	Printed at the top of each page
Page footer	Printed at the bottom of each page
Master header	Printed at the beginning of the first data level
Master data	Data of the 1 <sup>st</sup> data level– repeated for each master data record
Master footer	Printed at the end of the first data level
Detail header	Printed at the beginning of 2 <sup>nd</sup> data level
Detail data	Data of the 2 <sup>nd</sup> data level– repeated for each detail record
Detail footer	Printed at the end of 2 <sup>nd</sup> data level
Subdetail header	Printed at the beginning of the 3 <sup>rd</sup> data level
Subdetail data	Data of the 3 <sup>rd</sup> data level– repeats once for each sub-detail record
Subdetail footer	Printed at the end of 3 <sup>rd</sup> data level
Overlay	Printed on each page using a lower page layer (useful for printing watermarks)
Column header	Printed at the beginning of the column
Column footer	Printed at the end of the column
Group header	Group title printed at the beginning of group
Group footer	Printed after group
Cross header Cross data Cross footer	This group of bands is designed for creating cross-tab reports which have a variable amount of columns on the page.
Child	This band can be added to another band of any type (except Cross and Page footer). Child band is showed after parent band.

As you can see, the set of the bands differs from “classic” schema that is used in the ReportBuilder (here and after – ReportBuilder 4.x). On this schema, the report itself has a dataset and it presents the first data level. The second data level is formed by the detail band. In other words, classic schema allows you to create master-detail reports. If you want more detail levels, you obliged

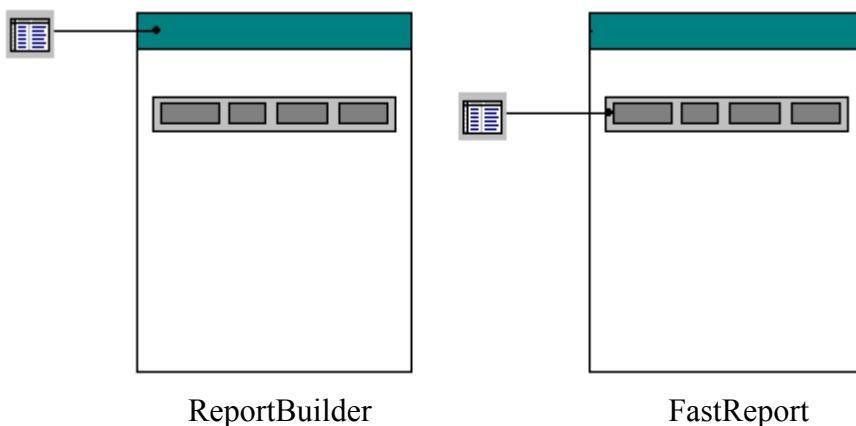
to use sub-report which is actually another report placed on the separate page. Sub-report also used for master-detail-detail reports.

With the schema that is used in the FastReport, you may not use subreports. As you can see in the table, you can use up to three data levels in one report (bands master data, detail data and subdetail data). Each data-band that presents one level of the data, must have a connector  or TfrUserDataset  assigned to it. If you want more levels, you can use subreports.

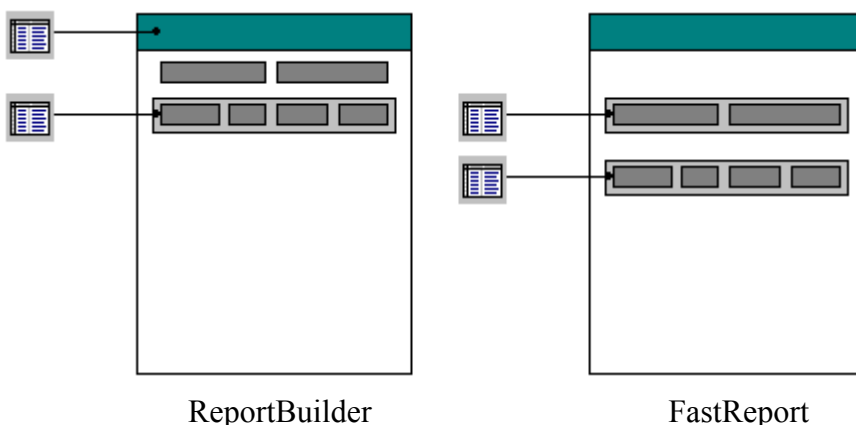
You can have several data-bands in one report. For instance, you can have two master data bands on the same page – it allows you easily create “master-master” report.

For illustration aforesaid we will show an instance of building some reports in FastReport and ReportBuilder.

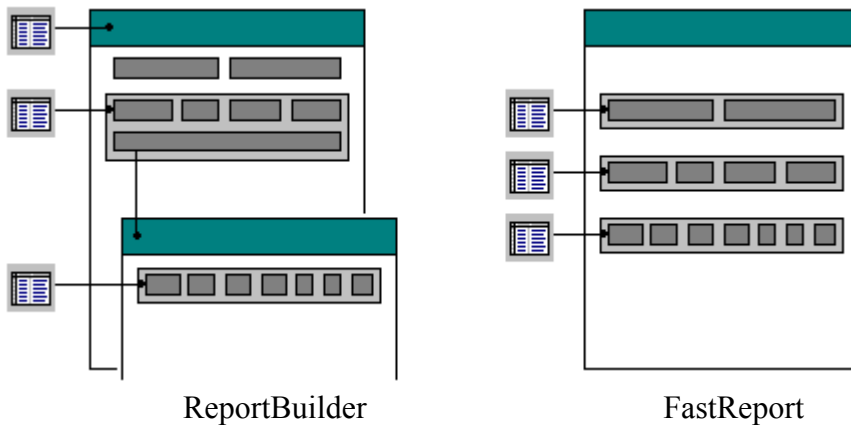
Example 1. Report with one level of data.



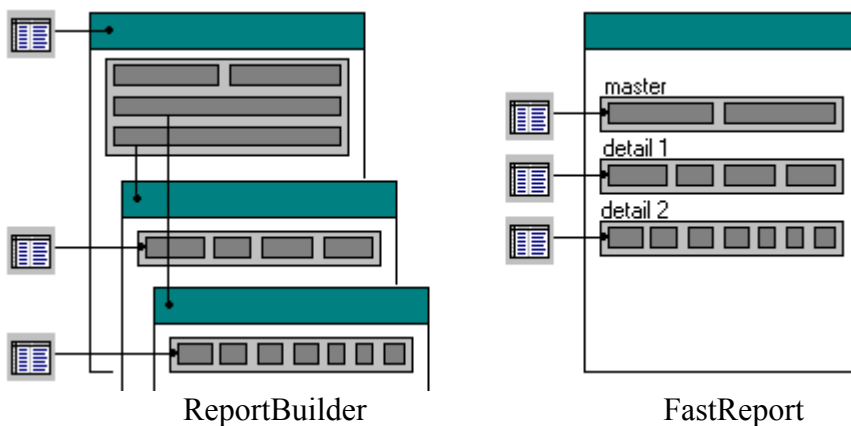
Example 2. Report with two level of data.



Example 3. Report with three data levels.



Example 4. Report of “master-detail-detail” type.



The schema used in FastReport provides more flexibility in formation the reports, as we see. Besides, FastReport allows to assign to the object “Report” data source. The object TfrReport has some corresponding features for it: ReportType (rtSimple,rtMultiple) and DataSet: TfrDataSet. If the ReportType = rtSimple (default), you may not assign data source to the band master data - it will be taken from the DataSet feature of TfrReport object. Otherwise (ReportType = rtMultiple) the report would be serially built so many times, as the number of notes in the data set, connected up to the DataSet feature. It is convenient to use for printing several copies of the report, if it is required to mark copies somehow (e.g. to write down on the first copy “Primary” and “For acquaintance” on the rest copies).

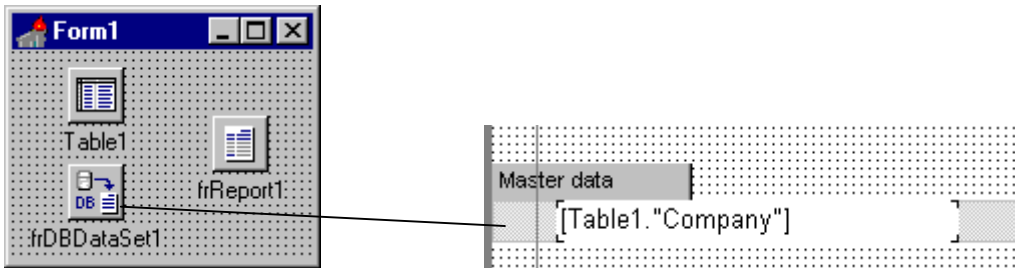
In addition to primary bands there are two auxiliary bands, not included into the classic schema - Overlay and Child. These bands are borrowed from QuickReport. Overlay band is used for printing background drawings (or other objects), which are output as the lowest layer; Child band could be attached to any band, except for Page footer and be output after it. You can also attach your Childs to the Child band. It is provided mainly for printing multisectional bands (i.e. bands consisting of several sections). Each section is formed by one Child band and can have variable height, depending on the height of the objects being in the section.



Master data	
[Company]	
Child	
[Address]	
Child	
[Contact]	

## Simple report (list)

This is a simplest kind of report. For creating it, you should place «Master data» band on the page, then place the required objects on it:



You also should connect «Master data» band to appropriate dataset (TfrDBDataset or TfrUserDataset components). Example of this and other reports you can see in demo.

## Master-detail report

For creating this report, place «Master data» and «Detail data» bands on the page, then place objects with appropriate data on these bands. Tables that used in this report must be linked by master-detail relationship.

Does not matter, in which order you place bands on the page - «Master data» band will be printed first. If appropriate detail list is empty, master record will be skipped. If you don't want this, turn on the option «Print if detail empty» of «Master data» band.

You can print new master data on new page - just turn on «Start new page» option of the band.

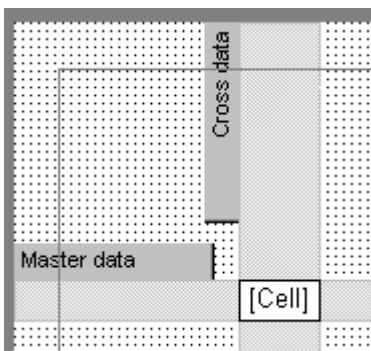
## Master-detail-subdetail report

For creating this report, place «Master data», «Detail data» and «Subdetail data» bands on the page, then place objects with appropriate data on these bands. All principles are like to the master-detail report.

## Cross-tab report

This report is intended for printing table with variable number of columns. During report building, all off-bounds columns will be printed on new page (like in MS Excel).

For creating this report, place «Master data» and «Cross data» bands on the page. Place object in the cross of these bands. This object will be printed as the cell of cross-table.



That is everything what is required for Base Report. You should only assign data sources for “master data” and “cross data”. If “cross data” band crosses several ordinary bands, it is necessary to assign data sources for each band. As you see you need at least two data sets, connected with each other by master-detail relation.

This kind of report can have variable height of data line. You have to enable Stretched option in the “master data” band. The cross table will be generated in two passes: maximum line height is defined in the first pass, data output - in the second one. The current FastReport version does not provide building cross-tab reports with the complicated headers (as in Report Builder). It will be possible in one of the next versions of FastReport.

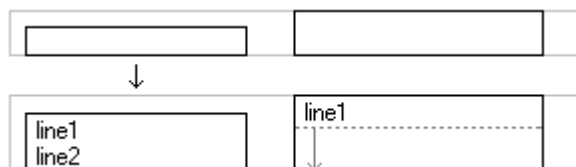
## Dynamic reports

Band height in Dynamic Reports depends on the objects included into the bands. The objects that can stretch depending on the length of the text inside them are the following: “Text”, “Text With Shadow” and “RichText”. The “Stretched” flag must be enabled to allow these objects to stretch. Besides you need to enable the same flag in the band, where the object is placed. When printing a stretched band, FastReport reads the maximum height of the object, located in the band and set the height of the band equal to the maximum height of the objects.

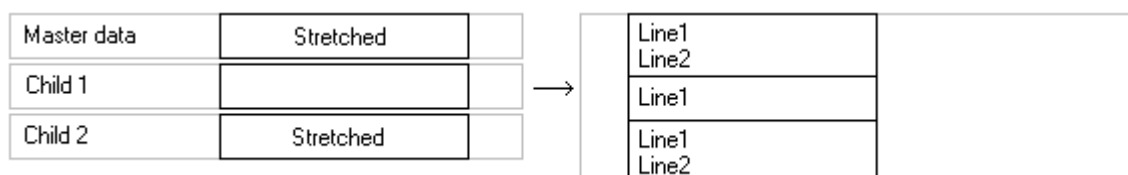


Stretched object and object with fixed height.

If there are several stretched objects side by side, the lower boundaries of all objects (in spite of their actual height) would coincide with the lower boundary of the band.



And what would you do if the stretched objects need to be put under each other or if non-stretched objects are put under stretched ones? Regions and StretchWithParent and ShiftWithParent object options are used for it in the Reportbuilder. Objects, placed under each other, one or several of them being stretched, need to be transferred to different bands.



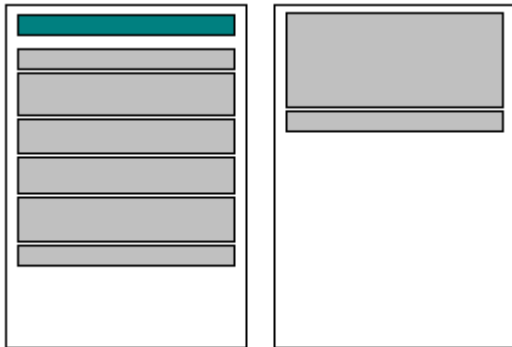
For all that Child Bands you have to set the “Stretched” flag. “ChildBand” feature in the object inspector serves for bands binding. In the above-mentioned example you must bind Master data with Child1 and Child1 with Child2.

It should be noted that all objects of FastReport have frames. By applying frames you can easily obtain a table-like view. When an object is being stretched, the frame stretches with it. It is very

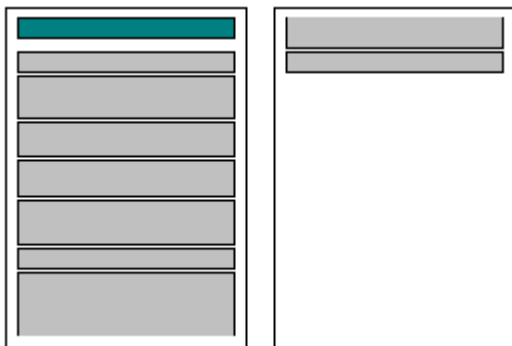
convenient, because text objects in the ReportBuilder have no frame and you need to do the frame using the “Shape” object. There are some difficulties in it - you have to align the frame and the object. Besides you can’t always join the frames of adjacent objects when printing. The frame is available in QuickReport but does not stretch together with the object when it stretches.

## Broken bands

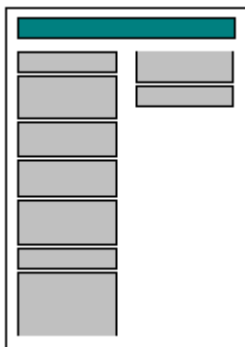
While printing a band FastReport checks, if there is enough space in the page and if this space is not enough, it generates a new page and print the band on it. Moreover the space in the page is not spent sparingly, especially in the cases when band height is big enough:



In order to print so many text lines in the sheet as possible, the band must have not only the “Stretched” option enabled, but also the “Broken” option enabled. Thereafter the Report would look approximately like:



If the Report is multicolumn, the contents of the objects will be transferred to the next column:



“Text”, “RoundRect” and “RichText” objects can break the contents. All other objects will be taken out to the page, where enough free space is available.

## Multicolumn report

In an ordinary report the generation continues from the next page after completing the previous one. In a multicolumn report it takes place at the same page but next to the first column, i.e. in several adjacent columns. The number of columns is set in the page option. You can convert an ordinary report into a multicolumn one only setting column count in the page options. “Column header” and “column footer” bands provides each column with a header and a footer.

Besides you can set column count separately for each data band in FR2.4. If we set Column >1 feature for a band, data lines output will take place from right to left and from top to bottom. You can also set column width and column gap. Bands display current sets in the designer with dash lines.


## Report with title page

It is easy to prepare multipage reports (i.e., the first page - title page, all the rest - other information) with FR. In order to create/delete the page, use keys of the panel or context menu, which appears after pressing the right button of the mouse at the page bookmark.

In essence multipage reports represents several reports in one. Each page may include its own report with its own band set and page relocation. Composite Reports are used to build similar reports in QR and “SubReport” is used in ReportBuilder.

Page contents could be put in the free space of the previous page, if the “PrintToPrevPage” option (PrintToPrevPage feature in the inspector) is enabled in the page relocation.

## Nested reports (subreports)

Nested reports are reports with «Subreport» object  placed in them. «Subreports» objects are references to other reports, placed in other pages. During report building, instead of «Subreport» object the appropriate report will be printed. When you insert a subreport object, you'll see that a new page will be added to your report.

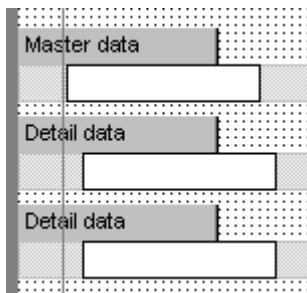
Subreport objects can be placed side-by-side. If you want to place subreport object one-under-other, spread it on separate bands (for instance, you may use child bands to do this).

There are some limitations on using subreports:

- Do not use columns in subreports.
- Do not use ReportTitle, ReportSummary, PageHeader, PageFooter, ColumnXXX bands.
- Do not use broken bands.
- Do not use groups.


## Master-Detail-Detail report

For creating this kind of report, place «Master data» and two «Detail data» bands on the page. Link all bands to the appropriated datasets.



You can also create «Master-Detail-Detail-Detail», «Master-Master», «Master-Detail-SubDetail-SubDetail» reports etc., except " Master-Detail-Master-Detail" reports. Such report must be multi-page – you must move master-detail sets to separate pages.

## Composite report

Composite reports are reports that includes several other reports. For creating composite reports, you should use TfrCompositeReport object . Place it on the form and fill its «Reports» property in run-time by references to other reports. Reports will be printed sequentially. If a reports page have «Print to previous page» option, it will be printed at remained space at the last page of the previous report.

## Report with BLOb fields

If you want to show BLOb data in an object, insert a reference to appropriate field to object's memo (or fill "DataField" property in the Objects Inspector). If you want to show BLOb data from non-DB source, you must do this in the OnBeforePrint event handler.

## Report without bands

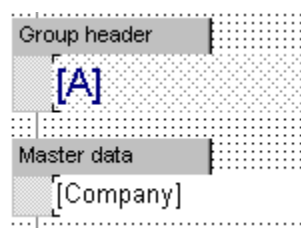
If you want to print a free-form report, containing data from only one record, you can not use bands. All the objects placed directly on the page, will be printed at their places.

## Report with groups

Groups are used for grouping data by some criteria. You can use any FR expression as a group condition (usually use expressions based on DB fields). When this expression changes, FR forms new group.

For creating this kind of report, place «Group header» and «Master data» bands on the page. Assign appropriate dataset to «Master data» band. In the editor of «Group header» band type the grouping expression. For example, to print customers list grouped by the first letter of customer name, type the following expression: Copy([CustomerName], 1, 1), where [CustomerName] is reference to appropriate DB field.

Report form



Prepared report

V	
Vashon Ventures	743
VIP Divers Club	32 M
W	
Waterspout SCUBA Center	7865

There are one limitation on using groups: you can not use groups in subreports.

**Note:** dataset used for «Master data» band should be sorted on grouping condition. You can do this by using queries with ORDER BY statement.

## Report with charts

Let's assume that we have the following table:

Action Club	\$1000
Action Diver Supply	\$12000
Adventure Undersea	\$5000

To build a report that displays charts it is necessary to have one data band i.e. master connected to appropriate TfrDataset. The band would have various rectangle objects placed on it. In our case, it will be two objects with "Name" and "Amount" fields placed in it. These objects, for instance, will be called Memo1 and Memo2. Place chart object in the band that will be shown after the entire master data have being shown – in our case, it can be report summary band.

In the chart object editor select the type of chart and options on the display tab, on the data tab enter the name of the memo to be used for the legends (Memo1 in our case), then enter the name of the memo which has the numerical data you wish to chart (Memo2).

If there is a large number of records you may wish to set the “top ten” group value to a small number. This will cut down the number of items shown in the chart. Size the chart rectangle to a reasonable size for displaying the chart. Preview the report and see your results. This is a brief summary, to display a chart you may have to use variables in some cases depending on the type of values being supplied in the underlying data.





# The designer

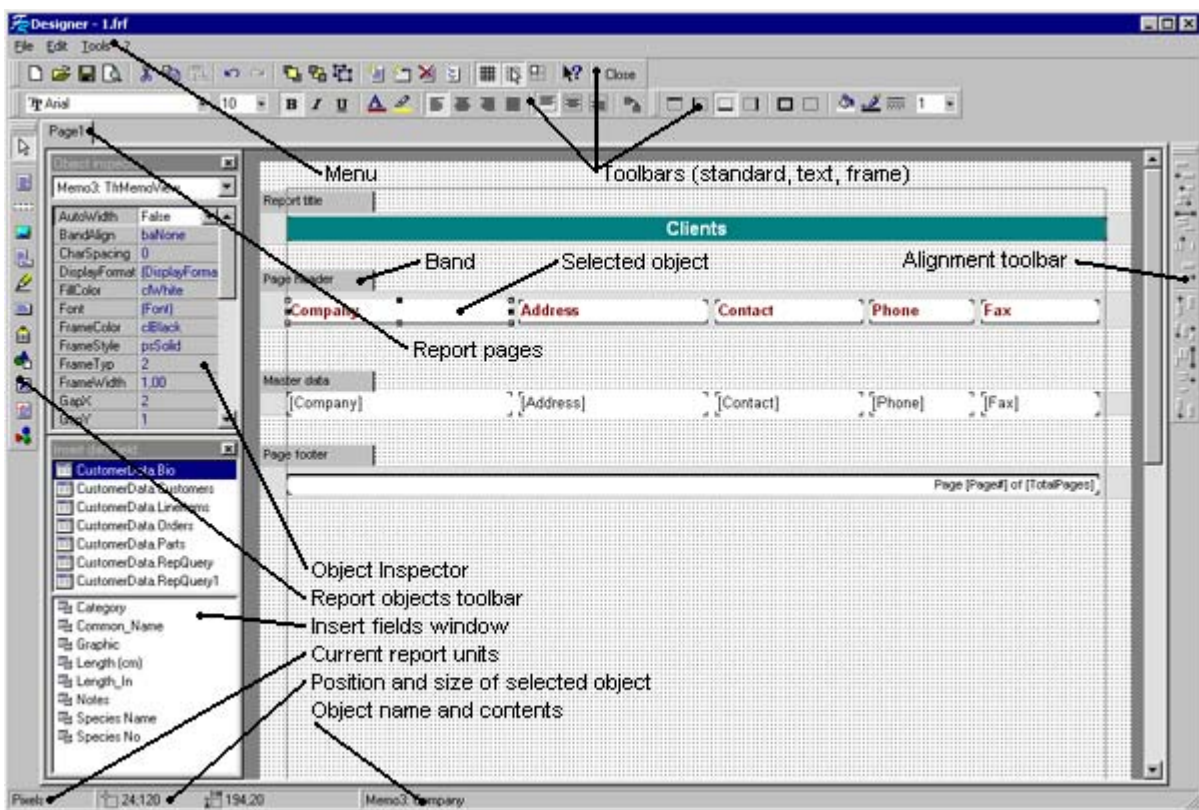
The designer

## The designer

FastReport is supplied with its own report designer, which can be accessed at design time by double clicking the TfrReport component. The designer allows quick and easy access to the report design and also for live previewing of the report while in the Delphi IDE.

The designer includes dockable panels (toolbars), which may be moved and altered to suit your particular needs. The position and visibility of all the panels are saved to the windows registry when the designer is closed and their positions and visibility are restored the next time you open the designer.

If you want the designer to be available at run-time, you need to include a TfrDesigner component on a form within your project (which is visible to the TfrReport component you are using). Alternatively, you can include the FR\_Design unit in the uses clause of your unit. The TfrDesigner component gives your end-user the capability to load, save, design and edit reports.



## Using the keyboard

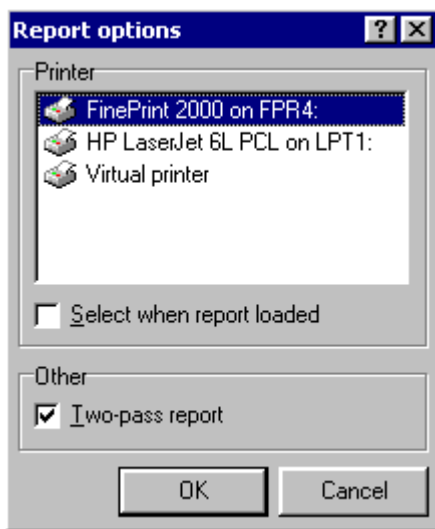
- Arrow keys – move to next object.
- Ctrl + Arrowkeys - moves selected object(s) in direction of arrow.
- Shift + Arrowkeys - increase or decrease dimension of selected object(s) in arrow direction.
- Enter - brings up editor of selected object.
- Del - delete selected object.
- Insert - show “Insert data field” toolbar.
- Ctrl + Enter - brings up memo editor of selected object.
- Ctrl + 1..9 - sets frame thickness of selected object.
- Ctrl + Z - undo last action.
- Ctrl + Y - redo cancelled action.
- Ctrl + G - toggles grid on/off.
- Ctrl + B, Ctrl + I, Ctrl + U - toggle bold/italic/underline font attributes.
- Ctrl + F - turn object’s framing off.
- Ctrl + D - turn object’s framing on.
- Ctrl + X - cut to clipboard.
- Ctrl + V - paste from clipboard.
- Ctrl + C - copy to clipboard.
- Ctrl + A - select all objects on Page.
- Ctrl + N - create new empty report.
- Ctrl + O - open report file.
- Ctrl + S - save report file.
- Ctrl + P - preview report.

## Using the mouse

- Left click - in Page window selects object; in visual component palette selects object to insert – after selection the LeftClick over the Page window inserts a new object.
- Right click – brings up the context menu for the selected object.
- Double click – brings up the default editor of the selected object. Double click over the free space on the Page brings up the Page-option dialog, where you may choose the Page options i.e. margins, size etc.
- Shift + left click – toggles object’s selection.
- Ctrl + left click – draws a selection rectangle. All objects that fit in this rectangle will be selected after you release the mouse key.
- To scale several selected objects, drag the red square on the right-bottom corner of the object’s group.

## Report options

To set Report Options choose “File|Report options...” from designer’s menu.



The list at the top of the dialog box lists all available printers on the system. If there is no printer installed on your system, you can select the “Virtual printer” option, which allows to use any paper dimension, but there is no possibility for a printout. There is only the possibility to work with the report in the designer mode, and also to view the preview of the report. You may also use the virtual printer option for designing a report for a printer, that is not installed on your PC (for instance, you want to design a report for A3 paper, but your printer allows only A4).

If “Select when report loaded” option is checked, the printer information is stored with the report and when this report is re-loaded, the stored printer will automatically be selected. If that printer is not found on the system, then the default windows printer will be selected.

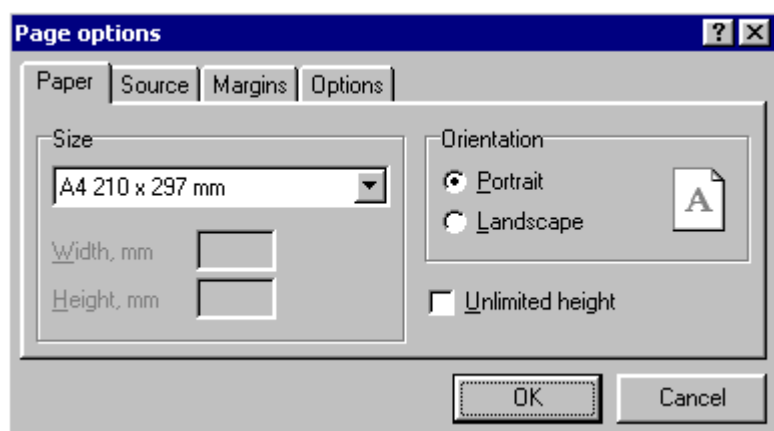
The “Two pass report” option must be checked if you need to use “total pages” function in the report, i.e. print “Page xx from yy”. If you’re using the TOTALPAGES function, but you have forgotten to turn on this option, you’ll get a zero instead of number of total pages in your report. Another benefit of making reports “two-pass” is to do some calculations during the first pass and show the results on the final pass. One of these calculations – show group totals in the group header – can be found in the demo reports.

After selecting any printer, the Page window in the report designer shows the available printable area for the Page size and printer selected.

## Page options

To set the page options for the current page of report, choose “File|Page options...” in designer’s menu, or double-click on an empty space on the page. The dialog has four tabs.

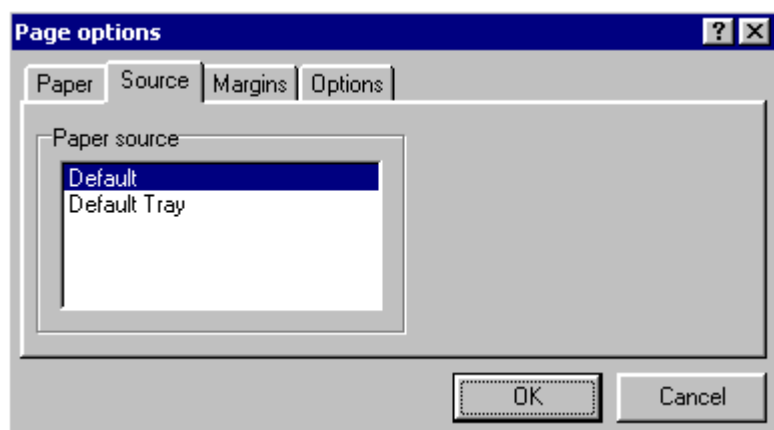
### The “Paper” tab



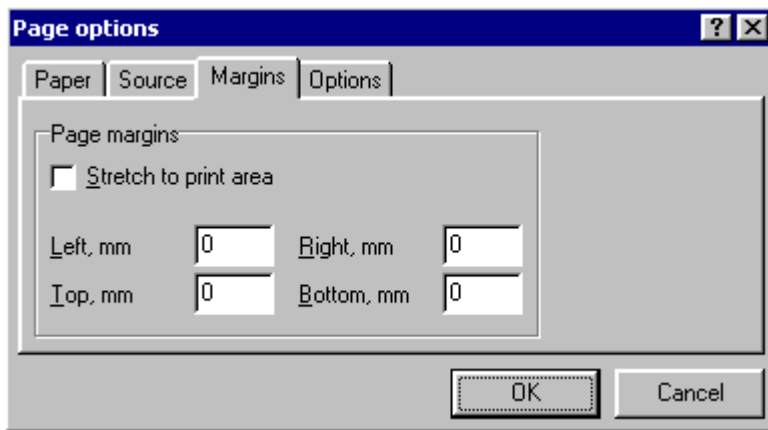
You can select the paper format from the drop-down list of paper formats supported by the current printer. If the current printer supports custom paper sizes and you select “custom” from the list, you will be able to enter the width and height of the custom size. Any other choice uses preset sizes. You may also choose the orientation for the Page and set "Unlimited height" option – it increases the height of a page (in case if you want more space to place a lot of bands on the page).

**Note** Not all printer drivers or printers support custom paper sizes, (for example, printer driver “HP LaserJet 6L” does not support page dimensions less than 76 \* 127 mm; printer driver “HP LaserJet 4L” does not support a custom size page at all).

### The “Paper source” tab



### The “Margins” tab

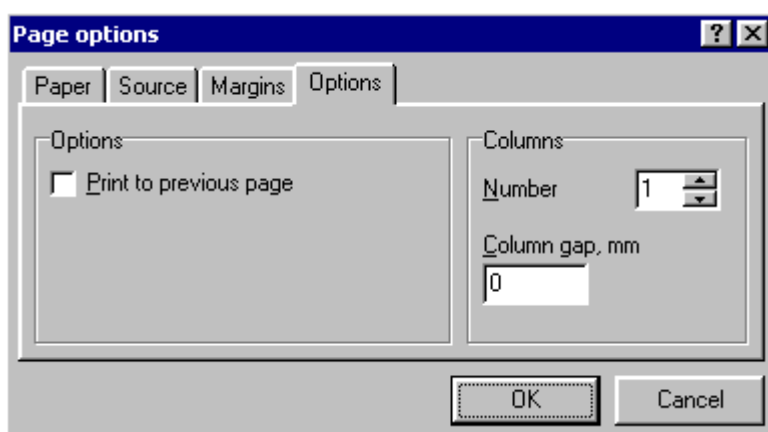


On the “Margins” page of the dialog you may select whether or not to use margins and the size of the margins. If the “Don’t use” option is selected, no printable area border will be shown in the designer’s Page window, and all page contents will be printed correctly on any printer. But object’s sizes will not be the same on different printers.

If you leave this option unchecked, and all margin settings are zero, the pageborder will reflect the maximum printable area for the selected printer. You may find some chopping when you switch between reports designed for one printer to another, particularly between ink-jet and dot matrix. Ink-jet printers usually have a smaller printable area than a dot matrix due to the way they feed the paper.

If margins are set to non-zero values, a border will be shown in the report designer’s page window (using a hash line style). If you use dot matrix printers, pay close attention to the printable area: some dot matrix printers will not print if the object to be printed overruns the printable area, others will produce pages with the over-run printed on them. This obviously produces weird looking reports. In this case, you should set margins manually.

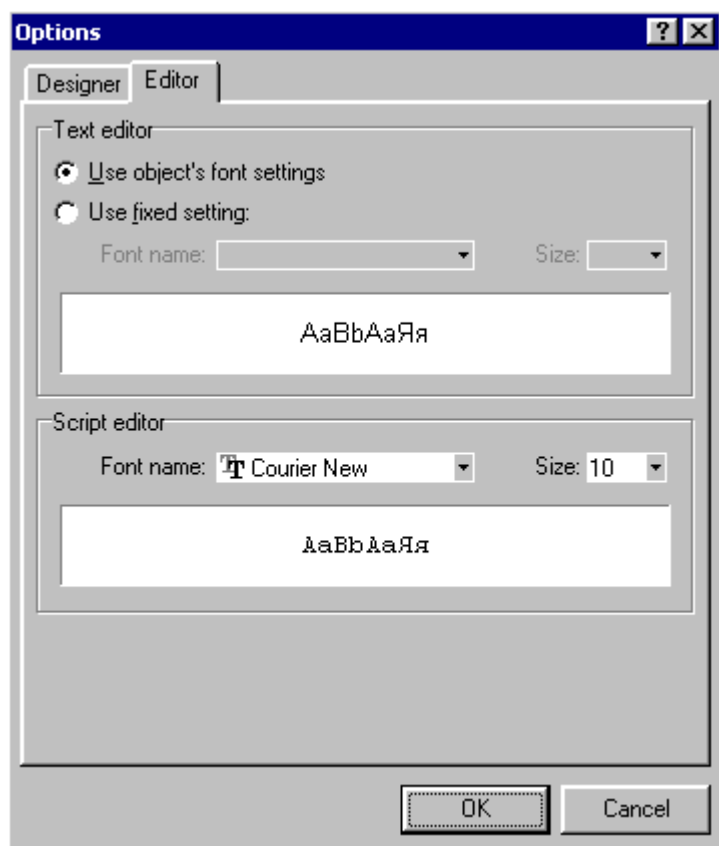
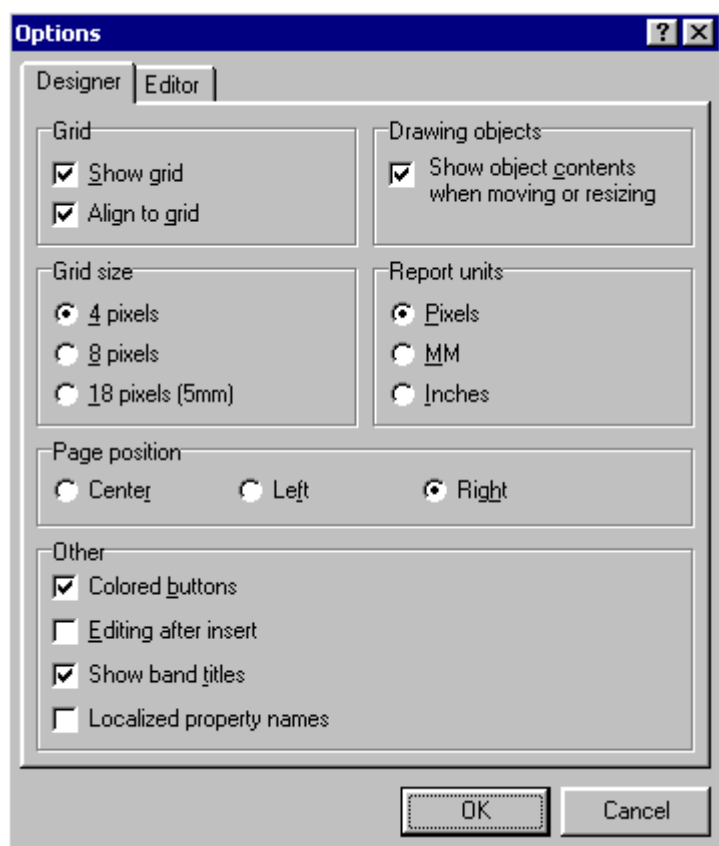
### The “Options” tab



On the “Options” page of the dialog, you may set many other options for the page. You may set the number of columns across the page width and the space between them. If “Print on previous page” flag set to true this will allow a new page to start printing on the unused space of the previous page.

## Designer options

To set the default options for the report designer choose “Tools|Options...” menu command.



Here you can set grid size, report measurement units: pixels, millimeters and inches. Grid size 18 pixels corresponds to 5mm.

You can also control how to paint objects when moving or resizing them: show wire-frames or full-repaint them.

The “page position” group allows you to choose the page position (this is needed by the Object Inspector).

If “Coloured Buttons” is turned off, then all Buttons are painted black and white.

The “Editing after insertion” option lays down the default action after inserting objects, if the default editor for the object appears after insertion or not. If inserting a large amount of empty rectangles turn this option off.

“Show band titles” allows you to turn band titles (tabs) off if you want more space on the page when designing.

"Localized property names" allows localization for Object Inspector.

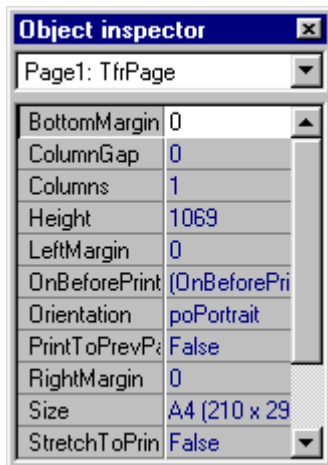
Snap to grid snaps objects to the nearest grid point when moving them.

On the “Editor” tab you can select the font of the editor window. You can select between fixed setting and object’s settings.



## The Object Inspector

The Object Inspector allows you to manipulate object's properties.

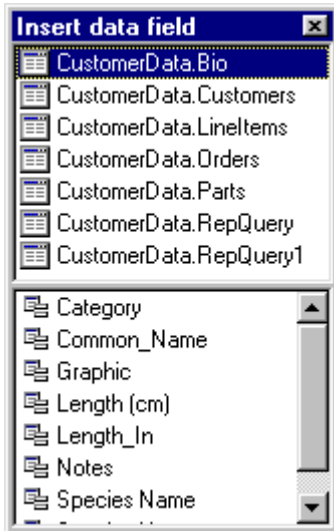


The Inspector works just like the Delphi-Object Inspector. Like the other toolbars, it may be shown or it may be hidden. To show the Object inspector, select the “Tools|ToolBars|Object Inspector” menu item. To shrink the Object Inspector, double-click on its title bar; double-click on the title bar while it is shrunken leads to re-expanding the Object Inspector.

## The “Insert data fields” window

You can quickly insert a DB field into the report using this dialog window. You can run it from the menu "Service|Toolbar|Insert a DB field" or by pressing “Insert” key.

The window contains two lists: table (query) list at the top of the window and fields list at the bottom. The field is inserted using the drag&drop method. You need to select a necessary field with mouse from the bottom list and holding down the mouse button, move it to the report page. When you release the mouse button, there should appear a new object “Text” with a link to the selected DB field.

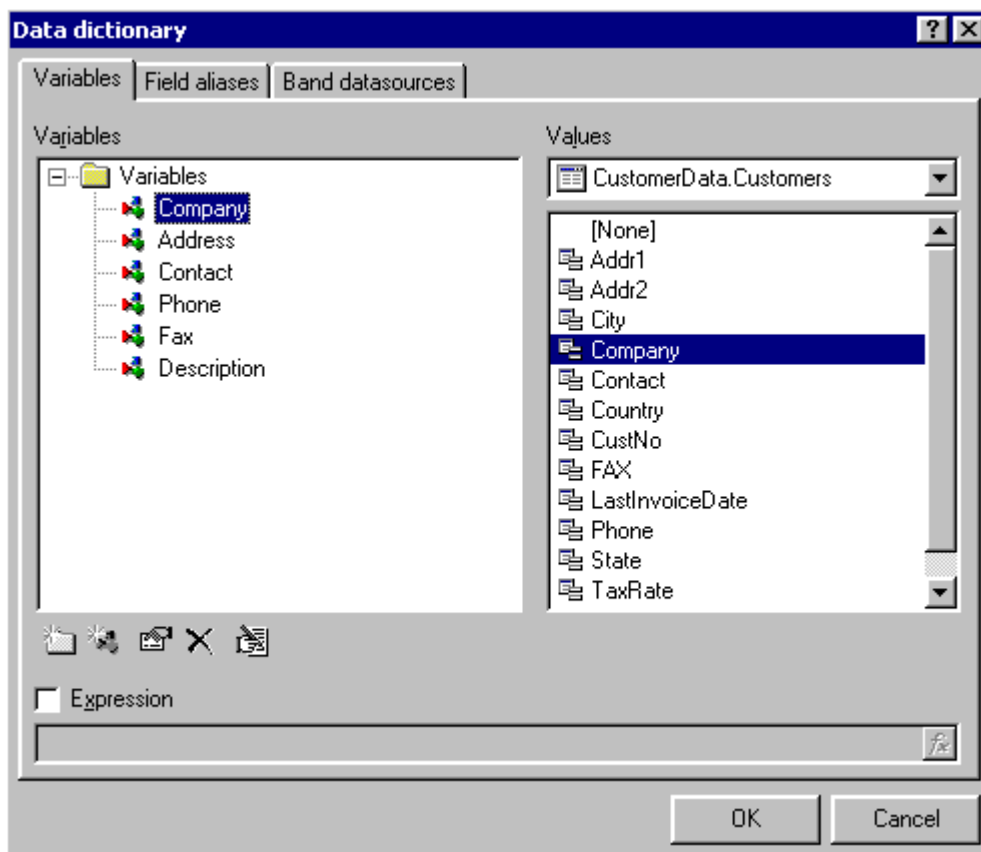


If the dialog is using often, it's useful to locate it under the object inspector or over it. In this case double-click the dialog window title and so, the window will be minimized and the inspector window will be maximized; Double-Clicking again will make a reverse, so it'll restore the standard widow state and the size of inspector window will be decreased.

## The Data dictionary

The "Data dictionary" window can be chose from the "File|Data dictionary..." menu. The window looks like a notebook with three bookmarks: "Variables", "Data from DB" and "Data sources for bands". The Data dictionary is saved in file with the report form, but there is a possibility to save it in separate file with .FRD extension and read data to the existing report when necessary. To do it, use the commands from the designer's menu "File|Open" and "File|Save As." and choose the file type "The Data dictionary FastReport" in the appeared standard dialog windows of opening and saving.

### "Variables" tab

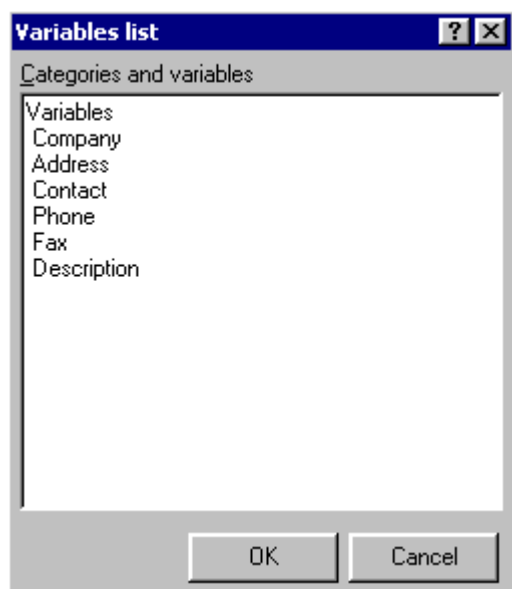


This tab is used to work with variables list.

The variables list is located in the left part of the window. As shown at the picture, the list structure contains 2 levels. It consists of categories, there maybe one or some variables in each category. The categories are needed just for visual variables ordering, but it is not inserted to the report.

When creating a new report the list is empty, and you can fill it by using the buttons under the list:


- adds a new category and - adds a new variable to the current category. These actions may be run using keyboard keys: "Insert" key adds a new variable, "Ctrl+Insert" adds a new category. For other icons: - edits the name of variable or category (keyboard analog – "Enter" key), deletes a variable or a category ("Delete" key). - calls the variables list's editor, where it's shown as a strings list. Here you can insert multiple variables from the clipboard, move variables from one category to other.



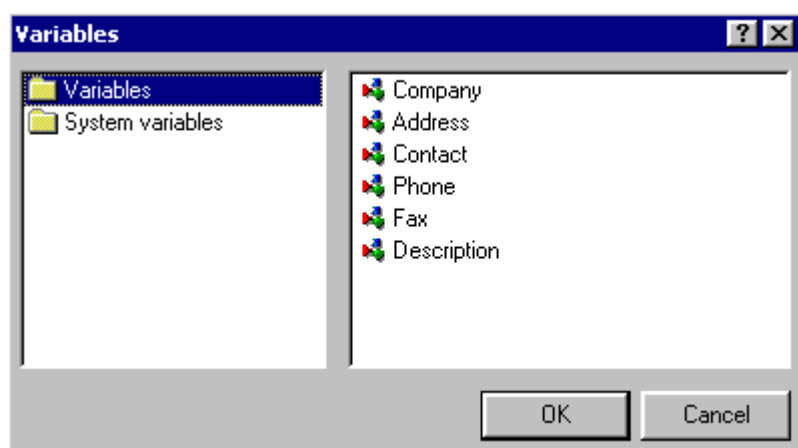
After the variables list is defined, you should set a value for each variable. To do it, you need to choose a variable in the list on the left and a value in the list on the right, using the mouse.

All data sources are shown in the list on the right (nonvisual dbaware-components, component-children TDataSet), are available at this moment, so the names of its fields are also available. In design-time all datasets, located in opened modules (units) will be available. In run-time all datasets, located in created forms or data modules (TDataModule) will be available. You can select a value "System variables" from the list box on the right, and then there will be a possibility to set one of the following values to the variable:

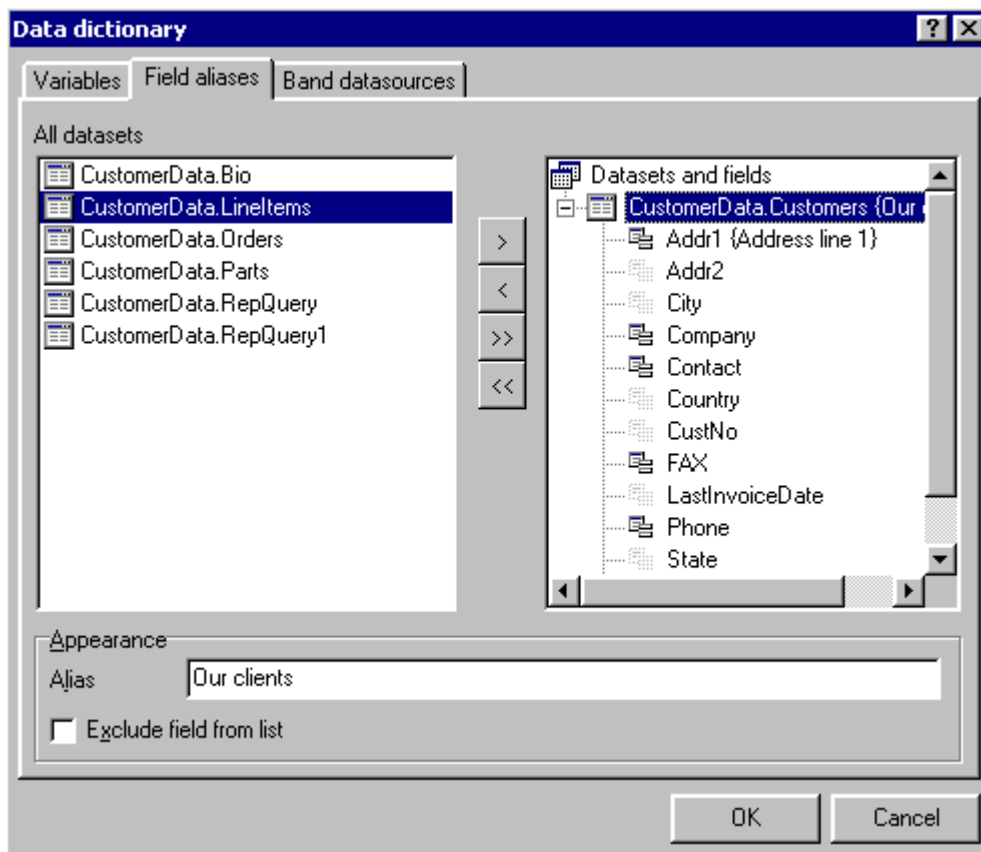
- page – number of the current page, it's equal the Page# function (see the description of internal functions);
- date – date of the beginning of creating a report, equal Date;
- time – time of the beginning of creating a report, equal Time;
- string, LineThrough#, column, CurrentLine#, Total pages – see the description of internal functions

An expression may be the value of the variable – check the "Choose a variable" checkbox and check "Expression" checkbox at the bottom of the window. So the text field for expression become active. For visual show of expression press this button .

The variables insertion window for highlighted example will be shown like this:



## "Field aliases" tab



On the left at this bookmark, there is a list of all available dataset – tables, requests, which are on all the forms of the project.

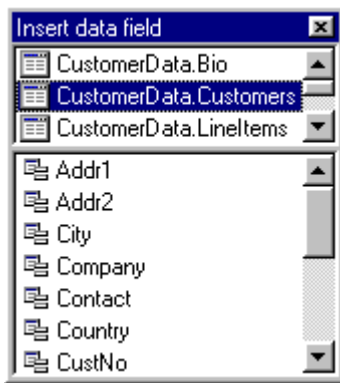
This dialog window is used to remove unnecessary data from the list and name it with more detailed names (pseudonyms). It's necessary because the report, as a rule, contains the data just from one-two requests or tables. FastReport, by default, offers to insert the fields from all the data sources, which it determines on all the project's forms. In large projects there are tens (or hundreds) tables and requests.

To use a dataset, you should move it from the left list to the right one. You can do it using drag&drop method, or double-click the necessary dataset, or using the buttons in the middle part of the window.

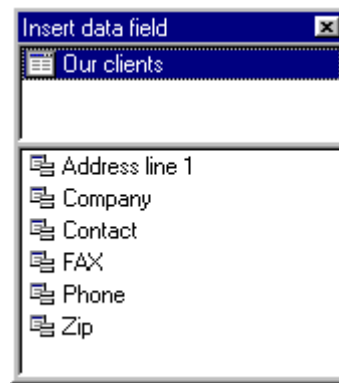
To remove the field from the list, choose a necessary dataset and its field and check the "Remove the field from the list" box. Checking the box again restores the including of the field to the list. The command is also available by pressing the "Space" key.

To set a pseudonym, choose a necessary элемент and enter a new name in the "Pseudonym" field. If a pseudonym is not required – clear all the field's data. This command is also available by pressing the "Enter" key.

As you see from the picture, creating pseudonyms for data sources and its fields make its use when building a report easier. Compare the dialog window for inserting fields to the report before applying pseudonyms and after it:

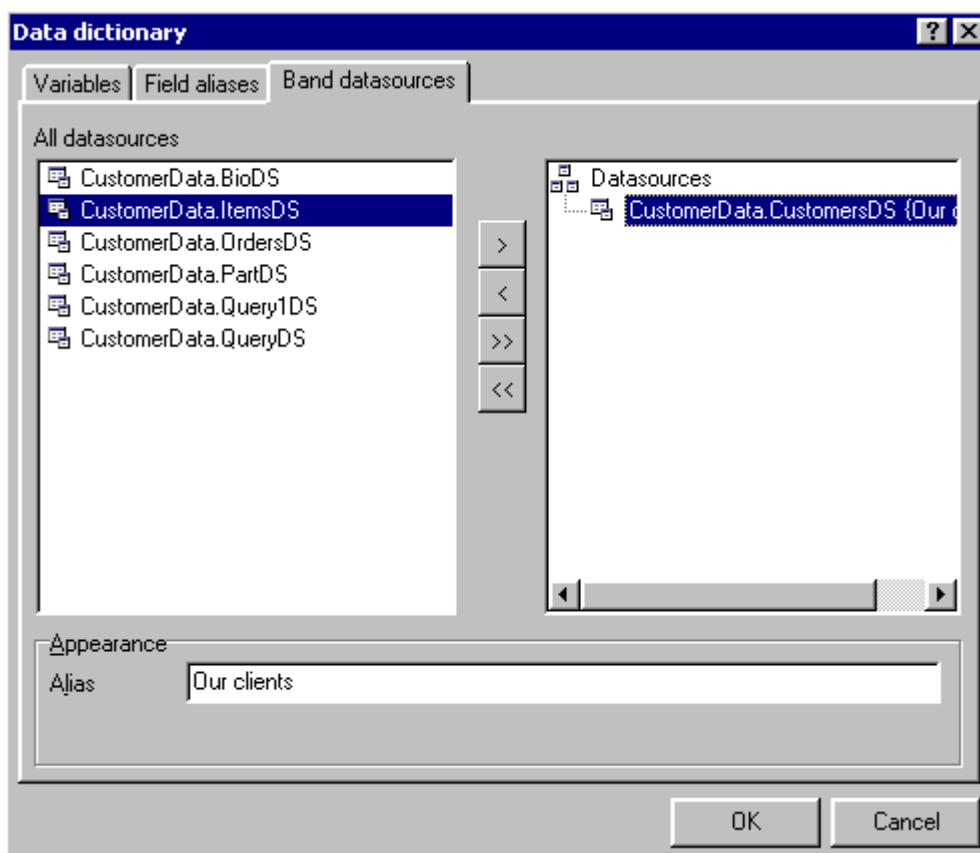


Without pseudonyms



Using pseudonyms


### "Band datasources" tab

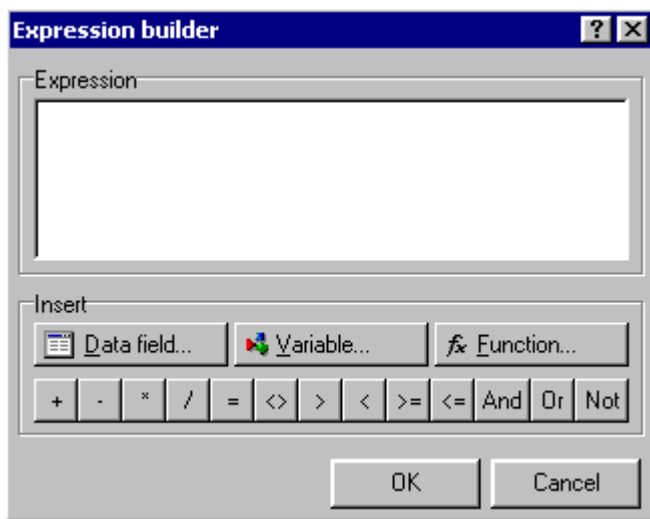


This tab shows a list of all available data sources for bands - components TfrDBDataSet, TfrUserDataSet, located on all the forms of the project. Like the previous dialog, you can use more detailed names (pseudonyms). The result is shown in the dialog with choice of data source for data-band:



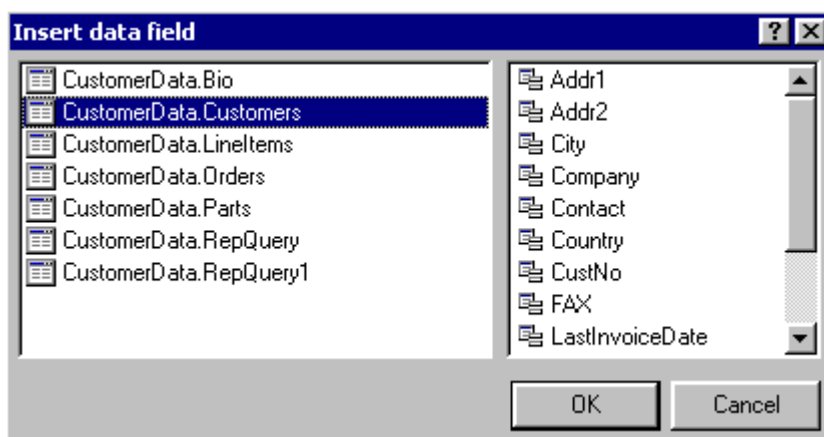
## The Expression builder

The expression builder window can be run from the text editor, by pressing  on the toolbar. You can also choose it from some dialog windows, where it's necessary to set an expression (ex., in editor of band group conditions, group header).



The window contains a field for entering an expression and buttons to call dialog windows for variables insertion, DB fields, functions, and buttons for quick arithmetical and logical operations signs insertion.

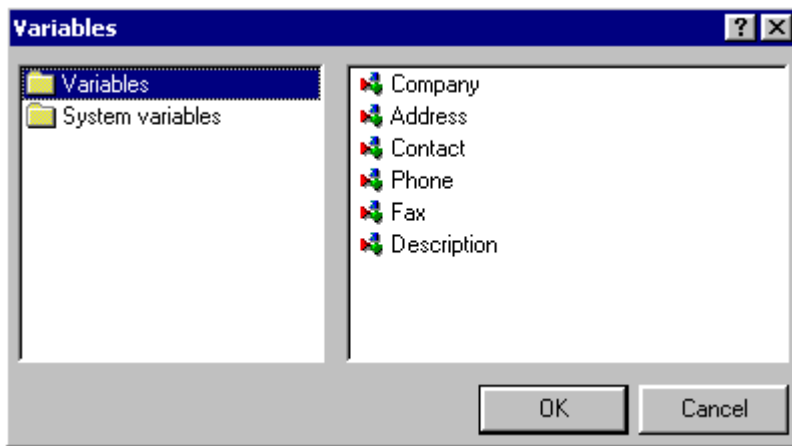
### "Insert data field" dialog



The dialog helps to choose a DB field to insert to the expression. On the left, there is a list of all available DB tables. On the right – the fields list in the selected table. To insert a field to the expression, choose a field and press "OK" button, or double-click the field.

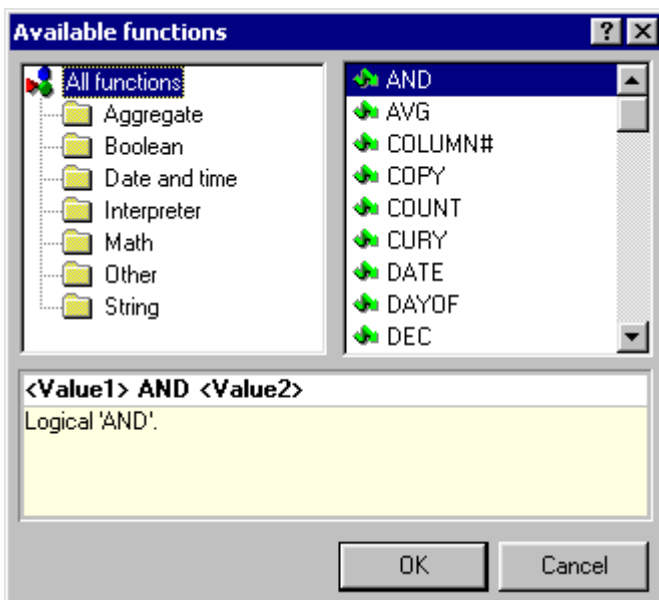
### "Insert variable" dialog





The dialog helps to choose a variable to insert to the expression. On the left, there is a category list. On the right – variables list in the selected category. To insert a variable to the expression, choose a variable and press “OK”, or double-click it.

### "Insert function" dialog



You can choose a function to insert to the expression in this dialog. While the function is selected, you can see a short description of the function and the list of its arguments in the bottom of the window. If the function has arguments, you’ll be asked to fill it after pressing “OK”:

**Function arguments** ? X

**AVG(<Expression> [,BandName [,1]])**

Calculates the average of <Expression> for [BandName] row given. If [1] parameter is used, calculates average for non-visible rows too.

Argument 1  *fx*

Argument 2  *fx*


















Argument 3  *fx*



OK Cancel

## Toolbars

### The "Standard" toolbar


















Icon	Name	Description
	New report	Create new empty report.
	Open report	Open existing report from file with FRF extention. Keyboard shortcut - Ctrl+O.
	Save report	Save the report to file with FRF extention. Keyboard shortcut - Ctrl+S.
	Preview	Run report and show it in the preview window. Keyboard shortcut - Ctrl+P.
	Cut	Cut selected objects to the internal clipboard. Keyboard shortcut – Ctrl+X.
	Copy	Copy selected objects to the internal keyboard. Keyboard shortcut - Ctrl+C.
	Paste	Paste objects from internal clipboard. Keyboard shortcut - Ctrl+V.
	Undo	Undo last operation(s). Number of undo levels – up to 100. Keyboard shortcut - Ctrl+Z.
	Redo	Redo last operation that was undone. Keyboard shortcut - Ctrl+Y.
	Bring to front	Bring to front selected objects.
	Send to back	Send to back selected objects.
	Select all	Select all objects on the page. Keyboard shortcut - Ctrl+A.
	New page	Create a new empty page.
	Delete page	Delete current page.
	Page options	Show page options dialog.
	Show grid	Show grid on the page. The size of the grid can be adjusted in the designer options dialog. Keyboard shortcut - Ctrl+G.
	Align to grid	When moving or resizing objects, its coordinates or sizes will changes according to grid size.

	Fit to grid	Change the position and size of selected objects so they fit to the grid cells.
	Help	Show the context help on selected element.
Close	Close	Close the designer window.

## The "Text" toolbar



Icon	Name	Description
	Font name	Drop-down list of all fonts installed in your system. Double-click on this control, and you will see a standard “Select font” dialog.
	Font size	Drop-down list of available font sizes for the selected font. If you want to enter a size manually, click on this control, enter appropriate size and press Enter.
	Bold	Toggle “bold” font attribute. Keyboard shortcut - Ctrl+B.
	Italic	Toggle “italic” font attribute. Keyboard shortcut - Ctrl+I.
	Underline	Toggle “underline” font attribute. Keyboard shortcut - Ctrl+U.
	Text color	Select text color from drop-down color palette.
	Conditional highlighting	Change conditional highlighting.
	Left align	Align text to the left of the object.
	Center	Align text to the center of the object’s width.
	Right align	Align text to the right of the object.
	Width align	Fit text to both sides, left and right.
	Top align	Align text to the top of the object.
	Middle align	Align text to the middle of object’s height.
	Bottom align	Align text to the bottom of the object.
	Text orientation	Change the text orientation (normal/90 degrees).

## The "Rectangle" toolbar








Icon	Name	Description
	Top line	Turn on/off the top frame line.
	Left line	Turn on/off the left frame line.
	Bottom line	Turn on/off the bottom frame line.
	Right line	Turn on/off the right frame line.
	All lines	Turn on all frame lines.
	No lines	Turn off all frame lines.
	Fill color	Select the object's fill color from the drop-down palette.
	Line color	Select the line color from the drop-down palette.
	Line style	Select the line style from the drop-down list.
	Line width	Select the line width from the drop-down list.

## The "Alignment " toolbar



Icon	Description
	Aligns the selected components to the left edge of the component first selected. (Not applicable for single components.)
	Moves the selected components horizontally until their centers are aligned with the component first selected. (Not applicable for single components.)
	Aligns the selected component(s) to the center of the window along a horizontal line.
	Horizontally aligns three or more selected components so that the middle components are equidistantly spaced between the outer components.
	Aligns the selected components to the right edge of the component first selected. (Not applicable for single components.)

	Aligns the selected components to the top edge of the component first selected. (Not applicable for single components.)
	Moves the selected components vertically until their centers are aligned with component first selected. (Not applicable for single components.)
	Aligns the selected component(s) to the center of the form along a vertical line.
	Vertically aligns three or more selected components so that the middle components are equidistantly spaced between the outer components.
	Aligns the selected components to the bottom edge of the component first selected. (Not applicable for single components.)

# End-user features

Introduction

The dialogue forms

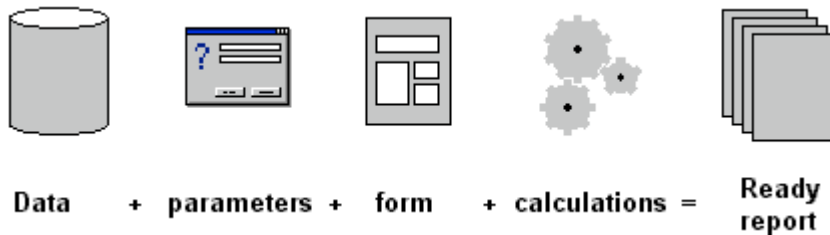
Data access components

The TfrDataStorage component

Built-in language

## Introduction

As were written in the “Introduction” chapter, the main report building stages are:



In the reports, which are built-in the project, all this stages can be made with Delphi services. So, data access is realized with using dbaware-components: TTable, Tquery and so like (as a rule, there is a common database for all the project and it's just possible to access to its tables); for parameters request you can use dialog forms, created for the project making process; processing stage is realized with using event handlers.

In spite of efficiency, this method (applying in the most report generators), is not universal. A little change in any of highlighted stages requires re-compiling all the project. But, if there are an information systems with a big amount of reports, this will follow the cluttering of the project with the dialog forms. So, this stage is not efficient in the projects, where create new kinds of reports and modify old reports.

History of the report generators (such as QuickReport, ReportBuilder, FastReport) goes step-by-step to become an independent tool from the project. So, at first the reports are modified in run-time. After, the end-user can determine data sources himself for report building. ReportBuilder version 4.0 or higher, helps you to make a data processing for the time of report building with using Pascal-relative language. FastReport has components for dialog form building.


And so, FastReport offers you to:

- Create new and modify existing reports;
- Create data sources, which a report will use to build;
- Create a dialog forms for report parameters request by user;
- Process report data and manage the dialog windows' work, using built-in Pascal-relative language.

As you see, it helps to create project-independent reports. Components' ideology is based on the Delphi's ideology.



## The dialogue forms

The development of dialogue forms uses the same designer as the report page forms. This button  on the designer toolbar will create a new dialogue form. The Objects toolbar will automatically change to reflect the type of form that is currently active: either a report page form or a dialogue form.

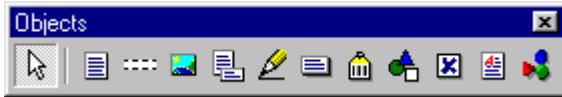


Fig. The 'Objects' toolbar when a 'Report Page' form is active.

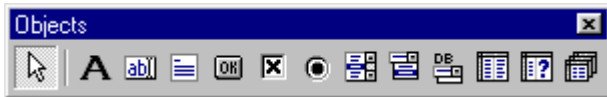
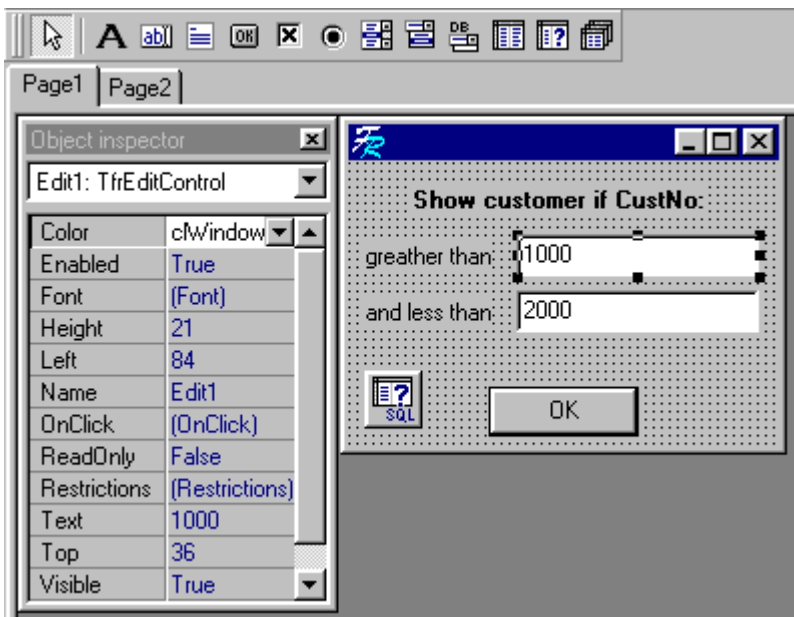










Fig. The 'Objects' toolbar when a 'Dialogue' form is active.

When a dialogue form becomes active, the controls of the designer will change from the report page style controls to the dialogue style controls. These controls can then be placed on the form to create a dialogue box:



Note that the process of creating a report dialogue form is very similar to how a standard dialog form is created in the Delphi IDE. Several dialogue forms can be created in each report. These dialogue forms will be executed in the order of their creation until all of the dialogue forms are executed and accepted by pressing the OK button. Then the main report will be prepared and executed.

## Dialogue Form Controls

Icon	Name	Description
	Label	Static text control.
	Edit	Single-line edit control.
	Memo	Multi-line edit control.
	Button	Single command button control.
	CheckBox	Boolean decision button control.
	RadioButton	Set of mutually exclusive choices button control.
	ListBox	Fixed size, scrollable list control.
	ComboBox	Drop down listbox control associated with an edit control.

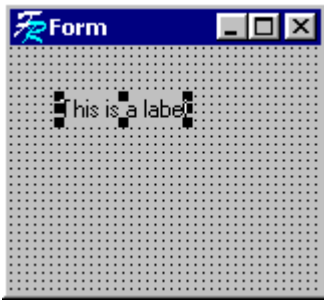
It is important to note that all dialogue controls in FastReport descend from the TfrStdControl class and have a the following common set of properties and methods:

Property	Default	Description
Color	clBtnFace	Background color of the object.
Enabled	True	Enable or disable the object.
Font	-	Specifies the font associated with the text of the object.
Height	-	Specifies the vertical size of the object.
Left	-	Specifies the coordinate of the left edge of the object.
Name	-	Specifies the name of the object as will be referenced in code.
OnClick	-	Occurs when the user clicks on the object – similar to the Delphi OnClick event. Specific code instructions can be placed here using FastReport Pascal.
Restrictions	-	Set of flags associated with the control to restrict the user from modifying the object (moving, deleting, editing, etc.).
Top	-	Specifies the coordinate of the top edge of the object.
Visible	True	Determines whether the object will appear on the screen.
Width	-	Specifies the horizontal size of the object.

Next we will take an in-depth look at each control.

### Label

This control is used to display static text, which is text that will never be changed by the user. Its normal purpose is to display explanatory text about other controls or large amounts of information to the user.



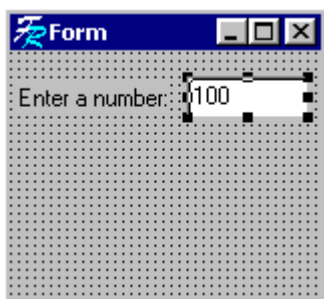
The Label control has the following properties (along with the above common properties):

Property	Default	Description
Alignment	taLeftJustify	Controls the horizontal placement of the text within the label
AutoSize	True	Determines whether the size of the label automatically resizes to accommodate the text.
Caption	-	Specifies the text string that the label will display.
WordWrap	False	Specifies whether the label text will wrap to a new line when it is too long for the width of the label. The AutoSize property must be set to 'False' when WordWrap is 'True'.



### Edit

This control is used to display and modify a single-line text string that is input from the keyboard.



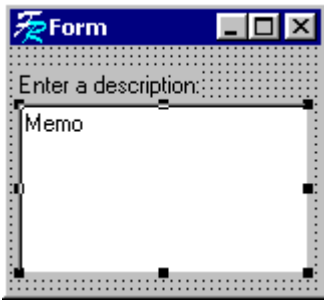
The control has the following properties:

Property	Default	Description
ReadOnly	False	Determines whether the user can change the text of the edit control.
Text	-	Contains the display text represented as a string, which will be shown in the control by default.



### Memo

This control is used to display and modify a multi-line text string that is input from the keyboard.



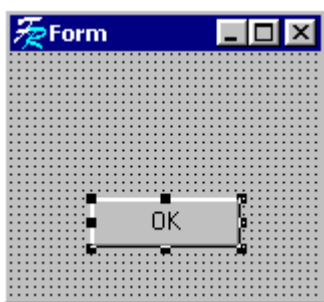
The control has the following properties:

Property	Default	Description
Lines	-	Contains the individual lines of display text represented as strings. These lines are shown in the control by default.
ReadOnly	False	Determines whether the user can change the text of the memo control.



### Button

This control is used to represent a command push button control.



The control has the following properties:

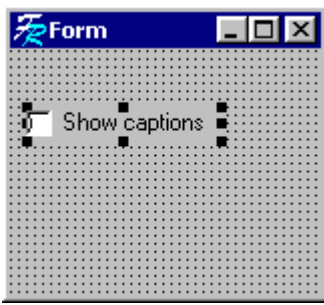
Property	Default	Description
Caption	-	Contains the text that is displayed within the button.
ModalResult	mrNone	Determines whether and how the button closes its (modal) parent dialogue form and what result will be returned.

Setting the ModalResult property is an easy way to make clicking the button close a modal dialogue form. When a button is clicked, the ModalResult property of its parent form is set to the same value as the button's ModalResult property. Usually a “OK” button will have a ModalResult = mrOk, and a “Cancel” button will be ModalResult = mrCancel.



### CheckBox

This control is used to represent a Boolean style option button. The user can check the box to select the option, or uncheck it to deselect the option. Examples would be Yes/No, On/Off, True/False, etc.



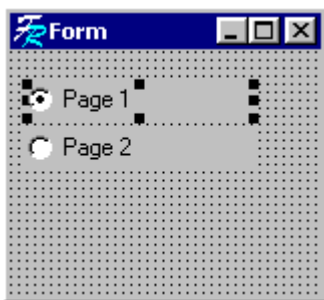
The control has the following properties:

Property	Default	Description
Alignment	taRightJustify	Controls the position of the check box's caption.
Caption	-	Contains the text that is displayed with the checkbox.
Checked	False	Indicates whether the check box is selected.



### RadioButton

This control is used to represent a set of mutually exclusive choices as buttons. That is, only one button in a set of buttons can be selected at any one time. A click on a button in the set of buttons will select that button and deselect all the other buttons in the set.



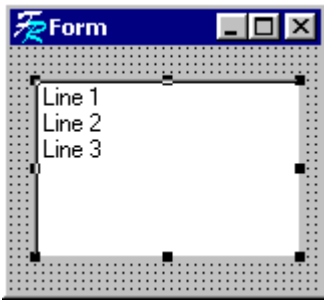
The control has the following properties:

Property	Default	Description
Alignment	taRightJustify	Controls the position of the radio button's caption.
Caption	-	Contains the text that is displayed with the radio button
Checked	False	Indicates whether the radio button is selected.



### ListBox

This control is used to display a scrollable list of items that the user can select from.



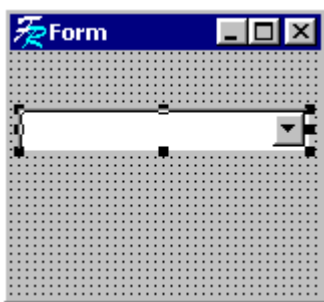
The control has the following properties:

Property	Default	Description
Items	-	Contains the individual items of display text represented as strings. These items are shown in the control by default.



## ComboBox

This control is used to display a drop-down, scrollable list of items that the user can select from without taking up the space that is required for a listbox control. Also with the associated edit control; it is possible to allow the user to enter items that are not present in the drop-down list.



The control has the following properties:

Property	Default	Description
Items	-	Contains the individual items of display text represented as strings displayed in the drop-down list region of the combo box.
Style	csDropDown	Determines the display style of the combo box.
Text	-	Contains the selected text in the edit region of the control.

These are three ComboBox Style property values:

- csDropDown - Creates a drop-down list with an editable edit control. The user can manually enter a text item that is not contained within the list. All items are strings of the same height;
- csDropDownList - Creates a drop-down list with a non-editable edit box. The user cannot manually enter a text item and the only items available to display in the edit control will be the items already contained within the list. All items are strings of the same height.
- csLookup - The strings in the list are represented in two parts: String1, String2. The Items property, as displayed in the drop-down list, will be shown with the first part of the string: String1. When one of the String1 values is chosen from the list, this value contained in String2 will be displayed to the edit control. For example, populating the control's Item property with strings like;

“January, 1”, “February, 2”, etc; could represent the months of the year. The selected value, String 2, that will shown in the edit control is available through the Text property.

The dialogue form also has a set of properties. These properties can be shown in the object inspector by clicking the mouse on an empty place on the form that is not occupied by controls.

Property	Default	Description
BorderStyle	bsDialog	The type of the dialogue form, fixed size or sizeable window.
Caption	-	The text display in the title bar of the window.
Color	clBtnFace	Specifies the background color of the window.
Height	-	Specifies the vertical size of the window.
Left	-	Specifies the coordinate of the left edge of the window.
OnActivate	-	Occurs after initialization of all elements of the window.
Position	poScreenCenter	Represents the size and placement of the window.
Top	-	Specifies the Y coordinate of the top edge of the window.
Type	ptDialog	Specifies the type of form, a dialogue form or report page form.
Width	-	Specifies the horizontal size of the window.

The OnActivate property closely corresponds with the same event in other Delphi forms. It is executed after all of the controls that are on the dialogue form are initialized and just before the form is ready to be displayed on the screen. Use the OnActivate property for initializing values in the form’s controls. For example, the Items property string list of a ListBox control can be populated with values at this time.

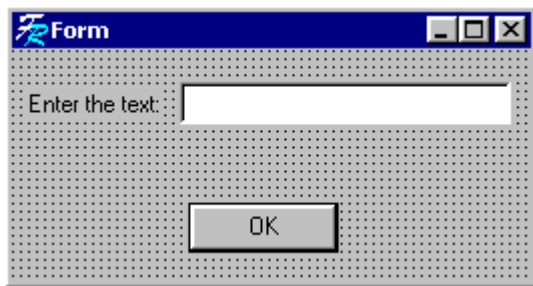
## Passing the information to the Report

It is frequently required to transfer values, entered in controls, to the report. There are two ways of doing this

1. Use of variables
2. Direct Calls

### Use of variables

Variables can be used to transfer the information from a control to the report. For example, when it is necessary to display text from an Edit control in header of the report.



This can be done in the OnClick event handler of the OK button:

```
begin
  TitleText := Edit1.Text;
end
```

In the header of the report, it is necessary to put the "Text" object with contents [TitleText] as shown below:



### Direct call

In many cases, it is easier to use direct call to control element for passing the entered values in the report. In this case, intermediate variables are not required:

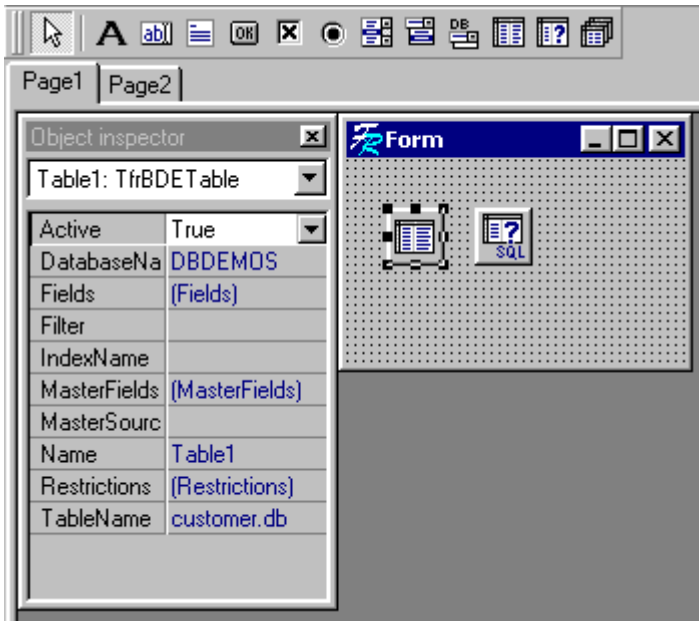





## Data access components

Majority of reports, as a rule, access data from the database (DB). For accessing such data, Delphi provides effective mechanisms, which are used in FastReport. These Data Access Components are TTable and TQuery, which can be used as the source of data for the report. In general, it is possible to use any component based on TDataSet for this purpose.

Except for a data access, which are created during design-time, FastReport allows the creation of new components at run-time. In FastReport, the process of creating components for data access is similar to that used when designing a Database application in Delphi environment. A required data access component is placed on the form, and its properties are customized in the Object Inspector.







## Description of FastReport DB-aware components

We shall discuss the use of components for data access with the help of BDE. These components are connected to the component TfrBDEComponents  from the palette FastReport, in the project and these components are TfrBDELookupComboBox, TfrBDETable, TfrBDEQuery, TfrBDEDataBase.



The purpose of these components is similar to the Database components TDBLookupComboBox, TTable, TQuery and TDataBase

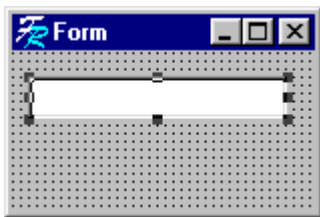
Icon	Name	Description
	TfrBDELookupComboBox	Drop-down list of lookup items for filling in fields that require data from another dataset.
	TfrBDETable	To access data from a Table
	TfrBDEQuery	To access data from a SQL Query.
	TfrBDEDataBase	To connect with a Database

Let's consider each component.



## TfrBDELookupComboBox

TfrBDELookupComboBox is used to provide the user with a convenient drop-down list of lookup items for filling in fields that require data from another dataset.



The element has the following properties:

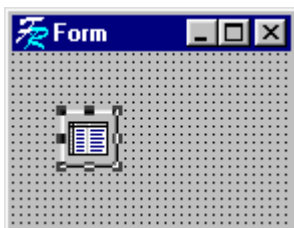
Property	Default value	Description
KeyField	-	Field - identifier of chosen value.
ListField	-	Field whose values are displayed in the lookup control
ListSource	-	Data source for the data displayed in the lookup control
Text	-	Currently selected lookup value

For connecting the lookup ComboBox with another dataset it is necessary to fill in values of three properties: KeyField, ListField and ListSource. Selected value can be accessed through Text property.



## TfrBDETable


This component is used to access the data from a Table.



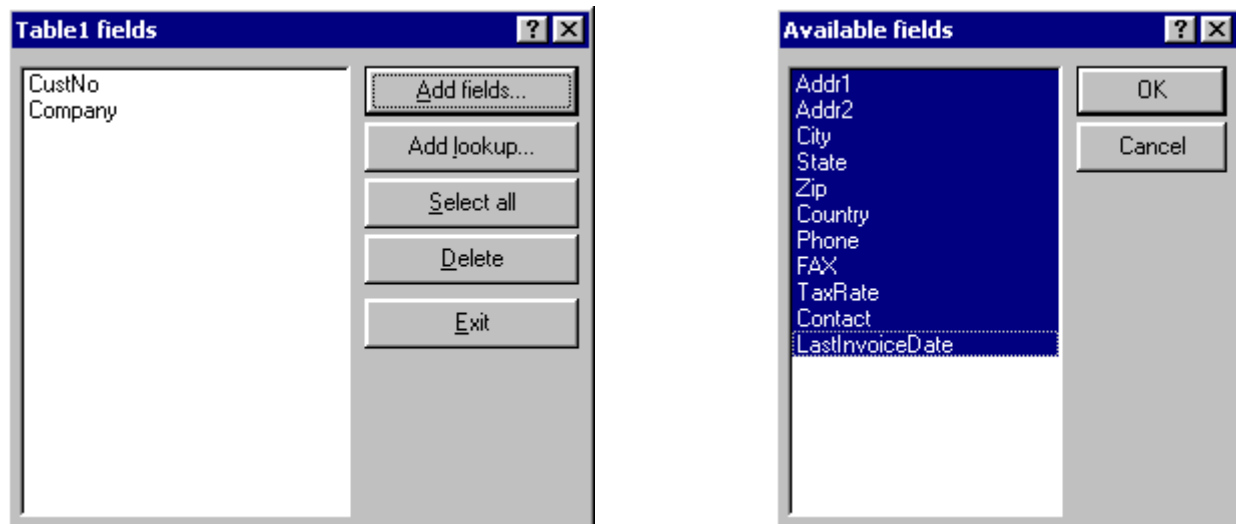
The component has the following properties:

Property	Default value	Description
Active	False	Determines, whether the table is active.
DatabaseName	-	alias or a pseudonym for the database.
Fields	-	The list of accessible fields.
Filter	-	Only the records matching this condition will be active, enabled or visible.
IndexName	-	Name of a secondary index.
MasterFields	-	The fields connected to a master-data set.
MasterSource	-	Master Dataset.
TableName	-	Name of Table .

The above properties are similar to properties of Delphi TTable component. For connecting the component to Table, it is enough to fill in the properties DatabaseName and TableName. Table is opened by setting the Active property of the above TfrBDETable to True.

Properties Fields and MasterFields are adjusted with the help of editors. For this purpose, it is necessary to press the button  in the Object Inspector.

The editor of property Fields allows us to choose the fields which will be accessible during the table look-up.



To add fields to the list, click on the "Add fields" button. A window will appear which will show all the fields, from there select the fields and click "OK" button. For selecting multiple fields or group of fields, click the mouse on the first field of group, then while keeping the "Shift" key pressed select the last required field. For deleting any of the selected fields from a list, choose the field to be deleted and press the "Delete" button.

To add a lookup-field, use the "Add lookup" button.

Following example demonstrates how to set up the lookup-fields.

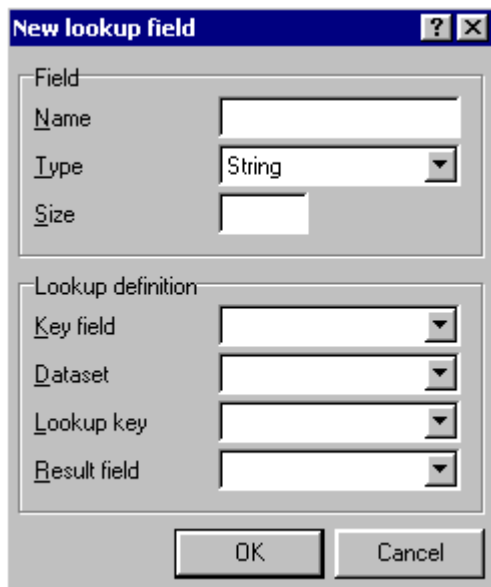
Let's assume that there are two database tables: the table "Orders" with fields N, Date, ClientID and Amount and a table "Clients" with fields ID, Name and Address. Table "Orders" contains the information on the orders (number, order date, identifier of the client who has made the order and sum of the order). Table "Clients" contains the information on clients (identifier of the client, full name, address).

To create an elementary report on the table "Orders" like

Order Number - Date - Name of the client - Sum,

it is required to link both tables on fields ClientID - > ID. This is done by creating a lookup-field, which one is added to the table "Orders" and represents the link on a field Name of the table "Clients".

Dialog box for creation of a lookup-field is accessible from the field editor.



The "New lookup field" dialog box is shown. It has a title bar with a question mark and a close button. The dialog is divided into two main sections. The top section, labeled "Field", contains three fields: "Name" (a text box), "Type" (a dropdown menu currently showing "String"), and "Size" (a text box). The bottom section, labeled "Lookup definition", contains four dropdown menus: "Key field", "Dataset", "Lookup key", and "Result field". At the bottom of the dialog are "OK" and "Cancel" buttons.

For creating a lookup-field it is necessary to set its Field Name and Field Type, and also the size (in case Field Type is of the type String). Further, it is necessary to fill in the following fields:

- Primary key field - field in the source dataset, which acts as the link on a field from a lookup-dataset. In our example it will be the ClientID field.
- Datasource - lookup-data set.
- Lookup key – the key field in the lookup-dataset. In our example, it is the ID field.
- It is necessary to substitute a resultant field - field of a lookup-set, which one in an source data set. In our example it is a field Name.

After that, in the table "Orders", there is a dummy field with a given name, which contains ClientID field. You can access to this field, as any other common field, but it is ReadOnly.

In the "MasterFields" property editor, you can link master and detail datasets visually, just like in Delphi IDE.



The "Table1 -> Table2" property editor is shown. It has a title bar with a question mark and a close button. The editor is divided into three main sections. The top left section, labeled "Detail fields", contains a list box with "ItemNo" selected. To its right is an "Add" button. The top right section, labeled "Master fields", contains a list box with "CustNo", "SaleDate", "ShipDate", "EmpNo", "ShipToContact", and "ShipToAddr1". Below these two sections is a "Joined fields" section, which contains a text box with "OrderNo -> OrderNo" and a "Clear" button. At the bottom of the editor are "OK" and "Cancel" buttons.

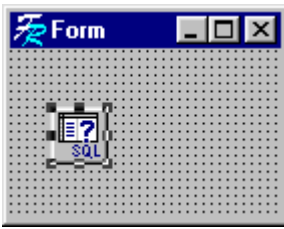
When the data sets are connected with each by Master-Detail, at moving on master data set the detail data set is filtering. So it consist only the records that have relation to master data set. For link fields of data sets select fields from list at left side (detail dataset), then select field from list at right side (master dataset), and click "Add" button. Thus the link fields moved to the lower list. To clear lower list push "Clear" button. The linked fields should have the identical type and to be

primary key.



### TfrBDEQuery

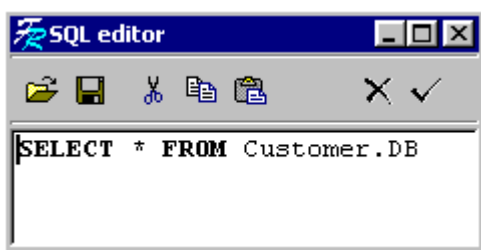
This component is used for performing of SQL searches on the Database.



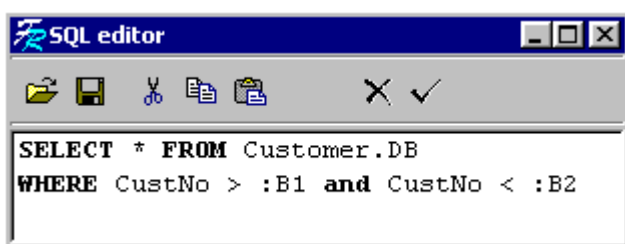
The component has the following properties:

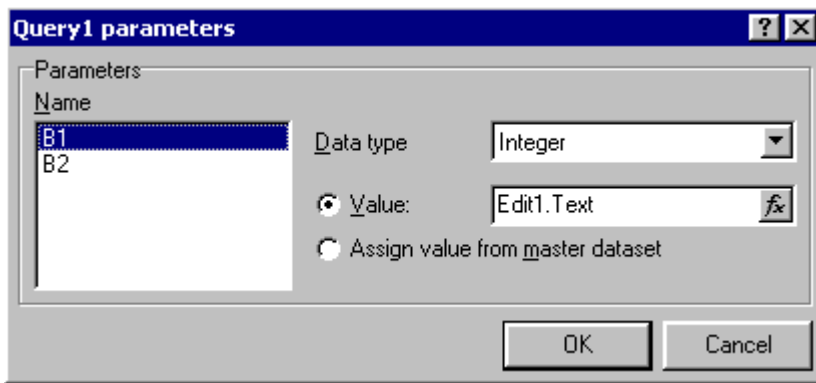
Property	Default value	Description
Active	False	Determines, whether the search is active.
DatabaseName	-	alias or a pseudonym for the database.
DataSource	-	Master - set.
Fields	-	The list of accessible fields.
Filter	-	Only the records matching this condition will be active, enabled or visible.
Params	-	The list of parameters in the SQL statement
SQL	-	SQL statement

Properties Active, DatabaseName, Fields and Filter are similar to the properties of component TfrBDETable as described above. The property SQL has an editor for giving the SQL Text.



Property Params also has an editor which is accessible when the SQL text contains parameters.



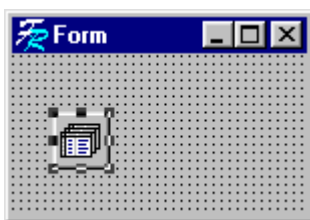


In this dialog box, it is possible to assign a type to each parameter, and to configure the source of the parameter value i.e either from a master-dataset or by assigning it a particular value. In case the parameter value is taken from a master-dataset, the dataset should contain the field with a name which should be same as the name of the parameter. Besides this field should be in the list of accessible fields (see fields editor). Thus it is optional to indicate the type of parameter.



### TfrBDEDataBase

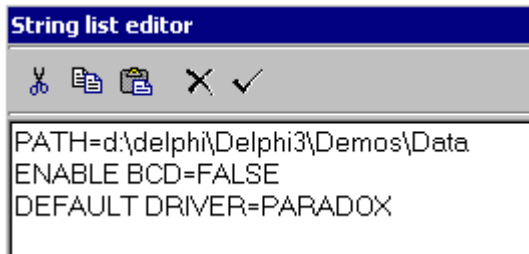
This component connects to a database. It is similar to the TDataBase component in Delphi.



Property	Default value	Description
AliasName	-	alias or a pseudonym, which will be used in the project, for the database
Connected	False	If True, makes the connection active
DatabaseName	-	Name which will be added in the list of alias
DriverName	-	Name of the driver providing connection to the Database.
LoginPrompt	True	Determines whether to prompt the user for Password while connecting to the Database.
Params	-	Connection Parameters

This component allows us to connect to the database (usually it is used to connect to the server control system of databases).

As well as in Delphi, you must set either AliasName or DriverName property. To set connection parameters, select “Params” property in Object Inspector and run property editor.



When "LoginPrompt" property is true, database login dialog will appear, when you connect to the SQL server. If you provide user name and password in database parameters, as shown below, you can make this property false.

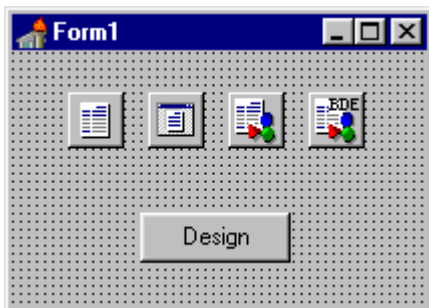
```
SERVER NAME=Path_to_gdb_file  
USER NAME=SYSDBA  
PASSWORD=masterkey
```



## Building Reports

Let's have a look at how simple reports with data access components are built. We will use the DBDEMOS demonstration tables from Delphi as the data source.

First we will make a new project for our experiments. To do that create a new project containing a form and put TfrReport, TfrDesigner, TfrDialogControls and TfrBDEComponents components on the form.



For the “Design” button define the following event handler:


```
procedure TForm1.Button1Click(Sender: TObject);
begin
    frReport1.DesignReport;
end;
```

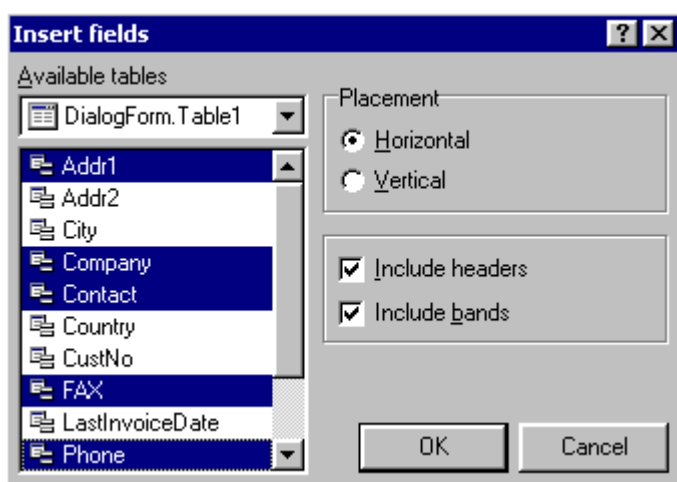
Then compile and run the project. That's all you need to make an end-user report designer.

When you click the “Design” button the designer with an empty report is loaded. Let's have a look at how simple reports are made in this environment.

## SIMPLE “TABULAR” TYPE REPORT

This report will show data from one table in a DB. To create the report follow these steps:

- Add a dialog form to the report. This form will be used to hold the data access component.
- Place a TfrBDETable component onto the form and change its settings as described below: DatabaseName = 'DBDEMOS', TableName = 'Customer.db'.
- Switch to the report form and click “Insert fields into the report” button  in the Wizard toolbar in the designer. In the open dialog box choose the required fields and click the OK button.



Now the report form will look something like this:


To view the report click the Preview button  on the toolbar.

## Report with parameters

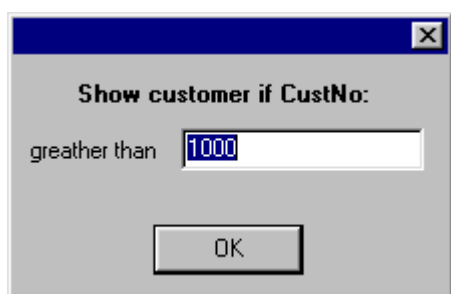
Let's look at more complex report building where the user will have to enter parameters in the dialog box before building the report. Follow these steps:

- Add a dialog form to the report.
- Place Query, Label, Edit and Button components onto the dialog form.


- Change the Query component's settings as shown below: DatabaseName = 'DBDEMOS', SQL = 'Select \* from Customer.db where CustNo > :B1'.
- Open the Params property editor of the Query component and change its parameter's properties.

- Switch to the report form and click the "Insert fields into the report" button  in the Wizard toolbar of the designer. In the open dialog box choose the required fields and click OK.

When the report is built you will be prompted for a customer number by the dialog box. Once you have entered the number and pressed OK the report will be built. All customers with numbers exceeding the value you entered will be printed.



## The TfrDataStorage component

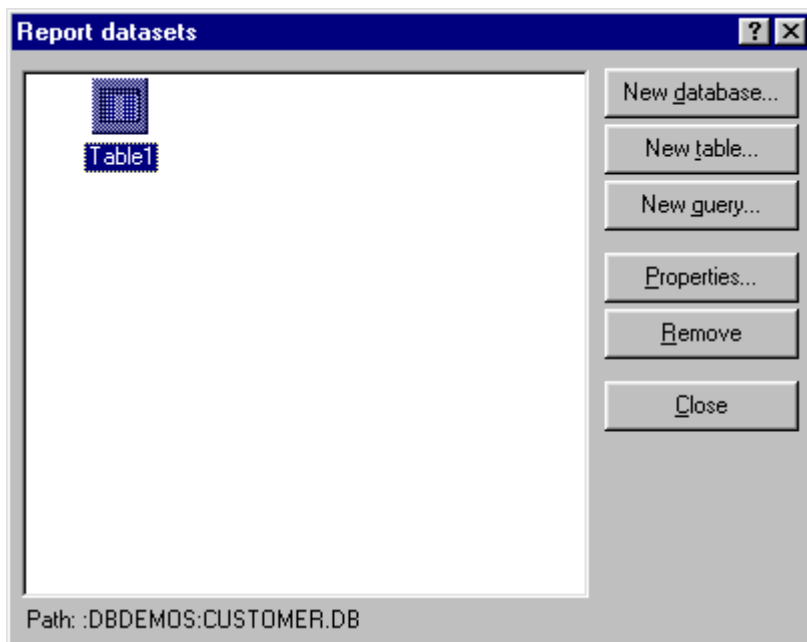
Component TfrDataStorage  is intended to create databases, tables and queries in run-time mode, setup of a list of accessible fields, creation of lookup-fields, master-detail relationship. This component allows to execute the same activity, as in Delphi IDE. The information is saved together with the report form.

**At the present time, component is outdated. Use TfrDialogControls + TfrBDEComponents instead.**

When component is connected, "Data Manager" and "Parameters dialog" added to the list of Designer Tools.



First allows to work with the data - to determine the databases, tables and queries and edit them. The second tool is intended for editing the parameters of the query. All databases, tables and queries created with this tool are in datamodule of "ReportData", which one is accessible in run-time mode. For each table or query there is a data source for FastReport with the same name, as well as for the table, but signed underlines ahead.



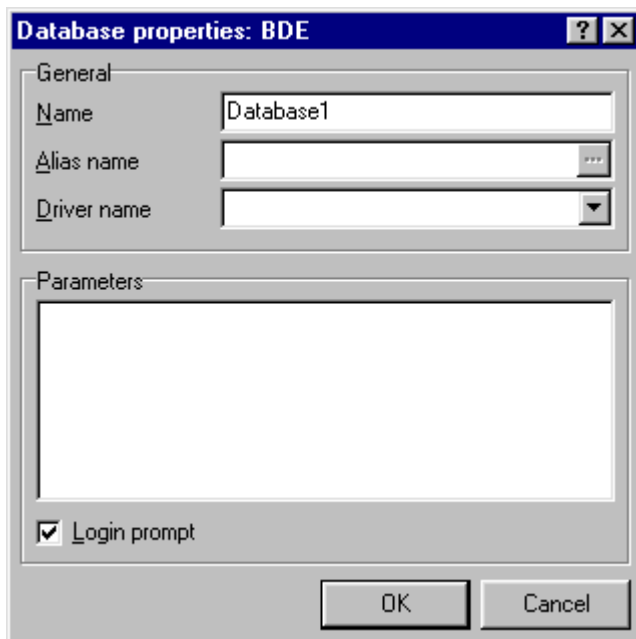
The data manager supports either the BDE, Interbase Express or Microsoft ActiveX Data Objects (ADO). IBX&ADO doesn't require the use of the BDE. To activate the appropriate DB engine, uncomment the appropriate line in the FR.INC file.

## CONNECTING TO A DATABASE

Connecting to a database is necessary if you using ADO or IBX data components. Before creating any tables or queries, you should create database first.

To create new database, start the tool "Data Manager". In the appeared window click on "New database" button. After that the dialog box "Database parameters" will open. If you using ADO

or IBX, set the name of database component and choose appropriate DB file. If you using BDE, set the name of database component, set alias name, which you want to add to the alias list, and choose appropriate DB driver.



You can also set DB parameters, such as user name and password for connection to SQL-based DBs. Note: different drivers uses different parameter names. For example, to connect to Interbase SQL server through BDE native driver, use the following parameters:

```
SERVER NAME=Path to your *.gdb file  
USER NAME=SYSDBA  
PASSWORD=masterkey
```

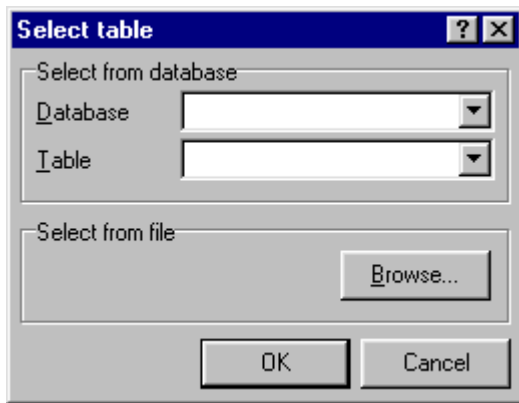
To connect to Interbase SQL server through IBX, use the following parameters:

```
user_name=SYSDBA  
password=masterkey
```

When "Login prompt" checkbox is turned on, database login dialog will appears when you connecting to the SQL server. If you also set user name and password in database parameters, you can turn this option off. This automatically connected to DB.

## OPENING A TABLE

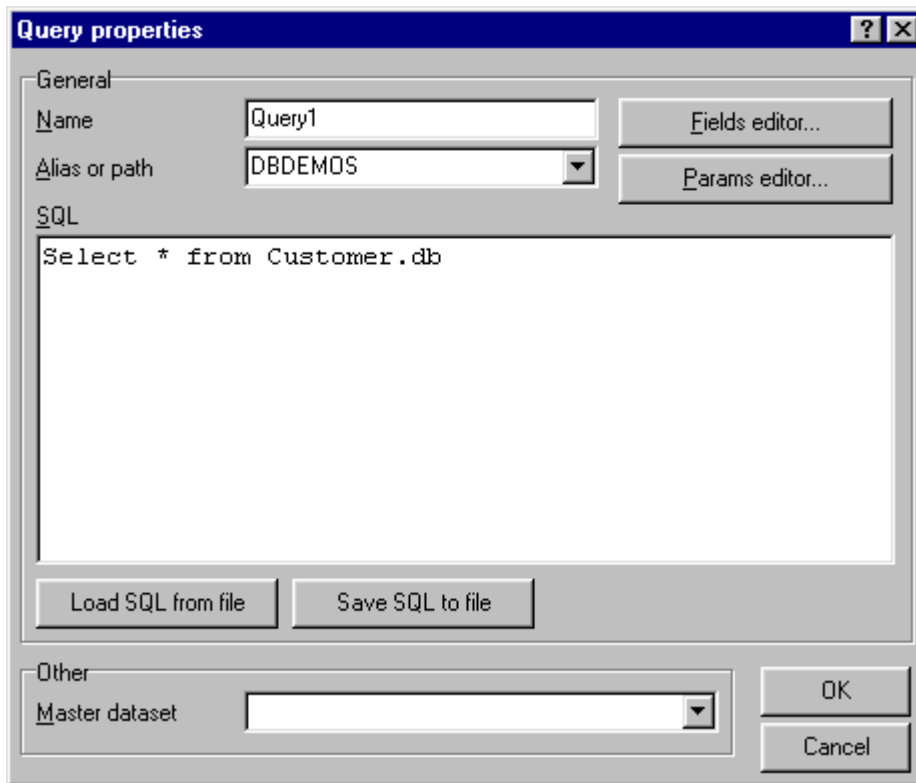
For creation of the new table start the tool "Data Manager". In the appeared window click on "New table" button. After that the dialog box "Select table" will open. You can choose necessary table from available DB's or load it directly from the file (only if using local Paradox or DBase tables with BDE). After the table is selected, the dialog box "Table Properties" will open. Here it is possible to set a table name (by default is offered "Table" + first free number). TDataSource and TfrDBDataSet created together with the table. The name of TfrDBDataSet, as well as for the table, but signed underlines ahead.



Also in this window you can execute fields editor, select index for the table (if table have secondary indexes), set filter expression. To set Master dataset for the table you can select necessary data set from a "Master" list, and in line "Field Master" select fields from a Master dataset (or, clicked in a right part of line to call dialog box for visual linkage of fields).

## GENERATING A QUERY

For creation of the new query start the tool "Date Manager". In the appeared window click on "New query" button. After that the dialog box "Query Properties" will open. Here it is possible to set a query name (by default is offered "Query" + first free number). TDataSource and TfrDBDataSet created together with the query. The name of TfrDBDataSet, as well as for the query, but signed underlines ahead.



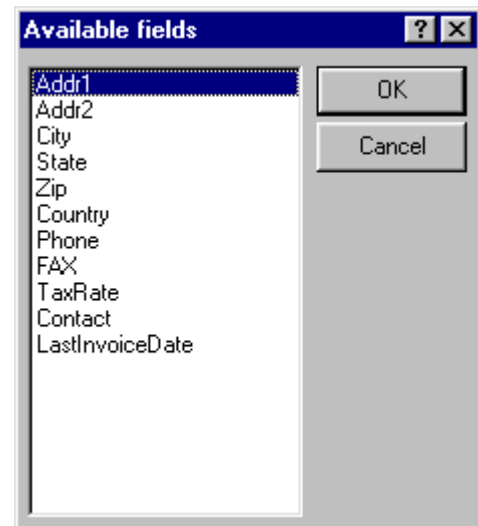
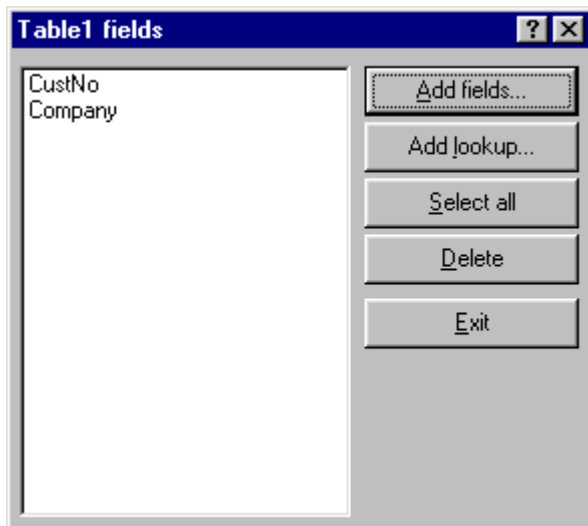
The "Alias" drop-down list contains path of the database tables, on which the query is built. Here there can be a value from a drop-down list or path to the catalog with the tables. It is a field it is possible to leave empty, in this case alias or path it is necessary to indicate in the text of the query. After that it necessary to type the text of the inquiry in a field "SQL Text". If in the text of the query consist the parameters (i.e. identifiers signed ":" in a start of a name), each parameter is necessary for describing in the editor of parameters.

Also in this window you can execute fields editor and set Master dataset. To set the Master dataset for the query you can select from a list "Master" the necessary data set. Thus text of the query should be contained parameters - names of fields from a Master-source. Besides each parameter should have an option "Assign from a Master-source".

At clicking on "OK" button the regularity of the query is checked up. If in the text of the query or in the description of parameters there is an error, the applicable warning will be issued and the editing of properties of the query will proceed.

## **FIELDS EDITOR**

In this editor is possible to fill in a list of accessible fields of the table or query. If the list is empty, all fields will be accessible.



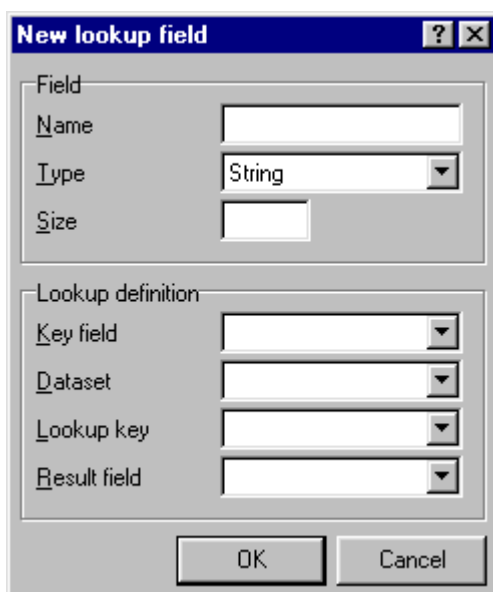
To add fields to the list click on "Add fields" button. From the appeared window select fields and click "OK" button. For selection of group of fields click the mouse on the first field of group, then at the pushed button "Shift" - on last. To add a lookup-field, use the "Add lookup" button. See "Creating lookup field".

For deleting selected fields from a list use "Delete" button.

## CREATING LOOKUP FIELDS

For what the lookup-fields necessary, we shall explain on an example.

Let's allow, there are two database tables: the table "Orders" with fields N, Date, ClientID, Amount and table "Clients" with fields ID, Name, Address. Table "Orders" is contained the information on the orders (number, date, identifier of the client who has made the order, and sum of the order). Table "Clients" is contained the information on clients (identifier of the client, full name, address). To create the elementary report on the table "Orders" like Number of the order - Date - Name of the client - Sum, it is required to link both tables on fields ClientID - > ID. This problem is decided by creation of a lookup-field, which one is added to the table "Orders" and represents the link on a field Name of the table "Clients".



Dialog box of creation of a lookup-field is accessible from the editor of fields. For creation of a lookup-field it is necessary to set it name and type, and also the size (in case is selected the string type). Further it is necessary to fill in following fields:

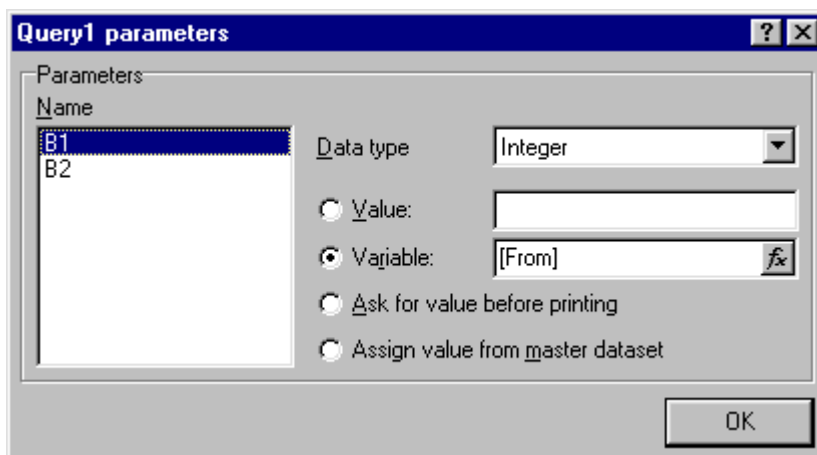


- Primary key field - field in an source dataset, which one as the link on a field from a lookup-dataset. In our example it will be by a field ClientID.
- Datasource - lookup-data set.
- Lookup key - field in a lookup-dataset being key. In our example it is a field ID.
- It is necessary to substitute a resultant field - field of a lookup-set, which one in an source data set. In our example it is a field Name.

After that in the table "Orders" there is a dummy field with a given name, which one contains decryption of a field ClientID. You can access to this field, as to a common field, but only for reading.

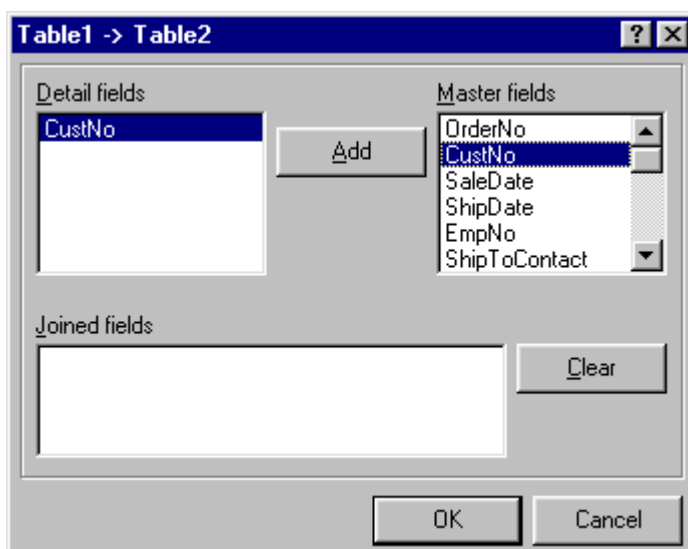
## QUERY PARAMETERS EDITOR

In this dialog box is possible to assign the type to each parameter, and also to point, whence to take an parameter value: from a master-dataset, from dialog box, or at once to assign particular value. In a case, when parameter takes from a master-dataset, the dataset should contain a field with a name conterminous to a name of parameter. Besides it is a field should be in a list of accessible fields (see fields editor). Thus it is optional to indicate the type of parameter.



## JOINING DATA

In this dialog box is possible visually to link fields master and detail of data sets.



When the data sets are connected with each by Master-Detail, at moving on master data set the detail data set is filtering. So it consist only the records that have relation to master data set. For link fields of data sets select fields from list at left side (detail dataset), then select field from list at right side (master dataset), and click "Add" button. Thus the link fields moved to the lower list. To clear lower list push "Clear" button. The linked fields should have the identical type and to be primary key.

## PARAMETERS DIALOG

If the report contains as the data source one or several queries with parameters, which one have an option "Request value" (see editor of query parameters), before creating the report on a screen will be show dialog box of parameters input values. In dialog box all parameters will be collected, which one meet in all queries indispensable for construction of the report.

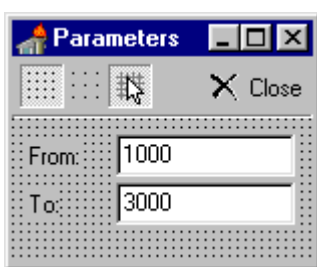
By default for input the value of parameter used command element "Edit box", which one has an description - name of parameter. If some parameters, the command elements are injected the self under the self. It is inconvenient - because parameter name is abbreviated English word. In this case it is possible to take advantage of the Dialogs designer.

The dialog box will be accessible, if the report is true in it requires. The empty report or report which is not inclusive of the links to the queries with parameters, does not show dialog box.

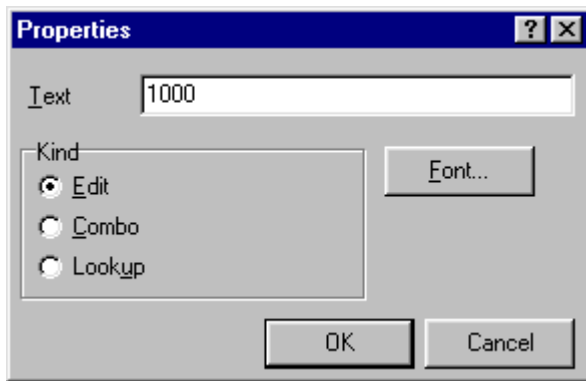
## DESIGNER OF PARAMETERS DIALOG

The Dialogs designer is very good means for change appearance of Dialog box of parameters input values. It allows to arrange control objects of dialog box in the necessary order, to change the descriptions (by default used parameters name). Besides the designer allows to replace the type of a command element for parameters input values - by default command element is "Edit box".

The designer does not allow to add new objects or to remove existing. It is possible only to change their position and sizes, to set properties (for example, parameters font). All operations are make by the mouse. The mouse works like in designer of report form.

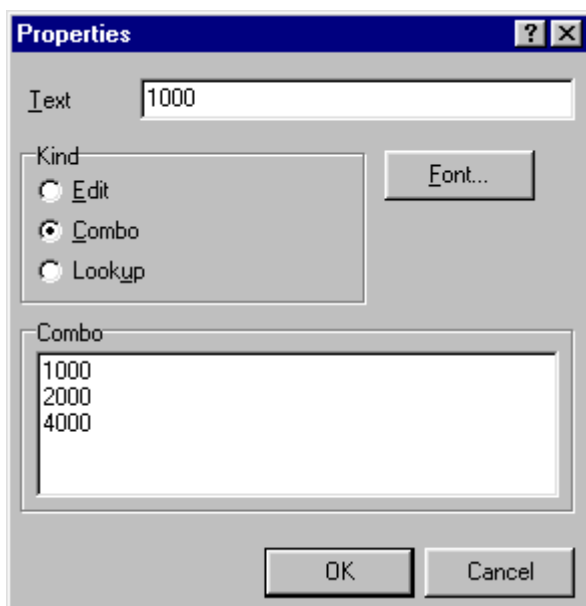


To set properties of object, make a double click by the mouse on a them. In the appeared dialog box it is possible to set the text, which one will be showing in object. The conversation also can be called, if some one-type objects are selected. If the selected object - command element, than in properties dialog box becomes accessible the switch "Object type".



It is possible to select three types:

- Edit box - represents line, in which one it is possible to type any text. If in a field "Text" of dialog box something is typed, this value will be imaged in edit box of input dialog box.
- The list - represents a drop-down list. The useful thing, if is necessary to submit the user selection from several values. If the line of a list contains a character ";", in a list the part of line up to a semicolon will be show, and by selection of this value actually in parameter the part of line after a semicolon will be show. In such a way conveniently, for example to select month: at usage of a list with lines such as "January; 1", "February; 2" in a list the titles of months will be show, and in parameter numbers will get.



- Lookup the list - is convenient for using for selection of value from the reference book (database tables with a primary key field and field inclusive a title). For setup lookup it is necessary to select the table, its primary key field, which one will be substituted in parameter, and the field, which one will be show in drop down list.

It is possible to change the sizes of the form - it is necessary only to allow, that after execute below of dialog box will be added button "OK".

## Built-in language

FastReport has a built-in Pascal-like language interpreter. This interpreter is a powerful means of writing language-independent (Delphi or C++Builder) scripts which are used for building reports.

The language used in the interpreter is a Pascal dialect with the following capabilities:


- operators: assignment operator; conditional statements, loop statements and unconditional jump: **if...then...else**, **while...do**, **repeat...until**, **for..to..do**, **goto**;
- statement parentheses **begin...end**;
- variables without type, arrays;
- references to properties and methods of FastReport objects through dot notation.

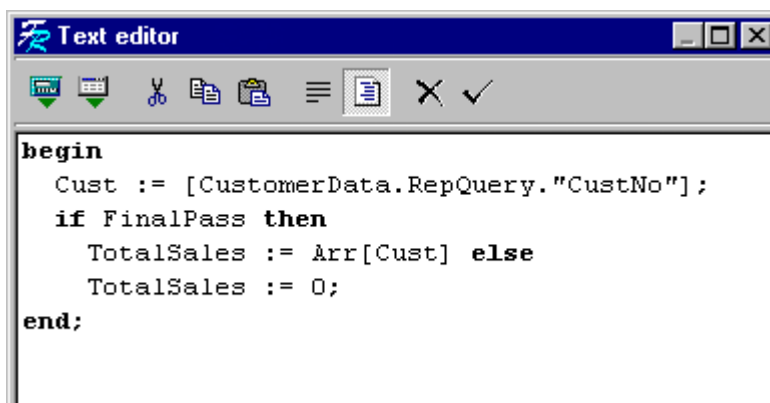
Compared to Object Pascal this language is much simplified. The following simplifications are used:

- all variables are of Variant type; there are no type definitions for variables;
- all variables are global, there are no local variables;
- there are no data types such as class, record, enumeration type etc.;
- you cannot write your own procedures or functions;
- there are no loop breaking operators (break, continue);
- the number of arguments passed to a procedure or a function cannot exceed 3;
- due to the fact that variables have no data type, data type control is unavailable; this should be taken into account when writing logical expressions;
- arrays can only be one-dimensional.

Despite these considerable simplifications, the interpreter allows you to perform rather complex data processing. From the script you have access to all the methods and properties of report objects, as well as to database table fields. In the script you can declare variables and arrays accessible throughout the whole report. Capabilities of the built-in language can be demonstrated by the fact that such a difficult task as printing group totals in the group header (because the totals are calculated at the bottom of the group) for FastReport is elementary.

## Scripts and objects

Each object can have one or more code blocks. Script editing is done in the text editor window (to see the script you have to click the  button at the top of the window). The script runs each time before printing an object. (The script is attached to the OnBeforePrint property of the object).



Not only objects can have scripts. Bands and report pages are scriptable too. To call the band script you must open the OnBeforePrint property editor of the band (either from the Object inspector or by selecting the band and pressing Ctrl+Enter). To call the script of the report page you must open the OnBeforePrint property editor of the page (to do that you can click on an empty place on the page and call the editor from the Object inspector). Both Dialog Forms and Report Pages have scripts attached to their OnActivate property. All other object's scripts can be accessed from their Memo property or by pressing Ctrl+Enter.

## Code writing

In scripts, you can use properties and methods of the report objects, database table fields and various constants. Also you can create variables and arrays accessible throughout the whole report. You can use procedures and functions as well.

### Using variables

There is no need to specify the "type" of variables, they all are variant. You can use Latin letters, digits and underline symbols in variable names. Variables from scripts can be used in objects and the variables from the list of variables can be used in scripts. Script variables are stored in **TfrVariables** object which can be accessed through the **frVariables** global variable.

This is an example of using an intermediate variable:

```
begin
  Cust := [CustomerData.RepQuery."CustNo"];
  if FinalPass then
    TotalSales := Arr[Cust] else
    TotalSales := 0;
end;
```

In this example we create a variable Cust and set it equal to the database table field value.

You can also call variables defined in the data dictionary, system variables and user variables. In this case the variables' names might contain symbols that are not normally allowed by syntax rules (take Page# system variable for example). To call such variables you must use square brackets:

```
begin
  a := [Page#];
end
```

### Referencing database fields

You can use references to database tables in your scripts. Here is the syntax of such a reference:

*[FormName.TableName."FieldName"]*

The full path is used when the table and the report are located on different forms (or data module). If the components are on the same form you can address them as *[TableName."FieldName"]*.

You can just write *["FieldName"]* if you address the table on which the band you are using is based. For example if you have master data band, which is connected to Customer.db table through a data source, you can refer to the fields of this table throughout the report using just the short path. Using the full path will not slow you down - FastReport stores database field names in cache.

## Arrays

Apart from variables you can also create arrays in your scripts. The arrays can only be one-dimensional but you can use their elements in the way that they will be treated as two-dimensional.

Example of using an array:

```
begin
  MyArr[0] := 'a'; MyArr[1] := 'b'; MyArr[3] := 'd';
  MyArr[2] := MyArr[0] + MyArr[1] + 'c' + MyArr[3];
end;
```

Actually the array elements' values are stored in the frVariables list in Arr\_array\_name\_index format. I.e. in the above example the contents of frVariables will be:

```
Arr_MyArr_0 := 'a'
Arr_MyArr_1 := 'b'
Arr_MyArr_2 := 'abcd'
Arr_MyArr_3 := 'd'
```

## Constants

You can use constants in your scripts. A simple example is using numeric, string and logical constants:

```
begin
  a := 0;
  b := 'abcd';
  c := True;
  d := 'That''s all!';
end;
```

Pay attention to using single quotes inside string constants – like in Pascal, they have to be duplicated: d := 'That''s all!'.

Apart from simple constants you can use such constants as color names, font type names etc. Below is the list of available constants:

- colors: clWhite, clBlack etc. – all standard colors + system colors;
- dialog box response constants: mrNone, mrOk, mrCancel;
- system: CRLF, Null;
- font style types: fsBold, fsItalic, fsUnderline;
- object frames: frftNone, frftRight, frftBottom, frftLeft, frftTop;
- text alignment in a“Text” object: frtaLeft, frtaRight, frtaCenter, frtaVertical, frtaMiddle, frtaDown;
- band aligning: baNone, baLeft, baRight, baCenter, baWidth, baBottom.

Besides these there are constants for add-in objects, for example, *csCheck* for "CheckBoxObject" objects. Anything you can see in a drop-down property list box in the Object inspector can be used as a constant in scripts.

## Referencing objects

You can reference a report's object's properties and methods in your scripts. Report objects are visual objects, control objects, bands, report pages and reports themselves. To reference an object

dot notation is used, for example: *Memol.Text*. To reference intrinsic properties and methods dot notation is not necessary.

Properties that can be referenced are shown in the Object inspector. Some combined properties like those of Font can be referenced by using *Font.Name*, *Font.Size* etc.:

```
begin
  Memol.Font.Name := 'Courier New';
  Memol.Font.Size := 10;
  Memol.Font.Color := clRed;
  Memol.Font.Style := fsBold + fsItalic
end;
```

Properties such as TStrings (Memo, SQL, Items etc.) can be referenced by their index:

```
if Memol.Lines[1] = 'a' then
  Memol.Lines[1] := 'b'
```

Such properties can also be referenced using Add, Delete, Clear and Count:

```
if Memol.Lines.Count > 10 then
  Memol.Lines.Delete(10)
else
begin
  Memol.Lines.Clear;
  Memol.Lines.Add('a');
end;
```

A Full list of object properties and methods can be found in the “Object properties and methods” paragraph. Make a note that referencing a non-existent method or property will not cause an error message, so be careful when writing code.

## Using procedures and functions

Scripts can contain procedures and functions calls. A peculiar property of the interpreter, or to be more precise the parser responsible for procedure/function processing, is that procedures and functions can not have more than 3 arguments. Scripts can use both built-in procedures and external procedures, defined in the project. The list of built-in procedures and functions is in the “Built-in procedures and functions” paragraph.

**Note.** When referencing a procedure or function there must not be a space between the procedure name and the opening bracket.

## Objects modification

In your scripts you can make any modifications to objects, like changing size, color, contents, etc. *You must remember that in a single pass report you cannot modify objects which have already been processed.* That is if you try to change the contents of an object in the report title from an object lying on a report summary band there will be no changes. However, you will be able to make such a modification if you make a two-pass report.

It works the same way in multi-page reports. You can reference any object by its name (names within a report are unique). But only non-processed objects can be modified. If you still need to modify an object which has already been processed you will have to make a two-pass report.

## Built-in functions

### Aggregate functions

Aggregate functions can be used in ReportSummary, PageFooter, MasterFooter, DetailFooter, SubdetailFooter, GroupFooter and CrossFooter bands.

- **Sum(<expression> [, band] [,1]).** Calculates the sum of the values passed in <expression> for the band row given. If the band parameter is not set, sum defaults to all of the data values (on the bands MasterData, DetailData and SubdetailData); otherwise a sum refers only to data on the named band. If the “1” parameter is used, the sum includes non-visible objects too. Example:  
Sum([Part total], Band1);  
Sum([Part total] + [Part price]);  
Sum([Part total], Band1, 1).
- **Avg, Min, Max.** Syntax is analogous to the Sum function. Function Avg calculates an arithmetic average, function Min returns minimum and function Max returns maximum value from a row.
- **Count(<band>).** Returns a count of data-rows. Example: Count(Band1).

### String functions

- **Str(<value>).** Converts number given in value to a string.
- **Copy(<string>, <from>, <count>).** Returns a substring of <string> with length <count> characters, starting at <from>, (same as Delphi function).
- **If(<expression>, <string1>, <string2>).** Returns string <string1>, if expression expression is true; otherwise returns <string2>.
- **FormatFloat(<formatstr>, <value>).** Converts a numerically significant value in string, making use of the mask in formatstr. The possible values of <formatstr> are described in the Delphi documentation's, «Formatting strings» topic.
- **FormatDateTime(<formatstr>, <value>).** Converts a date/time value to a string, making use of the mask in <formatstr>. The possible values of <formatstr> are as described in the Delphi documentation's, «Formatting strings» topic.
- **StrToDate(<value>).** Converts the string <value> to a date.
- **StrToTime(<value>).** Converts the string <value> to a time.
- **UpperCase(<value>).** Converts the string <value> to all upper case.
- **LowerCase(<value>).** Converts the string <value> to all lower case.
- 
- **NameCase(<value>).** Converts the first character of string <value> to upper case and the remaining characters to lower case.
- **Length(<string>).** Returns the length of <string>.
- **Trim(<string>).** Trims (removes) all spaces at the beginning and end of <string> and returns the result.
- **Pos(<substring>, <string>).** Returns the position of <substring> in the given <string>.

### Arithmetic functions

- **Int(<value>).** Returns the whole part of the number <value>.
- **Frac(<value>).** Returns the fractional part of the number <value>.
- **Round(<value>).** Returns rounded number.
- **value1 Mod value2.** Returns the remainder resulting from dividing <value1> by <value2>.
- **MinNum(<value1>, <value2>).** Returns the smaller of the two values.
- **MaxNum(<value1>, <value2>).** Return the greater of the two values.



## Other functions

- **Input(<caption> [, <default>])**. Shows a dialog window with the heading <caption> and an edit box. If “default” parameter is set, puts this string into the edit box. After user clicks OK, returns the input string.
- **Date**. Returns current system date.
- **Time**. Returns current system time.
- **Line#**. Returns the current Line number; counting begins at the start of each new group. For example:  
Master data  
    1. Detail data  
    2. Detail data  
    3. Detail data  
Master data  
    1. Detail data  
    2. Detail data
- **LineThrough#**. Returns the current Line number; counting begins at the start of the report. For example:  
Master data  
    1. Detail data  
    2. Detail data  
    3. Detail data  
Master data  
    4. Detail data  
    5. Detail data
- **Column#**. Returns the current column number in a cross-tab report.
- **Page#**. Returns the current page number.
- **TotalPages**. Returns the total number of pages in a completed report. To use of this function a report must be a two pass report.
- **DayOf(<date>)**. Returns day (1..31) of given date.
- **MonthOf(<date>)**. Returns month of given date.
- **YearOf(<date>)**. Returns year of given date.
- **MessageBox(<text>, <caption>, <buttons\_and\_icons>)**. Shows a message dialog window with text, caption and buttons. Returns a value that corresponds to the user’s selection (mrOk, mrCancel, mrYes, mrNo). Use the following values for the <buttons\_and\_icons> parameter:

Buttons	Icon
mb_Ok	mb_IconError
mb_OkCancel	mb_IconQuestion
mb_YesNo	mb_IconInformation
mb_YesNoCancel	mb_IconWarning

## Procedures and functions that can be used during building a report

- **CurY**. Returns the current Y position where the next band will be printed. You also can assign a value to CurY – it moves the current position accordingly. *To convert pixels to millimeters and back, use the following ratio: 18 pixels = 5mm.*
- **FreeSpace**. Return space remaining in the page in pixels.
- **FinalPass**. Returns true, if a report is two-pass and is now running the final pass.
- **PageHeight**. Returns page height in pixels minus the height of the page footer band.
- **PageWidth**. Return page width in pixels.
- **StopReport**. Terminates report building.

- `NewPage`. Starts a new page.
- `NewColumn`. Starts a new column in a multi-column report.
- `ShowBand(<band>)`. Shows band named <band>.

## Properties and methods of objects

All visual objects of the report are descendants of the class TfrView. The following properties and methods may be addressed from a script:

Property	Type	Description
BandAlign	Integer	Specifies the alignment of objects within a band. Possible values are: baNone, baLeft, baRight, baCenter, baWidth, baBottom.
Enabled	Boolean	Determines whether or not an object can respond to an event. Possible values: True or False.
FillColor	Integer	The background color of an object. The color can be specified by a constant clXXX.
FrameColor	Integer	Color of a frame of object.
FrameStyle	Integer	Specifies the style of a border. Possible values are: psSolid, psDash, psDot, psDashDot, psDashDotDot, psDouble.
FrameTyp	Integer	Type of border of an object - a set of constants frftTop, frftBottom, frftLeft, frftRight.
FrameWidth	Double	Width of a frame.
Height	Integer	Height of the selected object.
Left	Integer	Determines the horizontal coordinate of the left edge of an object relative to the form in pixels.
Memo	String	The text in memo of the selected object. The property can be accessed by reference to its index, for example: Memo[1].
Memo.Count	Integer	Returns the number of lines in a memo.
Name	String	Name of the selected object.
Stretched	Boolean	Whether the object is stretched to assume the size and shape of the control or retains its natural dimensions.
Top	Integer	Determines the y coordinate of the top left corner of an object relative to its parent.
Visible	Boolean	Determines whether or not the object appears on the screen. Possible values: True or False.
Width	Integer	Determines the width (horizontal size) of an object

Methods:

Method	Parameters	Description
Hide	-	Makes an object invisible by setting the visible property to False.
Memo.Add	String	Adds a new line to a string list in a memo.
Memo.Clear	-	Deletes all text from an object.
Memo.Delete	Integer	Removes a line specified with the given index parameter.
Show	-	Makes an object visible by setting the visible property to True.

## Standard objects

### Object "Text" (TfrMemoView)

In addition to the general properties and methods described above, an object has its own properties:

Property	Type	Description
Alignment	Integer	Specifies the alignment of text within an object. Possible values are: frtaLeft, frtaRight, frtaCenter, frtaVertical, frtaMiddle, frtaDown.
AutoWidth	Boolean	Determines if the object automatically resizes to the width of the text in object.
CharSpacing	Integer	Determines the space between characters.
Font.Name	String	Allows particular fonts to be specified to control the attributes of the text of an object.
Font.Size	Integer	The size of a font in pixels.
Font.Style	Integer	Determines whether the font has any of the attributes: bold, italic, underlined. Possible values: fsBold, fsItalic, fsUnderline.
Font.Color	Integer	Determines the color of the font.
GapX	Integer	Horizontal distance between an object's border and the text within that object.
GapY	Integer	Vertical distance between an object's border and the text within that object.
HideZeros	Boolean	If set to True then zero values of variables are ignored. Possible value: True; False.
LineSpacing	Integer	Spacing between lines of the text.
Suppress	Boolean	Specifies whether or not repeat values are suppressed. Possible values: True; False.
TextOnly	Boolean	Specifies whether or not variables are processed. If Textonly is set to True then variables are not processed. Possible values: True; False.
WordBreak	Boolean	When a word wraps at the right margin, Wordbreak specifies whether wrapping occurs at the end of a syllable. Possible values: True; False (Russian words only).
WordWrap	Boolean	Determines if text wraps at the right margin so that it fits into the object. Possible values: True; False.

### Object "Band" (TfrBandView)

Property	Type	Description
Breake	Boolean	Band breaks off. Possible values: True; False.
ChildBand	String	Band is derived from another band.
ColumnGap	Integer	Horizontal distance between columns within an object.
Columns	Integer	Number of columns in band.
ColumnWidth	Integer	Width of a column.
Condition	String	Specifies conditions for grouping. Applies to band group header.
DataSource	String	Determines where the object obtains the data it is to display.

EOF	Boolean	Determines whether or not the end of a dataset has been reached. Possible values: True; False.
FormNewPage	Boolean	Forces printing on a new page after printing this band and all its detailed bands. Possible values: True; False.
Master	String	Specifies the band which is used when grouping data.
OnFirstPage	Boolean	Print on the first page. Possible values: True; False.
OnLastPage	Boolean	Print on the last page. Possible values: True; False.
PrintChildIfInvisible	Boolean	Specifies whether or not to print child bands if the child's parent band is invisible. Possible values: True; False.
PrintIfSubsetEmpty	Boolean	Specifies whether or not to print a band if its child's band is empty. Possible values: True; False.
RepeatHeader	Boolean	Specifies whether or not to repeat this band on every page. Possible values: True; False.

Methods:

Method	Parameters	Description
First	-	Establishes the datasource for a band based on the first record.
Next	-	Establishes the datasource for a band based on the next record.
Prior	-	Establishes the datasource for a band based on the previous record.

### Object "Picture" (TfrPictureView)

Property	Type	Description
BlobType	Integer	Specifies the type of image contained within a blob field. Possible values: btBMP, btJPG, btICO, btWMF.
Center	Boolean	Specifies whether or not to centre an image within an object. Possible value: True; False.
DataField	String	Specifies the field of the table that contains the image.
KeepAspect	Boolean	Specifies whether or not to retain the relative proportions of an image when it is resized. Possible values: true; False.

Methods:

Method	Parameters	Description
LoadFromFile	String	Loads a picture from the file. If file does'nt exists, clears the picture.

## Add-in objects

### Object "Bar code" (TfrBarcodeView)

Property	Type	Description
DataField	String	The name of the field that contains the data.

### Object "CheckBox" (TfrCheckBoxView)

Property	Type	Description
CheckColor	Integer	Specifies the color of the cross when the checkbox. is checked
CheckStyle	Integer	Specifies the style of the checkbox. Possible values: csCross, csCheck.
DataField	String	The name of the field that contains the data.

### Objects "RichText", "RichText 2.0" (TfrRichView, TfrRXRichView)

Property	Type	Description
GapX	Integer	Horizontal distance between an object's border and the text within that object.
GapY	Integer	Vertical distance between an object's border and the text within that object.
TextOnly	Boolean	Specifies whether or not variables are processed. If Textonly is set to True then variables are not processed. Possible values: True; False.
DataField	String	The name of the blob field that contains the data.

### Object "Rectangle with a shadow" (TfrRoundRectView)

Since this object is a descendant of the object "Text", it possesses the same set of properties and methods as the parent together with the following:

Property	Type	Description
BeginColor	Integer	Specifies the initial color used in the gradient.
EndColor	Integer	Specifies the final color used in the gradient.
Gradient	Boolean	Specifies whether or not to use gradient flooding. Possible values: True; False.
RoundRect	Boolean	Specifies whether the corners of the rectangle are rounded or not. Possible values: True; False.
RoundSize	Integer	If the corners of the rectangle are rounded this property specifies the degree of rounding.
ShadowColor	Integer	Color of a shadow.
ShadowWidth	Integer	Width of a shadow.
Style	Integer	If the gradient effect is used then this property specifies the style of the gradient. Possible values: gsVertical, gsHorizontal, gsElliptic, gsRectangle, gsHorizCenter, gsVertCenter.

**Object "Shape" (TfrShapeView)**

Property	Type	Description
Shape	Integer	Determines the visual shape of an object. Possible values: skRectangle, skRoundRectangle, skEllipse, skTriangle, skDiagonal1, skDiagonal2.

## Dialog controls

All dialog controls are descendants of the class TfrStdControl and possess the following set of properties and methods:

Property	Type	Description
Color	Integer	The background color of an object. The color can be specified by a constant clXXX.
Enabled	Boolean	Determines whether or not an object can respond to an event. Possible values: True or False.
Font.Name	String	Allows particular fonts to be specified to control the attributes of the text of an object.
Font.Size	Integer	The size of a font in pixels.
Font.Style	Integer	Determines whether the font has any of the attributes: bold, italic, underlined. Possible values: fsBold, fsItalic, fsUnderline.
Font.Color	Integer	Determines the color of the font.
Height	Integer	Height of object.
Left	Integer	Determines the horizontal coordinate of the left edge of an object relative to the form in pixels.
Name	String	Name of object.
Top	Integer	Determines the y coordinate of the top left corner of an object relative to its parent.
Visible	Boolean	Determines whether or not the object appears on the screen. Possible values: True or False.
Width	Integer	Determines the width (horizontal size) of an object

Methods:

Method	Parameters	Description
Hide	-	Makes an object invisible by setting the visible property to False.
SetFocus	-	Gives the input focus to the control.
Show	-	Makes an object visible by setting the visible property to True.

## Object "Label"

Property	Type	Description
Alignment	Integer	Specifies the alignment of a line of text within an object. Possible values: taLeftJustify, taRightJustify, taCenter.
AutoSize	Boolean	Determines if the object automatically resizes to the width of the label text. Possible values: True; False.
Caption	String	The caption is the character string that is displayed on the label.
WordWrap	Boolean	Specifies if text wraps onto another line so that it fits onto the label. If set to true, then AutoSize should be set to False. Possible values: True; False.

## Object "Edit"



Property	Type	Description
ReadOnly	Boolean	Specifies whether or not a user can change the contents of an edit control. If readonly is set to true then the value cannot be changed. Possible values: True; False.
Text	String	Specifies the text that appears within the edit box.

### Object "Memo"

Property	Type	Description
Lines	String	Specifies the text lines in a memo object. Individual lines can be accessed by reference to a line's index e.g. Memo1.Lines [0].
Lines.Count	Integer	The number of lines of text in a memo
ReadOnly	Boolean	Specifies whether or not a user can change the contents of an memo control. If readonly is set to true then the text cannot be changed. Possible values: True; False.
Text	String	Specifies the text that appears in the memo object. It consists of all the line concatenated as one line (with symbols CR+LF for carriage return and line feed).

Methods:

Method	Parameters	Description
Lines.Add	String	Adds a new line to a string list in a memo.
Lines.Clear	-	Deletes all text from an object.
Lines.Delete	Integer	Removes a line specified with the given index parameter.

### Object "Button"

Property	Type	Description
Caption	String	The caption property is the text that appears on the button.
ModalResult	Integer	When the user chooses to close a dialog box by pressing a button the button click sets ModalResult to close the box. The value assigned to ModalResult becomes the return value of the ShowModal function call which displayed the dialog box. Possible values: mrNone, mrOk, mrCancel.

### Object "CheckBox"

Property	Type	Description
Alignment	Integer	Specifies the alignment of text relative to the box. Possible values: taLeftJustify, taRightJustify.

Caption	String	The caption is the character string that is displayed on the checkbox.
Checked	Boolean	Specifies whether a checkbox is checked (True) or unchecked (False). Possible values: True; False.

### Object "RadioButton"

Property	Type	Description
Alignment	Integer	Specifies the alignment of text relative to the switch. Possible values: taLeftJustify, taRightJustify.
Caption	String	The caption is the character string that is displayed on the checkbox.
Checked	Boolean	Specifies whether a radiobutton is checked (True) or unchecked (False). Possible values: True; False.

### Object "ListBox"

Property	Type	Description
Items	String	The items array holds the lines that will be displayed in the listbox. Individual lines in the listbox can be accessed by reference to the index of the item e.g. ListBox1. Items [0].
ItemIndex	Integer	Index of the selected line.
Items.Count	Integer	The number of lines in the array displayed by the listbox.

#### Methods:

Method	Parameters	Description
Items.Add	String	Adds a new line to the string list displayed in a listbox.
Items.Clear	-	Deletes all lines from the string list displayed in the listbox.
Items.Delete	Integer	Removes a line from the string list displayed in the listbox. The line to be deleted is specified with the given index parameter.

### Object "ComboBox"

Property	Type	Description
Items	String	The Items array holds the lines that will be displayed in the combobox. Individual lines in the combobox can be accessed by reference to the index of the item e.g. ComboBox1. Items [0].
ItemIndex	Integer	Index of the selected line.
Items.Count	Integer	The number of lines in the array displayed by the combobox.
Style	Integer	The style property determines how a combo box displays its items. These can be as a dropdown list with an edit box in which text may be entered (csDropDown), as a dropdown list with no edit box so the item cannot be edited (csDropDownList) or as a list from another source (csLookup). Possible values: csDropDown, csDropDownList, csLookup.
Text	String	Specifies the text that appears selected in the combobox.

Methods:

Method	Parameters	Description
Items.Add	String	Adds a new line to a string list displayed in a combobox.
Items.Clear	-	Deletes all lines from the string list displayed in the combobox.
Items.Delete	Integer	Removes a line from the string list displayed in the combobox. The line to be deleted is specified with the given index parameter.

## Data access components

### Object "BDELookupComboBox"

The object is successor of TfrStdControl and has the same base set of properties and methods plus the properties:

Property	Type	Description
KeyField	String	Field - identifier of chosen value.
ListField	String	Field whose values are displayed in the list.
ListSource	String	Source of the data.
Text	String	The chosen value.

### Object "BDETable"

Property	Type	Description
Active	Boolean	Determines, whether the table is active.
DatabaseName	String	Name of DB's alias.
Fields	Variant	The list of accessible fields. Property can be accessed by index - name of a field: a: = Table1. Fields ['Customer'].
FieldCount	Integer	Number of fields in a dataset.
Filter	String	Expression for records filtering.
IndexName	String	Name of a secondary index.
MasterFields	String	The fields used for joining with a master dataset.
MasterSource	String	Master dataset.
TableName	String	Name of DB table.
EOF	Boolean	True if the end of a set of records is achieved.
RecordCount	Integer	Number of records in the table.

Methods:

Method	Parameters	Description
Open	-	Opens the table. It is similar to substitution Active: = True.
Close	-	Closes the table. It is similar to substitution Active: = False.
First	-	Establishes the index on the first record in the table.
Last	-	Establishes the index on last record in the table.
Next	-	Establishes the index on the following record in the table.
Prior	-	Establishes the index on the previous record in the table.

### Object "BDEQuery"

Property	Type	Description
Active	Boolean	Determines, whether the query is active.
DatabaseName	String	Name of DB's alias.
DataSource	String	Master dataset.

Fields	Variant	The list of accessible fields. Property can be accessed by index - a name of a field: a: = Query1. Fields ['Customer'].
FieldCount	Integer	Number of fields in a dataset.
Filter	String	Expression for records filtering.
SQL	String	The text of query. Property can be accessed by index: Query1. SQL [0].
SQL.Count	Integer	Number of lines in the query's text..
EOF	Boolean	True if the end of a set of records is achieved.
RecordCount	Integer	Number of records in the table.

Methods: the same, as in component BDETable and some more:

Method	Parameters	Description
SQL.Add	String	Adds a line.
SQL.Clear	-	Clears lines.
SQL.Delete	Integer	Deletes a line with the given index.

### Object "BDEDataBase"

Property	Type	Description
AliasName	String	Alias whose adjustments are used for connecting to DB.
Connected	Boolean	If True, the connection is active.
DatabaseName	String	Name, which will be added into the list of aliases.
DriverName	String	Name of the driver providing connection to DB.
LoginPrompt	Boolean	Defines, whether it is necessary to request user to enter DB's password.
Params	String	Parameters of connection. Property can be accessed by index: DataBase1. Params [0].
Params.Count	Integer	Number of parameters' lines.

Methods:

Method	Parameters	Description
Params.Add	String	Adds a line.
Params.Clear	-	Clears lines.
Params.Delete	Integer	Deletes a line with the given index.

## Using the interpreter

Here are some examples of using the interpreter:

1. To highlight sum of an order: white, if sum is less than 2000; green, if sum is between 2000 and 10000; red, if sum is greater than 10000, type the following script in object with sum:

```
if [Summa] < 2000 then
  FillColor := clTransparent
else if [Summa] < 10000 then
  FillColor := clGreen
else
  FillColor := clRed
```

[Summa] is your variable or field with actual sum of order. You can use numeric constants for choosing color:

```
FillColor := 128 + 128*256 + 128*65536 // (gray color)
```

2. To show only data rows with order's sum greater than 2000, use the following code in the script of appropriate data band (you can show its memo editor by pressing Ctrl+Enter key):

```
if [Summa] > 2000 then
  Visible := 1 else
  Visible := 0
```

3. Use FreeSpace function to determine how much free space is on current page. If there is not enough space, call NewPage procedure to insert "hard page break" and start new page.

```
if FreeSpace * 5/18 < 30 then NewPage
```

4. Use CurY property to move current position. For instance, to put Report summary band to the bottom of page, use this code in the report summary script:

```
CurY := PageHeight - Height
```

# Programming

Event handlers

Variables

Expanding the functionality

## Event handlers

Often data needs to be extracted from other sources, which are not databases (for example, from file, array, etc.). There is TfrUserDataset component for these purposes. It generates following events: OnFirst, OnNext, OnCheckEOF. Besides, OnGetValue and OnBeforePrint event handlers of TfrReport component have to be implemented.

OnGetValue event handler is invoked each time there is a variable in the text of object and it is necessary to get its value. Internal handler of the component can handle the variable if it has assigned value. Otherwise it is necessary to attach external handler, for example:

```
procedure TForm1.Doc1GetValue(const ParName: string; var ParValue: Variant);
begin
    if ParName = 'Var1' then
        ParValue := '1'
    else if ParName = 'Var2' then
        ParValue := 2
end;
```

OnBeforePrint event handler is invoked before plotting any object. Usually it is used to load contents of memo-field or picture from DB into the object. An example of the handler is showed below:

```
procedure TForm1.Doc1BeforePrint(Memo: TStringList; View: TView);
begin
    if Memo.Count > 0 then
        if Memo[0] = '[Memo]' then
            Memo.Assign(Table1Memo)
        else if (Memo[0] = '[Picture]') and (View is TPictureView) then
            (View as TPictureView).Picture.Assign(Table1Picture);
end;
```

OnUserFunction event handler is invoked when variable or expression containing function call is found in the text of the object. The function can have up to 3 parameters of any type. An example of the handler:

```
procedure TForm1.Doc1UserFunction(const name: string; p1, p2, p3: Variant;
    var val: Variant);
var
    d: Double;
begin
    if name <> 'MONEYTOSTR' then Exit;
    d := frParser.Calc(p1);
    val := MyMoneyToStr(d);
end;
```

TfrUserDataset component located on the graphic palette of FastReport components is used to navigate through the data sources, which are not DB (for example, through arrays). It generates OnFirst, OnNext, and OnCheckEOF events; obviously the way of using them is clear.

## Other events of TfrReport object

Apart from the events described in the above paragraph FastReport has a number of other event handlers:



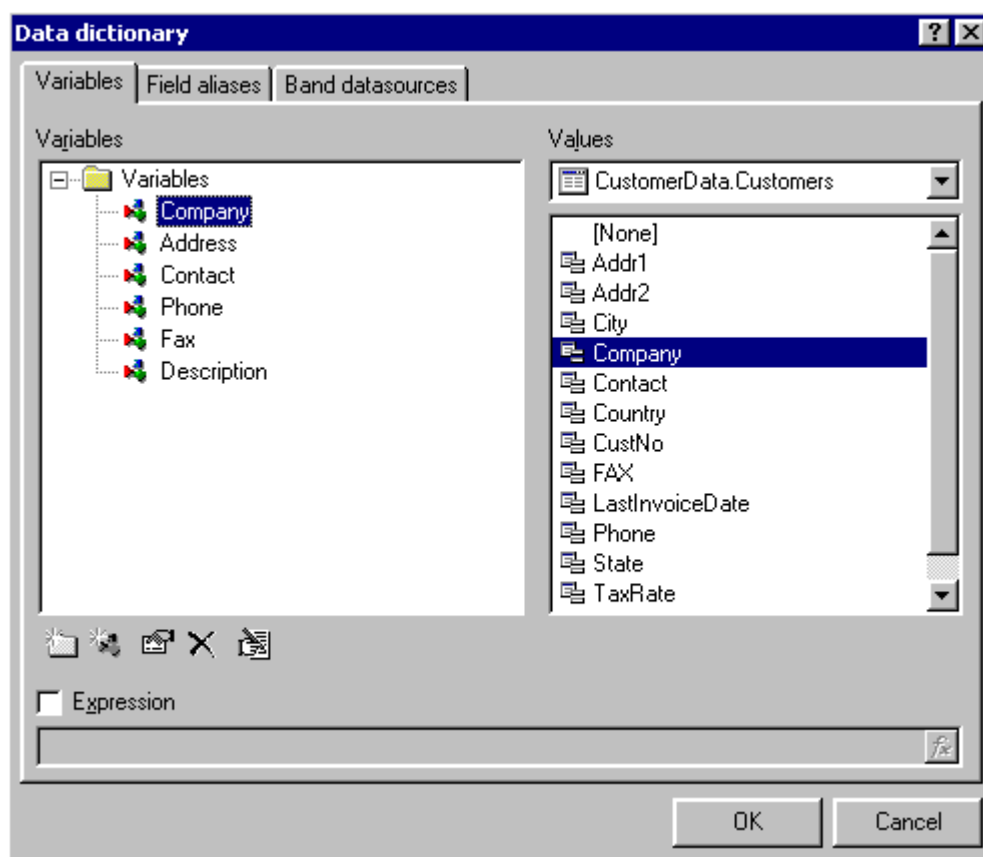
- OnBeginBand – event handler is called before the data section is formed. Reference to the section whose data is to be processed is passed to the handler as an argument.
- OnBeginColumn – event handler is called when reports with alternate number of columns are formed. Band parameter shows for which section the multi-column part is being built at the moment. The event handler can be useful for switching data sources (the multi-column part can have a title, first-level data, second-level data, etc., for which different data sources are used).
- OnBeginDoc – event handler is called before the report is formed.
- OnBeginPage – event handler is called before the page is formed.
- OnEndBand – the event is generated right after the data section is formed.
- OnEndDoc – event handler is called after the report is formed.
- OnEndPage – event handler is called after the page is formed.
- OnManualBuild – event handler allows building reports manually (by coding). If no handlers are assigned, the report is built as usually. If the report is built manually you can change the section output order.
- OnPrintColumn – the event is called when cross-tab report columns are formed. The column index and its default width are passed to the event handler. Altering the width you can get cross-tab reports with different column widths. In this case you will also have to alter the width of the object(s) in the table cells.
- OnProgress – event handler is called for indicating current task progress (report building, printing, exporting). This handler can use your own indicator.

## Variables

You can use variables in FastReport objects and expressions. What does this mean and what purpose do they serve?

Suppose one has a table for employees with field names N, Name1, Name2 and Name3. Sometimes it is difficult to later remember what such field names stand for, even for the database developer. Instead, one can assign conventional names (Table Number, Last Name, First Name, Middle Name) to the database fields and insert them into the objects. In addition, variables can be assigned not only to database field names but also to mathematical expressions. An example could include a running sum of a database field, a rounded value, a date, time value. More examples of using variables in any report can be found in the demo version of the program.

To work with a list of variables one must open a dialog box from the menu "File|Data Dictionary...".



The list of variables is in the left part of the screen. As one can see on the picture, the list has a two-level structure: It consists of categories and each of them can have one or a few variables. Categories are only used for visual grouping of the variables and are not inserted into reports. A more detailed description of this technique can be seen in "Designer" paragraph.

However, FastReport cannot work alone with the variables described in the data dictionary. If an unknown identifier is met in the code, FastReport successively checks if:

- There is such a variable in the data dictionary;
- It is a database field;
- It is one of special values of Page#, Date, Time or other types;
- It is one of variables from frVariables list;
- It is an object property;
- It is a constant from frConsts list.

Also, if a component has a turned on OnGetValue event handling, it is called before all the checks. If the handler returns a value, the variable is considered initialized and no further checks are made.

This approach gives much flexibility in choosing a way of passing a value to the report. It is recommended:

- When one passes static value which does not change from record to record, one can use a frVariables global object, such as:

```
frVariables['Reporting Period'] := 'January';  
frReport1.ShowReport;
```

- When one passes values which do change from record to record, one can use either a data dictionary or a TfrReport.OnGetValue event handler, such as:

```
procedure TForm1.frReport1GetValue(const ParName: String; var ParValue: Variant);  
begin  
  if AnsiCompareText(ParName, 'Reported Period') = 0 then  
    ParValue := Table1OtchPeriod.Value;  
end;
```

- One can also fill the list of variables programmatically:

```
with frReport1.Dictionary do  
begin  
  Variables['Number'] := 1;  
  Variables['Sum'] := '0.2 * Table1."Summa"';  
  Variables['Date'] := ''' + 'January' + ''';  
end;
```

*(An extra pare of quotes in this example is necessary for assigning string constants).*

FastReport regards string values, assigned to variables from data dictionary, as expressions to be computed. If a variable is not used in the data dictionary it works as usual.


One must remember that if you use variables from frVariables list or if you use them through OnGetValue event handler, they need not be put into the data dictionary.

## Extending FastReport functionality

FastReport report generator is open, meaning its functionality can be extended by writing your own function libraries, new visual components, export filters and wizards. This section contains information on the ways to develop such extensions.

### Making your own preview windows

Sometimes it is necessary to swap the standard preview window with your own. This may be caused by the necessity to run additional functions before the output (for example inserting the printing date value into the table) or the built-in preview window may not meet the needs of the developer.

Either way, FastReport allows developers to build their own preview windows. TfrPreview component  on the FastReport component palette is exactly for that purpose. The primary commands of this component are:

- First, Prev, Next, Last – for navigating to the first, the previous, the next and the last page of the report respectively;
- SaveToFile – saves the prepared report in a file of any of the supported formats;
- LoadFromFile – loads a pre-prepared report from a file in \*.FRP format;
- Print – prints the prepared report;
- OnePage – sets the scale of preview so that one whole page will be viewed;
- PageWidth – sets the scale for viewing a page by its width.

In addition to the commands outlined above, the TfrPreview component has a Zoom property which sets the page preview scale. The preview scale can be increased or decreased as a percentage from the original size. The TfrPreview component will also show scroll bars and a status bar with information regarding the current page number and total pages in the document.

To replace the standard preview window to the FastReport report generator, the corresponding TfrReport component's preview property should be changed to this window. You can find an example of a user preview window in the "Reports" subdirectory of FastReports "Examples" directory.

## Expanding the functions list

Earlier in this manual it was described how FastReport's functionality can be extended. This method is based on OnUserFunction event of TfrReport object. This method has one disadvantage: Each TfrReport object which uses a user function, needs to have an OnUserFunction event handler (i.e. user functions described this way are local to TfrReport object). Alternatively, you can organize a whole number of functions as a separate class (functions library).

FastReport supports this alternative approach, where you can write a class which derives from TfrFunctionLibrary. This class defines the basic set of properties and methods.

This is the way TfrFunctionLibrary base class is declared in FR\_Class module:

```
TfrFunctionLibrary = class(TObject)
public
  List: TStringList;
  constructor Create; virtual;
  destructor Destroy; override;
  function OnFunction(const FName: String; p1, p2, p3: Variant;
    var val: String): Boolean;
  procedure DoFunction(FNo: Integer; p1, p2, p3: Variant; var val: String);
    virtual; abstract;
  procedure AddFunctionDesc(FuncName, Category, Description: String);
end;
```

Following are the key methods:

- Create –the designer fills the List field with the list of functions desired, sorted alphabetically, in upper case;
- DoFunction – when called, this method has to return (return what?) to the computed function value in Val argument. Each function can have up to 3 arguments. These arguments' values are passed to DoFunction method as P1, P2 и P3 arguments;
- AddFunctionDesc – call this method to add the function description to the expression builder.

To add a list of additional functions to the “Insert Function” dialog box you will also have to call AddFunctionDesc procedure with the following arguments:

- Function name;
- This function category name;
- Text description of the function syntax and purpose. Note: «/» symbol in the function description is compulsory! It separates syntax description from the function itself.

Remember that each function has to be called separately by AddFunctionDesc.

Below is a simple example of a library with two functions:

```
type
  TMyFunctionLibrary = class(TfrFunctionLibrary)
  public
    constructor Create; override;
    procedure DoFunction(FNo: Integer; p1, p2, p3: Variant;
      var val: Variant); override;
  end;

constructor TMyFunctionLibrary.Create;
begin
  inherited Create;
  with List do
  begin
    Add('SPELLDATE');
    Add('SPELLSUM');
  end;
  AddFunctionDesc('SPELLSUM', 'My functions',
    'SPELLSUM (<Number>)/Returns value spelled out. ');
  AddFunctionDesc('SPELLDATE', 'My functions',
```

```

    'SPELLDATE(<Date>)/Returns value spelled out. ');
end;

procedure TMyFunctionLibrary.DoFunction(FNo: Integer; p1, p2, p3: Variant;
    var val: Variant);
begin
    val := 0;
    case FNo of
        0: val := My_DateConversion_Routine(frParser.Calc(p1));
        1: val := My_SumConversion_Routine(frParser.Calc(p1));
    end;
end;

```

To enable a prepared function library in FastReport, you have to register it. This means calling the `frRegisterFunctionLibrary` procedure and passing a reference to the class (not an instance of the class) to it as an argument. For example:

```
frRegisterFunctionLibrary(TMyFunctionLibrary);
```

To unregister the function library, call

```
frUnRegisterFunctionLibrary(TMyFunctionLibrary);
```

# Examples of reports

Examples of reports

## Examples of reports

FastReport is delivered with a number of examples which demonstrate different sorts of operation and different kinds of report composition. The example files are located in the DEMO subdirectory of FastReport home directory.

There are 9 projects included in the standard delivery which demonstrate the following possibilities:

- Insertion of graphs and diagrams in a report (CHART directory);
- Storing of a report template in Delphi form, instead of FRF file (DFMSTORE directory);
- Organizing the working environment for "end user" (USER directory);
- Usage of db-aware components (directory ENDUSER1);
- Controlling the logic of report composition using the OnManualBuild event (MANUAL directory);
- Manual report composition at runtime using code (RUNTIME directory);
- Printing of column reports with variable or unknown number of columns (PRNTBL1 directory);
- Printing of Column reports with variable width of the columns (PRNTBL2 directory);
- Construction of different reports using standard Delphi designer and development environment (REPORTS directory).

All the mentioned examples will be good manuals for those who would like to look at the majority of possibilities of the new report generator in short time, without re-reading the documentation.

Let's consider some of these examples in detail.

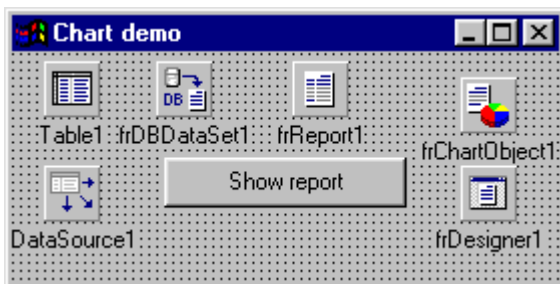


## Insertion of graphs and diagrams in the report

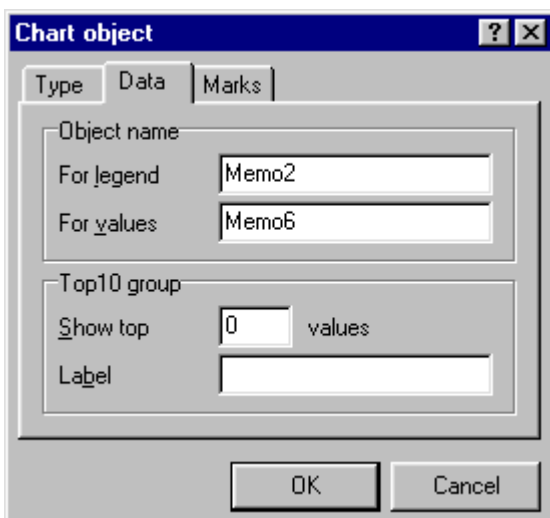
This example is located in the CHART subdirectory of FastReport examples directory. This project contains only one form which includes the following components:

1. A set of TTable/TDataSource components for extracting the report data from a database table (this example uses the COUNTRY.DB as data source which comes with the standard Delphi examples);
2. TfrDBDataSet components for binding the data source of a DB to the FastReport object;
3. TfrReport components – the report itself;
4. TfrChartObject components - "Graph" hook unit;
5. TfrDesigner components - run time report designer (end user report designer).

The form is represented in a figure.



The shown report consists of three pages with one graph on each page. The first one displays the contents of the AREA field from the COUNTRY table in vertical columns. The second graph represents similar data, but from POPULATION field. In order to let the two graphs show some data it is important to associate the "Diagram" object, which is placed on the report form, with the data that should be represented in it. This can be done with the editor of the "Diagram" component which is activated by a double click on the object. On the "Data" bookmark of this editor the fields «For the signature» and «For data» of the «Name of the object» group have to be filled in.

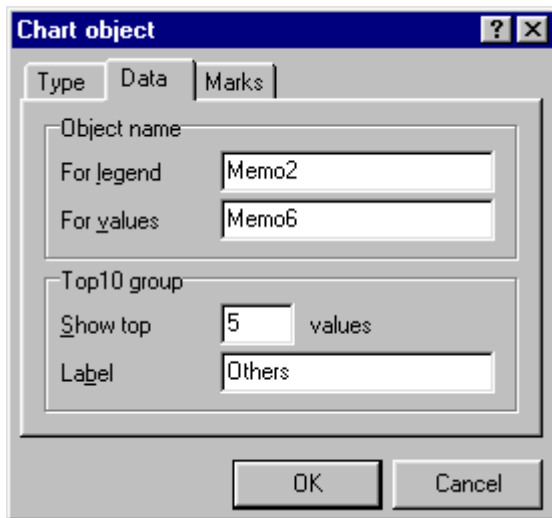


In this example these fields contain the values Memo2 and Memo6. Memo2 and Memo6 are the Names of the "Text" objects which are linked to the COUNTRY AREA and POPULATION Field of the table.

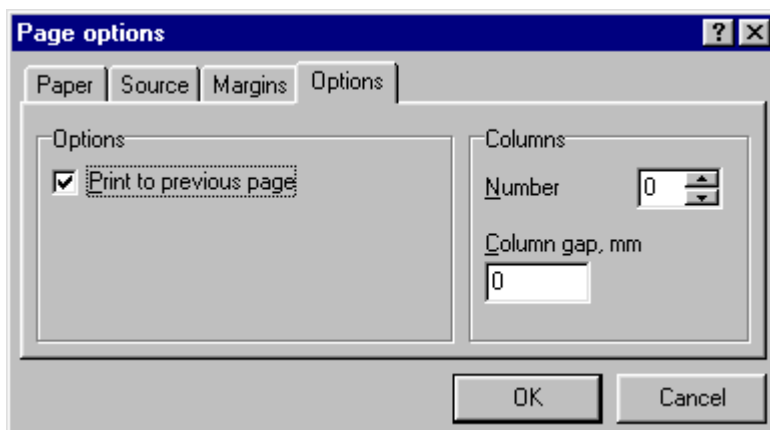
Thus, the data from the corresponding fields of all COUNTRY table records will be stored in the "Diagram" objects.

The third graph demonstrates the possibility of a so-called "TopX-graphs" composition, which only shows the X greatest values. The sum of the rest is represented in a separate column

named "Others". This can be done by setting a nonzero value to the «Show ... values» field on the "Data" bookmark of the component editor and setting the sum signature of the remaining values in the "Signature" field. The component properties editor dialog box for Top5 report is shown in a figure.



This report also shows that all pages of the report form are printed out closely to each other. This allows to fill a paper-list of the report more efficient. This feature can be activated by setting an appropriate flag in the page properties dialogue.



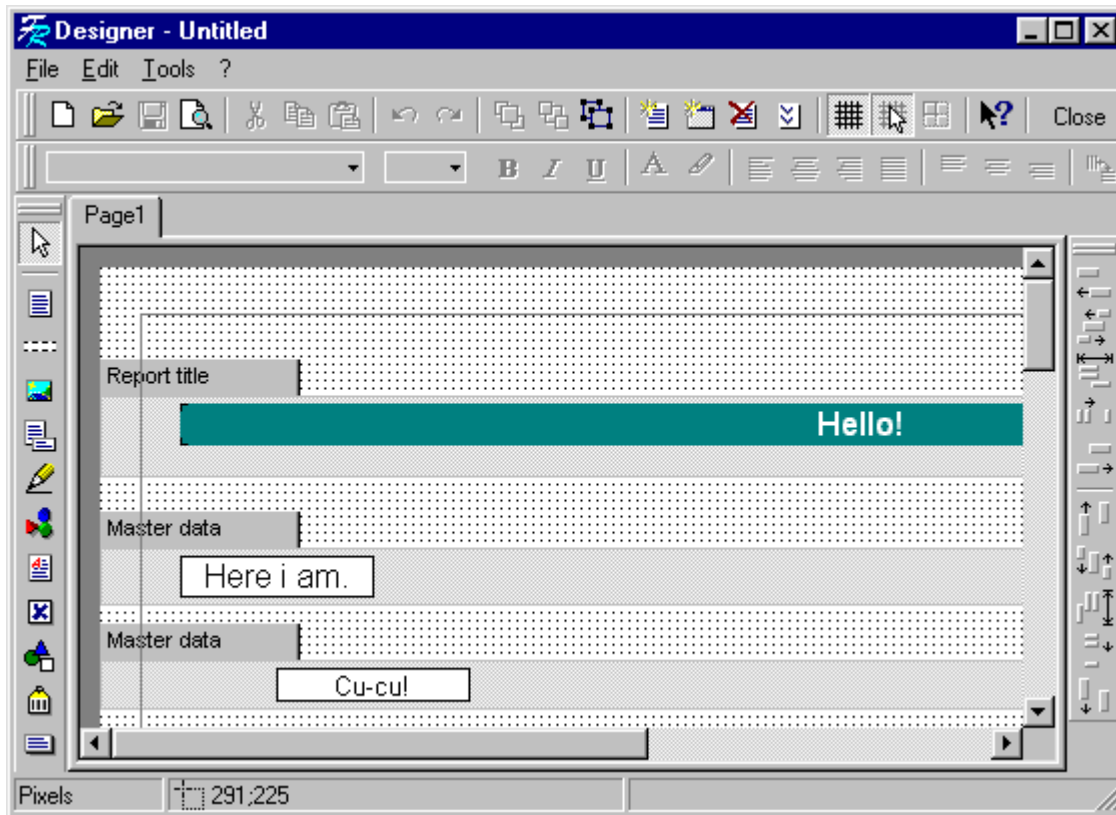
The preview window of the described report is represented below:



## Controlling the logic of report composition using the OnManualBuild event

The source code for this example is placed in the MANUAL subdirectory.

This example contains just one simple form with a button that starts the report preview. The form of the report in the FastReport designer looks like in the figure:



If you press the preview button in the designer now you will just see a “Hello!” line on a green background. This happens because the two sections of «Master data» are not associated with the data source and therefore they are not printed out.

However, if you execute this example, you will get the report with four pages on which the two sections of «Master data» are printed out in addition to the mentioned text line. This is achieved by the following OnManualBuild event handler of the FastReport object:

```
procedure TForm1.frReport1ManualBuild(Sender: TfrPage);
var
  i, j: Integer;
begin
  Sender.ShowBandByType(btReportTitle);
  for i := 0 to 3 do
  begin
    Sender.ShowBandByName('Band2');
    for j := 0 to 2 do
      Sender.ShowBandByName('Band3');
    if i <> 3 then
      Sender.NewPage;
  end;
end;
```

In this event handler the header section with all its fields is printed on the first page with *Sender.ShowBandByType(btReportTitle)*. Then the four pages are generated in a loop with the *Sender.NewPage* method and the section with the «Cu-cu!» text is printed out three times on each page.

This shows that with the OnManualBuild event handler the logic of the construction of the report can be controlled. That gives a greater flexibility to reports.

## Manual report composition at runtime using code

Sometimes the structure of a report is unknown at application designing stage or it can vary during runtime. In this case it is necessary to create the report template dynamically.

In such cases the report is not fixed and therefore can not be defined by the programmer as a separate file in contrast to a new form for a bookkeeping software when so called end-user reports can be used.

With FastReport it is possible to create a report form dynamically using your own program code just like the way it works, for example, with VCL objects.

This example is located in the RUNTIME subdirectory.

After pressing the button in this example a report that receives a list of companies from the CUSTOMER.DB table is created. The source code for the OnClick event handle of this button is shown below:

```
procedure TForm1.Button1Click(Sender: TObject);
var
  v: TfrView;
  b: TfrBandView;
  Page: TfrPage;
begin
  frReport1.Pages.Clear;
  frReport1.Pages.Add;           // create page
  Page := frReport1.Pages[0];

  b := TfrBandView(frCreateObject(gtBand, '')); // create MasterData band
  b.SetBounds(0, 20, 0, 20);
  b.BandType := btMasterData;
  b.Dataset := 'frDBDataSet1';
  Page.Objects.Add(b);

  v := frCreateObject(gtMemo, ''); // create data field
  v.SetBounds(20, 20, 200, 16);
  v.Memo.Add(' [Table1."Company"] ');
  Page.Objects.Add(v);

  frReport1.ShowReport;
end;
```

First this code deletes all pages available in the report form and then creates an empty one:

```
frReport1.Pages.Clear;
frReport1.Pages.Add;           // create page
```

Next a «Master data» section is created on this page and is associated with the data source:

```
Page := frReport1.Pages[0];

b := TfrBandView(frCreateObject(gtBand, '')); // create MasterData band
b.SetBounds(0, 20, 0, 20);
b.BandType := btMasterData;
b.Dataset := 'frDBDataSet1';
Page.Objects.Add(b);
```

The following step creates one “Text” object which is associated with the COMPANY field of the CUSTOMER.DB table:

```
v := frCreateObject(gtMemo, ''); // create data field
v.SetBounds(20, 20, 200, 16);
v.Memo.Add(' [Table1."Company"] ');
Page.Objects.Add(v);
```

At last the mentioned event handler shows the prepared report in a preview.

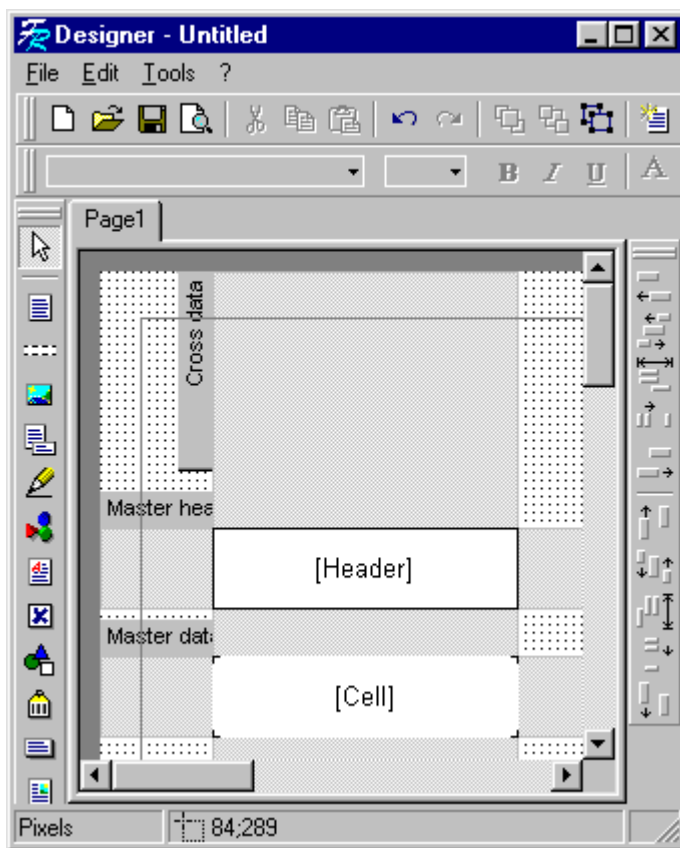
## Printing of column reports with variable or unknown number of columns

In practice it is frequently necessary to print out a column report with an unknown number of columns or with a number of columns that can be changed during program execution. This is needed, for example, at development of a database editor with the possibility of printing out the data of a table. In this case the report could be created by using program code but FastReport gives an easier solution for this problem.

So-called CrossTab-reports are used for that purpose. The distinctive feature of such a report is printing out the data in columns. It's not necessary to know the quantity of columns before. Especially for this type of reports CrossXXX sections are placed vertically, not horizontally. On these sections the "Text" objects are allocated on the intersections of the horizontal (for example «Master data») and the vertical (for example «Cross data») sections. In this case the report is created by the following principle: for all records in the data source of the horizontal section all appropriate records in the data source of the vertical section are searched and assigned.

So, during printing the contents of the table the data source for «Master data» contains the records for the table itself, and the list of fields of each single record is the data source for the «Cross data» section.

The simplest example of a CrossTab report can be found in the PRNTBL1 directory. The report form in this example looks like it is shown in the figure.



This figure shows that the "Text" objects are allocated on the intersections of the «Master header» and «Cross data» sections, respectively of the «Master data» and «Cross data» sections. If the report is constructed the records in the CUSTOMER.DB table are used as data source for the «Master data» section. The number of columns in the report is determined by two TfrUserDataset virtual sources which links the number of "records" with the number of fields of the CUSTOMER.DB table. The values of the fields are set in the OnGetValue event of the TfrReport object:

```
procedure TForm1.frReport1GetValue(const ParName: String; var ParValue: Variant);  
begin
```

```

if ParName = 'Cell' then
  ParValue := Table1.Fields[frUserDataset1.RecNo].Value;
if ParName = 'Header' then
  ParValue := Table1.Fields[frUserDataset2.RecNo].FieldName;
end;

```

If this example is executed, all records of CUSTOMER.DB table can be previewed. The following figure shows this preview:

CardNo	Company	6481	6482
1221	Kassat (Israel) Shoppers	61279 Sheperd Hwy	State 121
1221	Libanon	PO Box 2042	
1281	Right Drive	1 Maple Lane	
1284	Cayman Elms (Field Limited)	PO Box 541	
1288	Tom Ruyson Dining Center	6251 76th Pkwy/Sec	
1282	Blue Jack Super Center	25178 Fieldington Lane	State 212
1284	VIP Elms Club	22 Main St.	
1612	Queen Paradise	PO Box 8748	
1613	Paradise Aguilera	233 888 8706/27 A.L.	
1681	Marcel Elms Club	872 Queen St.	
1682	The Dapple Change	16241 Underwood Pkwy.	
1683	Blue Sports	203 126 Ave. Box 748	
1624	Mohel RCB&A Club	PO Box 9134	
1648	Kallos Club	PO Box 9874P	
1681	Jamaica RCB&A Center	PO Box 88	
1682	Island Fishers	8123 113 Elms Avenue	
1684	Adventure Undersea	PO Box 744	
2718	Blue Sports Club	82388 Ave Prince Street	
2728	Park's Elms Supply	1488 14th Ave St.	
2788	Dany Jones' Louche	2481 South 78th Place	
2783	RCB&A, Haines	PO Box 248274	
2788	Shangri-La Sports Center	PO Box 134888	
2778	Elms of Cardiff, Inc.	Marionel Place 84	
2384	Elms Winepharm	42 Aspen Lane	
2679	George Elms & Co.	873 King William Hwy	

City	State	Zip	Country
Kassat Israel	IS	94788/1234	LIB
Panquet			Rakomon
Kala Paphos			Cyprus
Reed Cayman			Malles (West Indies
Chathamland	St. Carlo	22832	LIB Virgin Islands
Stapulus	HI	86778	LIB
Chathamland	St. Carlo	22832	LIB Virgin Islands
KallianRena	HI	84768	LIB
Wagala			Colombia
Kilbarnet	Ontario	8384 2871	Canada
Marathon	FL	38823	LIB
Wideland	OR	87182	LIB
KallianRena	HI	84768	LIB
Flanada	FL	33374	LIB
Kingil	Jamaica		West Indies
St Elmsen Isle	RA	32821	LIB
Widene City			Widene
Largo	FL	34884	LIB
Wagman	OR	85427	LIB
Vancouver	BC	8877 8P1	Canada
Kilbarnet			Rakomon
Panquet			Rakomon
Ayles Mallesden	Carls		Rakomon
Hawden	SC	27278	LIB
Lugoff	NC	28578	LIB

## Column reports with variable width of the columns

In the previous examples it is supposed that all columns of the report has the identical width. However, in most cases it is necessary that the width of each column is determined by the data within it. So if, for example, there are two columns, one contains a line counter and the other presents data from a memo field, it is obvious that the first column should be much narrower than the one containing the memo field text.

The enhanced source code is placed in the PRNTBL2 subdirectory. This example is completely similar to the previous one except that the width of the columns in the report are set depending on the data within. The two new event handlers for the TfrReport in this example make it possible to change the width of the columns dynamically.

To define the width of a column based on its data the OnPrintColumn event handler of the TfrReport object is used. In this example the width of a column with text boxes is determined as a product of the size of the field and the width of the letter “W”; for “Date and time” fields the width of the column is set to a width of 15 “W” letters; the width of all other columns are set to 64 pixels.

The complete source code of the OnPrintColumn event handler is shown below:

```
procedure TForm1.frReport1PrintColumn(ColNo: Integer; var Width: Integer);
var
  Field: TField;
begin
  Field := Table1.Fields[ColNo - 1];
  if Field is TStringField then
    Width := Field.Size * Canvas.TextWidth('W')
  else if Field is TDateTimeField then
    Width := 15 * Canvas.TextWidth('W')
  else
    Width := 64;
  FWidth := Width;
end;
```

The OnBeforePrint event handler sets the width of a column to a value which is defined in the previous event handler:

```
procedure TForm1.frReport1EnterRect(Memo: TStringList; View: TfrView);
begin
  View.dx := FWidth;
end;
```

In this example a more effective filling of a page at report printing is achieved because every column has a width which is sufficient for the output of the data within. TfrPrintTable and TfrPrintGrid components which print TDBGrid table and component contents work just like this.