

AfalinaSoft XL Report

Version 4.1

This document is a guide to XL Report by Afalina Co., Ltd. – a tool for report generation and data analysis in Delphi applications through the use of Microsoft Excel.

Last revised 9-Jul-02

Copyright © 1998, 2002 Afalina Co., Ltd. All rights reserved.

For detailed license information, see LICENCE.TXT file in XL Report setup folder.

Table of contents

Table of contents.....	2
Why XL Report?.....	5
Examples - "QDemo.dpr"	5
DBDEMOS.....	6
XL Report and Microsoft Excel	6
XL Report Setup.....	6
Support	6
XL Report 4.....	7
What's new	7
Compatibility with XL Report G2	7

Developer's Guide.....9

Let's go	9
Quick Start	9
Creating the component dynamically	13
Time to look back - quick start	13
NoRange-datasets.....	15
Data transfer methods.....	15
Once more on conditional formatting	15
NoRange-datasets data in Excel formulas	16
NoRange-datasets and Memo-fields.....	17
NoRange-datasets - only necessary fields	18
Create XL Report component dynamically	19
Time to look back - NoRange-datasets	20
Range-datasets - data in ranges.....	21
List range	21
Range options and column options.....	23
Formatting an arbitrary range	24
Range-dataset data in Excel formulas	24
Simple range options.....	24

Time to look back - Range-datasets	25
Options in a list range.....	27
Sorting a list range	27
Totals in a column	27
Subtotals in a list range	28
Hiding detail data in an outline.....	29
Time to look back - list range options.....	29
Pivot tables.....	30
Your first pivot table.....	30
Effective cross-tables	31
Formatting a Pivot table.....	33
Static Pivot tables.....	33
Time to look back - pivot tables	35
Using VBA	36
Simple example.....	36
Excel charts	37
Passing parameters to VBA procedures	38
Time to look back - XL Report and VBA.....	38
Complex reports.....	39
Ranges' placement in XL Report	39
Nested ranges - designing master-detail reports.....	39
Totals in a column in master-detail reports.....	42
Sorting nested ranges	43
Multiple-sheet master-detail reports.....	44
Time to look back - complex reports	44
Additional features.....	45
Other options	45
Sheet options	45
Report options.....	45
Visualize report generation.....	46
OnBeforeBuild and OnAfterBuild events	46

TxlReport.Options - options of a report	47
TxlDataSource.Options	48
Add a report to an existing workbook	48
Several reports in one workbook	49
Custom options - the XLOptionPack technology	49
Time to look back - other options	49
Unbound data transfer	50
Report parameters	50
Unbound data in ranges	50
XL Report ProOptionPack™	52
What is it?	52
QDemoPro.dpr	53
Using ProOptionPack™	53
Grouping without subtotals	54
Advanced subtotals	58
Static charts	61
Microsoft Excel - serious things	64
End	64
Appendices	65
Appendix A: Guidelines for creating a list on a worksheet from Microsoft	65
Appendix B: Guidelines for naming cells, formulas, and constants in Microsoft Excel	66
Appendix C: Standard options	67
Appendix D: ProOptionPack options	70

Why XL Report?

XL Report is a set of Delphi-components for report generation in Microsoft Excel using Automation (formerly known as OLE Automation). Although Automation can be slow a process, XL Report overcomes this problem by using highly optimized algorithms so that it is now easy to transfer large volumes of data to Excel workbooks. These algorithms use only native data types and early binding to the Excel Type Library. *Variant* data type variables are used sparingly and mostly to work around some bugs in the Excel Type Library.

With XL Report, Microsoft Excel can be an integral part of your application. As a report building and data analysis tool, Excel is an excellent alternative to common report generators, and using Excel's built-in features can make your reports much more responsive.

Use XL Report as a tool for data transfer to Excel. Then use Excel visual instruments: formatting (including conditional formatting), AutoFilter, Pivot tables, charts, VBA, ActiveX controls, Web-publishing tools (Excel 2000 only) to construct a versatile data analysis system. With XL Report, you can move a lot of report programming and tuning into Excel. XL Report templates are simple and our algorithms are fast – we carefully count every millisecond – so you waste less time on routine report programming and get surprisingly fast results. If you want to master such a versatile tool as Excel – XL Report is an excellent choice.

Furthermore, XL Report doesn't operate with the usual concepts of band-oriented report tools: *Footer*, *Header*, and *Detail*. So you get a much greater degree of freedom in report construction and design, and the easiest possible integration of Borland Delphi and Microsoft Excel.

And finally. XL Report works with *TDataSet* descendants. It means that XL Report works with BDE, ADO, ODBC, and IB Express.

Examples - "QDemo.dpr"

To aid you in learning how to use XL Report, several examples are included in the installation package. All the examples that you will see in this **Developer's Guide** are collected in a demo project, that can be found in the */QDemo* subfolder of the XL Report installation folder. This project includes forms corresponding to chapters of the **Developer's Guide**.

Each form is well documented, including form details, form class name, unit name and filename of the template workbook used. Each example in the document references a specific form. XL Report uses several Excel workbooks as templates in report building. These files reside in the */QDemo/Templates* folder. All of them were created in Microsoft Excel 2000 but you can use Excel 97 also if MS Office Service Release 2 is installed. If you use Excel 2002 we recommend you to check the *Trust access to Visual Basic Project* checkbox in *Tools|Macro|Security|Trusted Sources*. Some examples contain macros written in VBA. It is beyond the scope of this manual to give a full VBA tutorial. However, you can use the Excel Help files and MSDN (see also <http://www.afalinasoft.com/links.html>) to gain a fuller understanding of the VBA code you will see in the XL Report examples.

DBDEMOS

QDemo uses the DBDEMOS tables from the Delphi installation CD. In order to use **QDemo**, the DBDEMOS must have been previously installed. There was no intention to make a finished application with a logical DB structure. The QDEMO project was intended just to show you XL Report capabilities.

XL Report and Microsoft Excel

XL Report builds reports in MS Excel. Experienced Excel users will find the methods and procedures of XL Report very familiar. XL Report works with MS Excel 97 SR2, MS Excel 2000, and MS Excel 2002. All features of XL Report have been tested on each version.

Note

In all versions of Excel, you should open a template workbook, open VB Editor and clear the "Notify before state loss" flag in *Tools|Options|General*.

To work with Excel 2002 you should check the "*Trust access to Visual Basic Project*" checkbox in *Tools|Macro|Security|Trusted Sources*.

XL Report Setup

The XL Report Setup procedure description can be found in **readme.txt** in the XL Report installation folder.

Support

Please email any questions or comments to support@xl-report.com. But first, always check <http://www.afalinasoft.com/support/> where you will find the latest news, FAQ, and tips and tricks sections. To get our latest news by email, click [here](#) or subscribe directly at <http://www.afalinasoft.com/>. You can find the latest release of the software at <http://www.afalinasoft.com/downloads.html>.

The most current version of this document is available at <http://www.afalinasoft.com/xl-report/docs/>.

XL Report 4

What's new

Reworked software. We revised the design strategy of XL Report. It still is a single component on the component palette (see "Compatibility with XL Report G2", though) that allows you to define report data and properties in a fast and convenient way. But from within, it became a more deep hierarchy of classes allowing creation of several levels of abstraction and solving the task of extension of product's features effectively.

XLOptionPack technology. XL Report core is built on this technology, which bases on two classes – *TxlOption* and *TxlOptionPackage*. The first one allows describing an option and process it or overload an existing option, extend its features and maybe change its behavior completely. The second one (it descends from *TComponent*) allows creation of option packages and their distribution independently of XL Report. Having been placed on any form of an application such a package will register its options in XL Report core automatically, allowing in such a way to use the options from the option package along with standard options. In the context of this technology, all options are separate classes now. These classes are gathered into the standard option package, which is installed automatically when you install XL Report.

Unlimited Master-detail. XL Report can process nested ranges in order to reflect one-to-many relation between datasets. You create such a template and set the *MasterSourceName* property of the *DataSources* collection items. The number of nesting levels is not limited; any range can contain an unlimited number of nested ranges. Design restrictions are minimal.

Composite reports. You can make use of effective built-in feature of merging several reports in one report.

Open source code. In order to allow better understanding of XL Report principles, we have opened source code of base classes and standard options. You can use them as samples of extending XL Report core features.

Compatibility with XL Report G2

XL Report contains the only component class. Its name in version 4 is *TxlReport*. For compatibility on the source code level the *TxlReportG2* component is installed, which is ported to XL the new XL Report core. For all of your new projects we recommend to use the *TxlReport* component only. This guide describes only this component. The “**XL Report G2 Developer’s Guide**” devoted to the *TxlReportG2* component class is available on the AfalinaSoft site.

XL Report 4 doesn't provide support for the following features of XL Report G2:

- **MultiSheet** option has been replaced with *TxlReport.MultisheetAlias* and *TxlReport.MultisheetField* properties.
- **TargetBook** option is replaced with the *TxlReport.ReportTo* method, which accept the path and file name of the target workbook.
- **OnGetDataInfo** and **OnGetFieldValue** events are excluded. We offer several new properties and events that allow flexible defining your data. The newly developed mechanism operating with *non-TDataSet* data is oriented on large data volumes in the first place.

- **ExtXXXOptions** properties and corresponding **OnXXXOption** events are excluded because of introducing XLOptionPack technology.
- **TxlReport.ClearDataSource** method is not supported any more. The replacement is the **TxlReport.DataSources.Clear** method.
- **TxlReport.ReportToOLEContainer** method is excluded.

Developer's Guide

Let's go

XL Report components allow you to create a variety of reports using one or more *TDataSet* instances, or its descendants. XL Report is based on the *TxlReport* component. This class contains the properties and methods required to access an Excel workbook and to transfer data to it. To get a report you must create a special Excel workbook that will serve as a template and will contain appropriate formatting as well as description of data transfer, placement and processing.. XL Report will find the template, create a new Excel workbook based on this template, fill report cells with the data of the datasets specified by you, and then open the Excel window. XL Report's properties and methods allow you to create reports simply and easily.

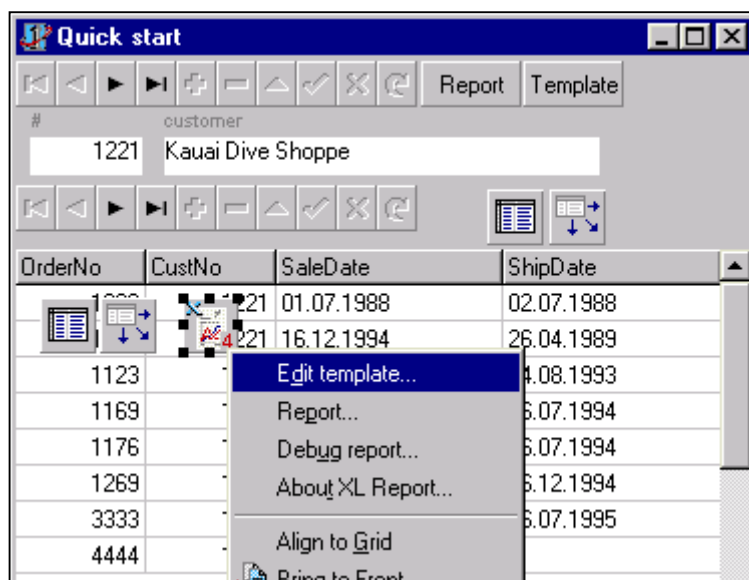
Quick Start

Where? *QDemo.dpr* form: *frmQuickStart*; unit: *tQuickStart*; template: *tQuickStart.xls*

For this example, we have included the *frmQuickStart* form in the **QDemo.dpr** project. We have also inserted two instances of *TTable* (*tblCustomers* and *tblOrders*) and two instances of *TDataSource* (*dsrcustomers* and *dsrcOrders*) onto the form and bound them to *Customer.db* and *Orders.db* tables. Then we linked these tables (one-to-many relation) by setting the appropriate values to the *MasterSource* and *MasterFields* properties of *tblOrders*, indexing on the *CustNo* field. To view the data we have added an instance of *TDBNavigator* and several *TDBEdit* edit fields for the *Customer.db* table as well as an instance of *TDBNavigator* and the *TDBGrid* for the *Orders.db* table. To build the report and to edit its template, two command buttons (*TSpeedButton*) with *Report* and *Template* captions (*Caption* property) were placed on the form and both of the *TTables* were made active.

XL Report adds two nonvisual components, *TxlReport* and *TxlReportG2* onto the Delphi component palette. We built all examples on *TxlReport* because *TxlReportG2* is left for compatibility only and isn't recommended for use in your new projects. So an instance of the *TxlReport* component has been added to the form, which is named *xlReport*.

As stated earlier, to create a report you must first create a report template. To create the template for this example, double-click on *xlReport* or right-click on it and choose *Edit template*. Because Microsoft Excel is a component editor for *TxlReport* it will start with an empty workbook opened.



Note

The correct setting of the *XLSTemplate* property ensures the right workbook opening. If it is empty then XL Report launches Excel and opens a new empty workbook. If you omit the path to the template then at design-time XL Report will try to open the template in the same folder where **.dpr** file resides. At run-time XL Report will look for the template in the application start-up folder. You are also allowed to use a relative path name from the current directory, e.g. "..\Templates" or ".\Templates".

The first thing you will notice is that there are a lot of **#NAME?** error messages (if there isn't enough place for this text Excel replaces it with # signs). This is normal. Pick any such cell and look at the *Formula bar*. Almost all the formulas include the pattern **=Alias_FieldName** – we call such a pattern a **field formula** (the underscore is required) – where **Alias** is equal to **tblCustomers** or **tblOrders** and **FieldName** – one of field names of the dataset. Since Excel cannot find these cells in the workbook immediately it displays **#NAME?**.

You can apply any formatting to any workbook cells, insert pictures, and modify any of the parameters of the workbook itself. In this example, we have turned off the zero values display and hidden the gridlines (*Tools|Options*). XL Report will preserve all changes to the template.

To produce a report of all orders for the current customer we created a specially formatted set of cells – **B11:L15**. Looking at the design of the report you can see that certain cells have been merged and some columns have been hidden. The hidden cells **D15** and **E15** contain the formulas– **=tblOrders_ShipToAddr1** and **=tblOrders_ShipToAddr2** respectively. The values of these cells are used in the formula of cell **G15** – **=""&D15&" "&E15** – to create the full shipping address. As a reminder - it is permissible to use all of the standard Excel formulas as well as functions, references, etc. This report includes a *Special tax* that is less than the “standard” *Tax Rate* by the value in cell **L4**. The formula for this *Special tax* rate – **=I13-L\$4** – can be seen in cell **J14** (Note the use of **\$** character in referencing the **L4** cell – this is an example of an absolute reference. See Excel Help for details).

We froze the top horizontal pane just under the report head (*Window|Freeze Pane*). Then we added colors to the report. To distinguish the orders honored by Visa cards, we applied conditional formatting to the **G13** cell that contains the payment method – **=G13="Visa"**. To see this, select this cell and choose *Format|Conditional formatting*. So, for Visa orders, the field value will be displayed in a different color. The same trick was used for the *Special tax* cell. As the result, all negative values in this cell are shown in red.

As the final step in our work on the template, the **A13:L16** range was named *OrdersRange*. You can find it in the *Name box* at the left of the *Formula bar* (to create the name we selected the cells and printed the name in the *Name box*). Select this name from the *Name box* and you can see this range selected on the worksheet. Note the empty column at range's left and the empty row below. These cells are used in XL Report to describe additional actions that used to manipulate the data. They have been left empty in this example.

The screenshot shows an Excel worksheet with a report template. The 'OrdersRange' is selected, covering cells A13:L16. The report includes fields for Customer, Address, City, State, Country, Zip, Order No, Sale date, Ship date, Payment method, Items total, Tax rate, Amount paid, Phone, Fax, and Contact. The 'special tax' field is highlighted in red.

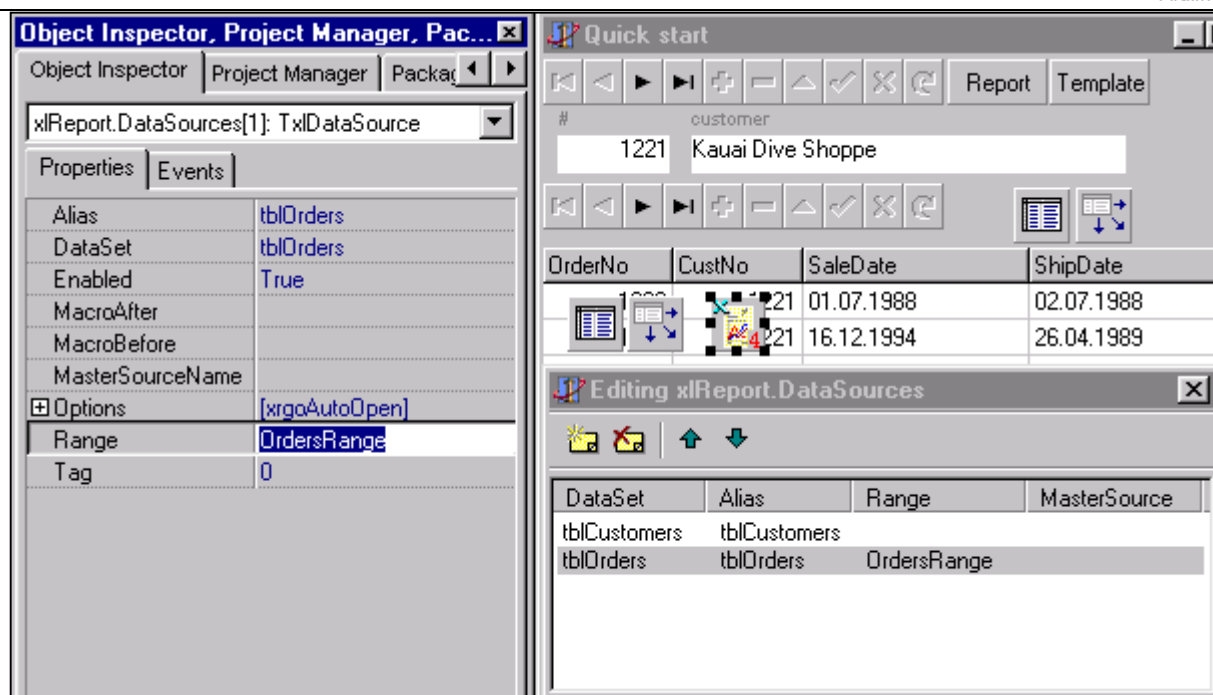
Order No	Sale date	Ship date	Payment method	Items total	Tax rate	Amount paid
#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	special tax	#NAME?
#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?
#NAME?	ship address	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?

Why A13:L16?

An Active XL Report range consists of three regions: the first one describes the output format for a single record of a dataset bound to the range, the second one is an additional row just below the first region (option row), and the third is an additional column at the left of the first region (option column). While processing the dataset, Active XL Report multiplies the first region as many times as needed to output the dataset. This description implies that header row(s) containing column labels must not be included in the range. Nevertheless, the header row must be placed above some of Active XL Report ranges. See the "List range" chapter below.

The workbook has been saved in the project folder. You can choose any other folder. We named the workbook **tQuickStart.xls**. You can choose any other name. Then we closed Excel and returned to our Delphi form.

There must be a link between the *TxlReport* component and the report template. To create the link we set *XLSTemplate* property of *TxlReport* to the path and file name of the template. If you set it to the file name only, the program will attempt to locate the template according to the following rule – at design time, the template must reside in the project folder and at run time it must reside in the app start folder.



To finish the design of the report the *TxlReport* component must be bound to datasets by using the *DataSources* property of the *TxlReport*. This property is a collection, each item of which represents a dataset. Two items have been added to the collection and their *DataSet* properties set respectively to **tblCustomers** and **tblOrders**. For the last one the *Range* property has also been changed by setting it to **OrdersRange** – the name of the range for the orders of the current customer. It is worth of mentioning that the *Alias* property takes the value of the *Name* property of a given Dataset automatically.

If you follow this process yourself, then clicking on *Report* in context menu of *TxlReport* will produce the report. To create a report at run-time add a line of code to the *Click* event of the *Report* button:

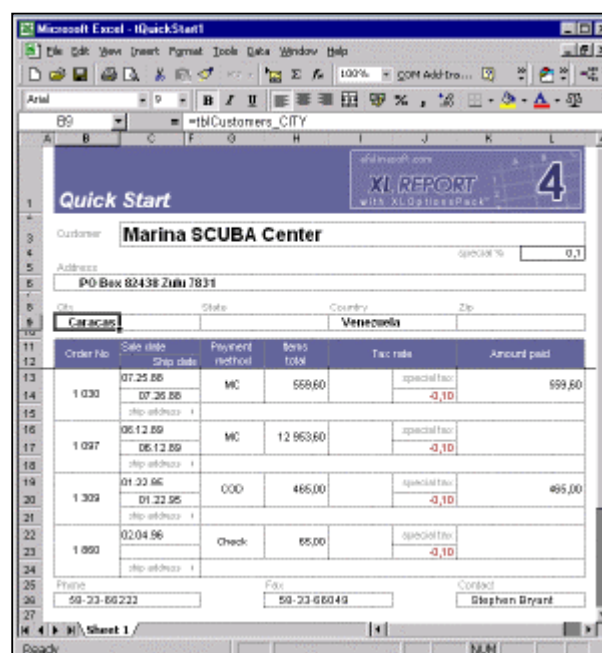
```
procedure
TfrmQuickStart.btnReportClick(Sender:
TObject);
begin
    xlReport.Report;
end;
```

To allow your users edit the template at run-time we simply add the following line to the *Click* event of the *Template* button:

```
procedure
TfrmQuickStart.btnTemplateClick(Sender:
TObject);
begin
    xlReport.Edit;
end;
```

Test the app by building reports for several customers.

Note the report workbook name – it was created by adding serial numbers to the name of a report template.



Note

You can use the *Options* property to change the default behavior of XL Report. If Excel has been launched at the time you run the report building procedures, XL Report will use this instance by default. If you include *roNewInstance* in *Options*, XL

Report will use the new Excel instance instead. If you exclude *roOptimizeLaunch*, you make XL Report release all Excel interfaces immediately after the report building. Here "To release the interfaces" does not mean "to unload the Excel process". The user must close Excel but *roOptimizeLaunch* affects the Excel process unloading. There is a feature in XL Report that forces the Excel interface that is releasing to ignore *roOptimizeLaunch*. In this case use the *TxlReport.ReleaseExcelApplication* class method that resides in the *xlReport* unit.

Creating the component dynamically

Where? *QDemo.dpr* form: *frmQuickStart2*; unit: *fQuickStart2*; template: *tQuickStart.xls*

To demonstrate the use of dynamically created *TxlReport* instances we took the previous form and deleted the *TxlReport*. Then we renamed the *Template* button in *Report2* and rewrote *Click* events of both buttons. *Report2* button as follows:

```
procedure TfrmQuickStart.btnReport1Click(Sender: TObject);
var xlReport: TxlReport;
    Dsr: TxlDataSource;
begin
    xlReport:= TxlReport.Create(Self);
    xlReport.XLSTemplate := 'tQuickStart.xls';
    try
        Dsr := xlReport.DataSources.Add;
        Dsr.DataSet := tblCustomers;
        Dsr := xlReport.DataSources.Add;
        Dsr.DataSet := tblOrders;
        Dsr.Range := 'OrdersRange';
        xlReport.Report;
    finally
        xlReport.Free;
    end;
end;
```

The two first lines of the code create an instance of *TxlReport* and bind it to the template. Then the new item is added to the *DataSources* collection and bound to *tblCustomers* dataset. Another item representing *tblOrders* is processed the same way. Data output of this item is aimed at the *OrdersRange*. Finally, the *Report* method builds a report.

This is the standard method of dynamic component creation. To shorten the code even more the XL Report team has developed several useful methods. Look at the code of the second button:

```
procedure TfrmQuickStart.btnReport2Click(Sender: TObject);
var xlReport: TxlReport;
begin
    xlReport:= TxlReport.CreateEx(Self, 'tQuickStart.xls');
    try
        xlReport.AddDataSet(tblCustomers);
        xlReport.AddDataSet(tblOrders, 'OrdersRange');
        xlReport.Report;
    finally
        xlReport.Free;
    end;
end;
```

It's simple, isn't it?

Time to look back - quick start

To summarize the topics covered in this section:

- XL Report adds two non-visual components to the Delphi Component palette – *TxlReport* and *TxlReportG2*. The last one is left for compatibility only and not recommended for use in new projects.
- Creating a report means mostly “to create a template in MS Excel”.
 - Template editing and report building options are available at design time in the component's context menu.
- Special **field formulas** define data transfer.
- To transfer **one** record (namely the current one) of a dataset in a report, you have to add an item in the *DataSources* collection of *TxlReport* and bind it to the dataset.
- To transfer **all** records of a dataset in a report, you have to add an item in the *DataSources* collection of *TxlReport* and bind it simultaneously to the dataset and to the named range of the special format in the report template.
- XL Report preserves all the template cell formats in a report as well as the parameters of the worksheets and the workbook.
- To build a report, you have to place the *TxlReport* component onto a form, set the properties needed, and call the *Report* method.
- To edit a template at run time, you have to place the *TxlReport* component onto the Delphi form, set the properties needed, and call the *Edit* method.

NoRange-datasets

We distinguish two dataset types according to the *Range* property value of the corresponding *DataSources* item. If this property is empty then the dataset is a **NoRange-dataset**. Such a dataset type is used to transfer **one** record – namely, the current one – to a report. In the previous example the *tblCustomers* dataset is a NoRange-dataset. You can use NoRange-dataset field formulas in any cell of any sheet of the template workbook and Excel will find their values at run-time. How? XL Report adds a hidden worksheet in a report workbook and transfers values of **all** fields for the current record. Then it names all these data cells. The names are given in accordance to the **Alias_FieldName** pattern (the XL Report term for this is **field formula**). This allows the NoRange-datasets to work correctly. Currently supported field types are:

- String – ftString, ftFixedChar, ftWideString, ftMemo, ftFmtMemo (without formatting);
- Integer – ftSmallint, ftWord, ftInteger, ftLargeint, ftAutoInc;
- Real – ftFloat, ftCurrency;
- Date/time – ftDate, ftTime, ftDateTime;
- Logical – ftBoolean.

For other field types, the **Null** value is transferred.

Note

You can access the hidden worksheet mentioned above via setting the *TxlReport.Debug* property to True and selecting the Format|Sheet|Unhide menu in a report workbook.

Data transfer methods

XL Report provides three methods of data transfer: variant array, comma-separated values (CSV) via the clipboard, and Fast DDE Table Format using DDE calls. Your choice must be reflected in the value of the *DataExportMode* property of the *TxlReport* instance but you should be aware of restrictions of data transfer methods:

- **xdmVariant**. Data portions are transferred in a variant array with direct assignng of its values to cells. Usually this method is the slowest one and used by XL Report core in some situations. Because of an error in Excel algorithms in all Excel versions, strings longer than 1500 characters are truncated. There also are some problems with data type recognition, e.g. the "00033" string can be recognized as a number. This method is left for compatibility with previous versions of XL Report.
- **xdmCSV**. Data portions are transferred through the Clipboard in CSV format. Middle speed of data transfer. There are some problems with data type recognition, e.g. the "3/7" string can be replaced with either result of division or with a date. Use this method with long strings (the limit is 8 Kb).
- **xdmDDE**. The default method. Data portions are transferred using Fast DDE Table Format. The fastest method because of use of Excel's native data types. Data type recognition problems were not observed. Strings longer than 255 characters are truncated.

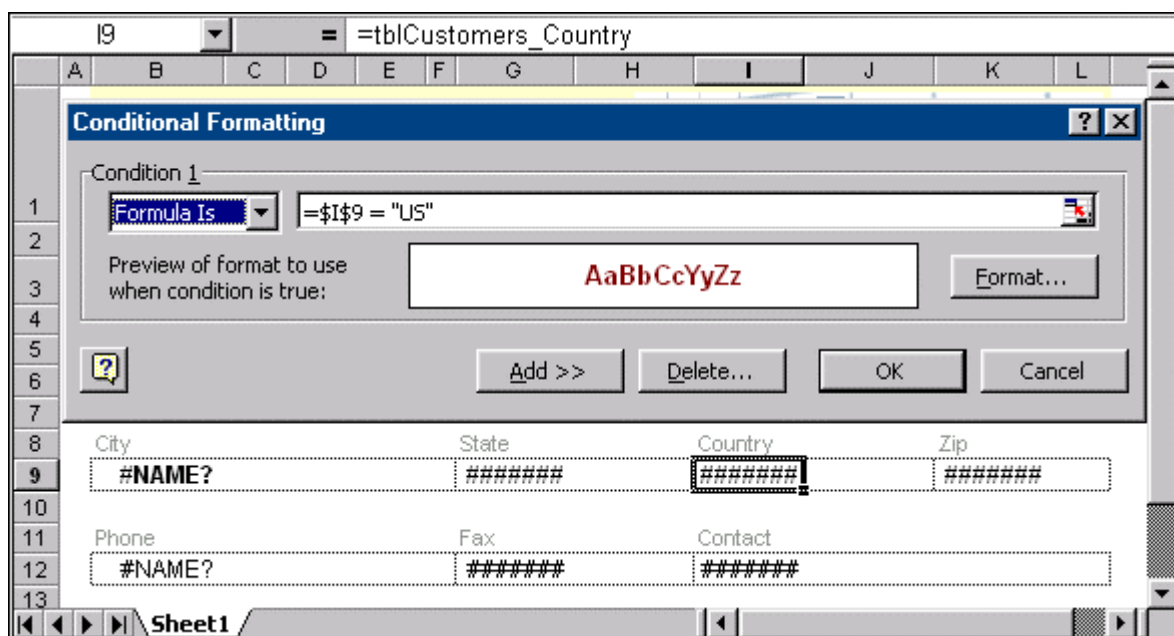
Once more on conditional formatting

Where? *QDemo.dpr* form: *frmNoRange1*; unit: *fNoRange1*; template: *tNoRange1.xls*

For this example the *frmNoRange1* form has been added to *QDemo.dpr*. Just one dataset – *TTable* – has been added referencing the *Customer.db* table, and it is named *tblCustomers*. To show the current record of this

dataset, an instance of *TDataSource* has been added, along with several labels, edit fields, and navigator. They are bound to *CustNo*, *Company*, *State*, and *Country* fields. A *TxlReport* component was added, and an item was added in its *DataSources* collection and bound to *tblCustomers* dataset. Two buttons were also added. Their names – *Report* and *Template* – reflect the method of *TxlReport* they call. The *TxlReport.XLSTemplate* property was set to the name of the template – **tNoRange1.xls**.

The previous report template was saved as a **tNoRange1.xls** workbook in the project folder. Every sign of *tblOrders* was erased and the *tblCustomers_FieldName* cells were left undisturbed. In this example we wanted to single out the *Country* cells when the *Coutry* field would be equal to **US**. To achieve this we used conditional formatting with the **=*I\$9*="US"** formula and chose to highlight the cells in red. Save the template and quit Excel. Then start the project and build the report for several customers including those that have **US** in *Country* field. You will see the difference. You can replace the reference of the **I9** cell in the conditional formatting formula with the direct reference to the *Country* field – *tblCustomers_Country*. This is the preferable approach because of the possibility that the end user might change the meaning of the **I9** cell.



NoRange-datasets data in Excel formulas

Where? QDemo.dpr form: *frmNoRange2*; unit: *fNoRange2*; template: *tNoRange2.xls*

You have seen how XL Report adds a hidden worksheet to a template workbook and transfers the values of all the fields of supported types. Then data cells are named according to the *Alias_FieldName* pattern (**field formula**). In a generated report, you can check this fact by just using a field formula for any of your field names in any cell of the workbook.

Field formulas (in fact they are just cell names) can be used in any Excel formulas the same way as any other Excel names. You can also use them in VBA. In the example we invented a fictitious term *Next Sale* and decided that it must be equal to last sale date from *tblCustomers_LastInvoiceDate* plus some additional days. We also determined that the additional days should be equal to *tblCustomers_CustNo* divided by 25. So the formula would appear as `=tblCustomers_LastInvoiceDate+tblCustomers_CustNo/25`. Another template was created, using the previous template as a sample, and it was saved under a new name **tNoRange2.xls**. Your instance of *TxlReport* must be informed of this change – **TxlReport.XLSTemplate="tNoRange2.xls"**. Put this formula in an arbitrary cell of the template. To control the accuracy of computations put the field formula for the *LastInvoiceDate* field in another cell. Then you can test the result – just click *Report* in the context menu of the *TxlReport* instance to see the results.

Note

We have shown here the use of conditional formatting with a formula using the data of our application. Some users of XL Report use conditional formatting to hide some values in a report by defining conditions that set a cell's background color equal to font color.

NoRange-datasets and Memo-fields

Where? `QDemo.dpr` form: `frmNoRange3`; unit: `fNoRange3`; template: `tNoRange3.xls`

You can use Memo-fields as any other fields but working with them you should be aware of data transfer method used.

Another form was added to our project and *TTable* instance was placed onto it. This *TTable* instance is named *tblCustoly* after *Custoly.db* from DBDEMOS. To display the current record of this dataset we added to the form: *TDataSource* instance, some labels, some edit fields (*CustNo*, *First Name*, *Last Name*, and *Phone*), and navigator. Then the instance of *TDBMemo* was added and bound to *Remarks* field. Finally, the *TxlReport*

component was placed onto the form, and an item (of *TxlDataSource* type) added to its *DataSources*, and we bound the item to *tblCustoly*. Then the two standard buttons *Report* and *Template* that call corresponding *TxlReport* methods were placed on the form. The final stroke – ***TxlReport.XLSTemplate = “tNoRange3.xls”***.

Field formulas refer to the above-mentioned fields and the name of the template workbook is **tNoRange3.xls**. To output the *Remarks* field we merged several cells and specified the appropriate field formula. Now try to build the report. If the string length for the *Remarks* field is greater than 255 then the string is trimmed because of the default data transfer method – Fast DDE Table Format. Any other transfer method will produce the full string.

NoRange-datasets - only necessary fields

Where? QDemo.dpr form: *frmNoRange4*; unit: *fNoRange4*; template: *tNoRange4.xls*

If you plan to use only some fields of a NoRange-dataset, why you should have to transfer all the fields when you need only some of them? This is the default behavior and is correct as far as it goes. Do you understand this? To allow you to transfer only some of the fields there were only two possibilities – either to scan all the cells in a template for field formulas or to allow you to define the fields. We chose the last approach because the first one leads to a time wasting operation.

You can specify all the fields needed in any free cell of the template (**tNoRange4.xls**) by separating them with a semicolon. In order for XL Report to find the cell you must give it the same name as the dataset.

During report generation, XL Report will find this cell, recognize the fields, transfer their values, and clear the cell. We named the cell *tblCustoly* and set its value to **Remarks;First_Name;Last_Name** in the above example.

tblCustoly		= Remarks;First_Name;Last_Name	
1	Only necessary fields		
2	First name		
3	#####	#NAME?	
4	Last name		
5	#####		
6			
7	Address		
8	#NAME?		
9			
10	City	Country	Post code
11	#NAME?	#NAME?	#NAME?
12			
13	Phone	Fax	VIP status
14	#NAME?	#NAME?	#NAME?
15			
16	Remarks;First_Name;Last_Name		
17			

Create XL Report component dynamically

We showed you how to create the component dynamically in **Quick Start**. Here we extend and improve on that example. The code that builds the report on a single NoRange-dataset is as follows:

```
procedure TfrmQuickStart.btnReport2Click(Sender: TObject);
var xlReport: TxlReport;
begin
  xlReport:= TxlReport.CreateEx(Self, 'tQuickStart.xls');
  try
    xlReport.AddDataSet(tblCustomers);
    xlReport.Report;
  finally
    xlReport.Free;
  end;
end;
```

Note that the *AddDataSet* call adds a NoRange-dataset to the *DataSources* collection of the *Report* component.

But there is an even better technique – using the global *TxlReport* instance. The sequence is as follows:

- Create an instance of *TxlReport* in the **initialization** section of an arbitrary unit;
- Destroy the instance in the **finalization** section of the same unit;
- Before every call of the *Report* method clear the *DataSources* collection and only then add the needed datasets.

Example:

```
unit Reports;
...
var xlReport: TxlReport;

implementation

procedure TfrmQuickStart.btnReport2Click(Sender: TObject);
begin
  xlReport.DataSources.Clear;
```

```

Try
    xlReport.XLSTemplate := 'tQuickStart.xls';
    xlReport.AddDataSet(tblCustomers);
    xlReport.Report;
finally
    xlReport.DataSources.Clear;
end;
end;
...
initialization
    xlReport:= TxlReport.CreateEx(Self, 'tQuickStart.xls');
...
finalization
    xlReport.Free;
...
end.

```

Time to look back - NoRange-datasets

We have focused on these themes:

- XL Report operates on two types of datasets recognized by the empty-nonempty state of the *Range* property of *DataSources* items.
- If this property is empty then the dataset is a **NoRange-dataset**.
- XL Report creates a hidden worksheet to keep data for all NoRange-datasets.
- XL Report transfers only string, integer, real and logical values.
- All other field types produce **Null** (the Excel interpretation of **Null**) values in a report.
- Field formulas for NoRange-datasets can be used in any cell of any worksheet of a template workbook.
- Arbitrary formatting (including conditional formatting) can be applied to cells containing field formulas.
- Field formulas can be used in standard Excel formulas as well as in VBA programs.
- There are three data transfer methods: **xdmVariant**, **xdmCSV**, and **xdmDDE**.
- There is a way to limit the number of fields being transferred.
- If you plan to use fields longer than 255 characters, you will need to use the **xdmCSV** data transfer method.

Range-datasets - data in ranges

We have addressed how XL Report handles datasets in the previous pages and introduced the term – **NoRange-dataset**. This term is derived from the empty value of the *Range* property of a *DataSources* item. Such datasets are used when you need to report the data from one and only one record – namely the current record.

But what if you need all of the records? In that case, fill the *Range* property with the name of a range in a template workbook and it becomes a **Range-dataset**. It's that easy. These datasets are used when you want to output **all** records of a dataset. The records are placed in the range supplied by you. Thus XL Report takes great advantage of the *Range* object, a basic object of Microsoft Excel used in a large number of Excel Type Library methods and in VBA as well. XL Report further differentiates between two range types. The first one, a **list range**, is a range where each row represents the data of only one record. Such a range must have a header containing one row (note: the header isn't included in the named range but is an essential part of a list range). To put it another way – a list range must suit the definition of a table in a relational database, which is referred to as a "list" in Excel terminology. See [Appendix A](#) or Excel Help for Microsoft's recommendations on creating a list. The second range type is the **arbitrary range** where every kind of formatting is allowed. In these ranges you can merge cells, skip rows and columns, omit the header, and so on. The *OrdersRange* range in **Quick Start** is an example of such an arbitrary range.

In summary, an XL Report range definition meets the following standards:

- it has a name.
- it is a continuous rectangle.
- its cells can contain any allowable value (and can therefore be empty, softening the requirement for contiguity).
- **it must contain at least two rows and two columns.**

Note: The leftmost column and bottom row (**option column** and **option row**) of a range are treated in a special way. Having built a report, XL Report clears the option column values and destroys the option row (except for summary options case – see below).

List range

Where? *QDemo.dpr* form: *frmRange1*; unit: *fRange1*; template: *tRange1.xls*

You can use all Excel formatting features in list ranges except for cell merging. In this example, we converted the *OrdersRange* range from **Quick Start** into a list range. You can see that the range now contains only one row of the field formulas as well as the option row and column.

We applied a date format to the cells referencing fields *SaleDate* and *ShipDate*, then checked the *Use 1000 Separator ()* checkbox for *Items Total* and *Amount Paid* fields. We left unchanged the conditional format of the *Payment Method* field. You can see the report in the figure below. Note the difference between the *OrdersRange* selected in both figures. The range in the report now has no option row and contains the transferred data. In a report, you can refer to the data by the range name, and perform any operations needed with VBA.

OrdersRange =

List range

afalinasoft.com
XL REPORT
with XLOptionsPack™ **4**

Customer: #####

Address: #####

City: #NAME? State: ##### Country: ##### Zip: #####

Order No	Sale date	Ship date	Addr1	Addr2	Payment method	Items total	Tax rate	Amount paid
#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#####	#NAME?

Phone: #NAME? Fax: #NAME?

Sheet1

OrdersRange =

List range

afalinasoft.com
XL REPORT
with XLOptionsPack™ **4**

Customer: **Unisco**

Address: **PO Box Z-547**

City: **Freeport** State: Country: **Bahamas** Zip:

Order No	Sale date	Ship date	Addr1	Addr2	Payment method	Items total	Tax rate	Amount paid
1 060	02.28.89	03.01.89			Check	15 355,00		15 355,00
1 073	04.15.89	04.16.89			MC	19 414,00		19 414,00
1 102	06.06.92	06.06.92			Credit	2 844,00		2 844,00
1 160	06.01.94	06.01.94			Check	2 206,85		2 206,85
1 173	07.16.94	07.16.94			MC	54,00		54,00
1 178	08.02.94	08.02.94			Credit	5 511,75		5 511,75
1 202	10.06.94	10.06.94			Credit	4 205,00		4 205,00
1 278	12.23.94	12.23.94			Credit	11 568,00		11 568,00
1 302	01.16.95	01.16.95			Credit	24 485,00		24 485,00

Phone: 809-555-3915 Fax: 809-555-4958

Sheet1

While processing any type of range (including a list range) XL Report works in the following order:

- Inserts the required number of rows into the range (not rows into the sheet! This can be checked by inserting an arbitrary value into any cell to the right of the range).

- Transfers all visible records (active filters and *SetRange* boundary conditions are taken into account) into the rows added.
- Applies all formats of the template cells to the range.
- Deletes the template cells.
- After processing all options, deletes the option row, except for the case of summary options.

Range options and column options

Where? *QDemo.dpr* form: *frmRange2*; unit: *tRange2*; template: *tRange2.xls*

In addition to data transfer, XL Report incorporates a number of other highly useful features. For instance, you can sort, total, and group your data, filter rows, and pivot tables among other things.

To use these additional features you need only change a template workbook specifying range and column options in the heretofore unused option row and option column respectively. **An option itself is a string recognizable by XL Report analyzer.**

Let's take the previous example, open the template and write down **Sum** (it's a column option) just below the field with the field formula for the *Amount Paid* field. We checked *Use 1000 separator ()* checkbox in the *Properties* of this cell. To complete the task we wrote down **Total** (it's just a string) in the adjoining cell. You can see the part of the report that was changed. Instead of the **Sum** option, you see the sum total for the *Amount Paid* column.

Payment method	Items total	Tax rate	Amount paid
Check	15 355,00		15 355,00
MC	19 414,00		19 414,00
Credit	2 844,00		2 844,00
Check	2 206,85		2 206,85
MC	54,00		54,00
Credit	5 511,75		5 511,75
Credit	4 205,00		4 205,00
Credit	11 568,00		11 568,00
Credit	24 485,00		24 485,00
Total			85 643,60

This was an example of **column option** use. Such options are applied only to the column they reside under. The **Sum** option is the most commonly used. Other options are described later.

Range options describe actions to be taken over the range as a whole. They are to be specified in the first cell of the option row. We put down **AutoFilter** and **OnlyValues** strings, separated by a semicolon, into this cell. It's easy to deduce that the first one should apply the *AutoFilter* command to the current sheet. The result should look like:

Order No.	Sale date	Ship date	Addr1	Addr2	Payment method	Items total	Tax rate	Amount paid
1 060	02.28.89	03.01.89			(All)	15 355,00		15 355,00
1 160	06.01.94	06.01.94			(Top 10...)	2 206,85		2 206,85
1 102	06.06.92	06.06.92			(Custom...)	2 844,00		2 844,00
1 178	08.02.94	08.02.94			Check	5 511,75		5 511,75
1 202	10.06.94	10.06.94			Credit	4 205,00		4 205,00
1 278	12.23.94	12.23.94			MC	11 568,00		11 568,00
1 302	01.16.95	01.16.95			(Blanks)	24 485,00		24 485,00
1 073	04.15.89	04.16.89			(NonBlanks)			
1 173	07.16.94	07.16.94			MC	19 414,00		19 414,00
					MC	54,00		54,00
						total		85 643,60

OnlyValues lets you substitute all formulas of the range with their values. Check the *Amount Paid* grand total cell to verify this statement. Using this option, you can “turn off” all formulas of a range, a sheet, or a report. So you see that sheets and reports have their options.

Formatting an arbitrary range

Where? QDemo.dpr form: frmQuickStart; unit: fQuickStart; template: tQuickStart.xls

You probably know by this time that an **arbitrary range** is a range that does not meet list range specifications (see [Appendix A](#)). As we said before, you can merge cells in a range or its header, or omit the header. We refer you to **Quick Start** once again, as it shows almost every possible formatting feature. We also say again that arbitrary ranges are well suited for primary documents (such as invoices) especially if you want to pretty them up.

Quick question: If an arbitrary range is deprived of list range capabilities, then what will calculate the things needed?

Quick answer: VBA.

Range-dataset data in Excel formulas

Where? QDemo.dpr form: frmRange3; unit: fRange3; template: tRange3.xls

When working in Excel you have probably come across the necessity of copying formulas across several cells. Then you probably know that the known effect works because of relative cell references in formulas. XL Report works the same way. Let's look more closely at the **Quick Start** example.

There is a formula **=I13-L\$4** in the **J14** cell of the **QuickStart.xls** template. Look at the *Special Tax* cells in the generated report. Relative referencing is preserved and calculated values are right. So you can create almost any formula that is permissible in Excel.

We demonstrate the only restriction in the same example. Enter the formula **=K13/10** into the **L17** cell of the template. Save the template, close it and run the report. You might suppose that the appropriate cell residing under the *OrdersRange* range must contain a value equal to *Amount Paid* from the first row of the range divided by 10. But instead of the value you see the **#REF!** error message. Why? The reason is simple – according to the schedule the template cells must be deleted after data transfer. In our case the **K13** cell was deleted and so the formula referencing this cell became erroneous. Conclusion: **you can't refer the range's cells outside of the range.**

Simple range options

Where? QDemo.dpr form: frmRange4; unit: fRange4; template: tRange4.xls

All through the past several chapters, you prepared to begin to use the range options. Do you remember that these options (range options) must be put in the lower left cell of the range and separated by semicolon? Here we list the simplest range options:

- **OnlyValues** – as it was said earlier, this option means substituting all formulas in the range for their values.

- **AutoFilter** – applies the *AutoFilter* command (*Data|Filter|AutoFilter* menu in Excel) to list ranges. The use of this option in arbitrary ranges leads to an exception.
- **RowsFit** – this option automatically makes the range row height fit the range cell's contents. It can be done manually – select the rows needed and double-click on the boundary between any of the selected rows. This option is especially useful if you want to output strings containing the carriage return.
- **ColsFit** – The effect this option produces can be

The screenshot shows an Excel spreadsheet with a form template. The form includes fields for Customer, Address, City, State, Country, Zip, Phone, Fax, and Contact. A table with 6 columns (Order No, Sale date, Ship date, Pa y m, ax rate, Amount paid) is also visible. The cell A16 contains the text 'colsfit'.

obvious. So this option is especially useful if you want to output strings whose lengths are unpredictable.

Just for a test, let's shrink all range cols in the template of the previous example down to just one or two character width. Having this done, put **ColsFit** option in the lowest left cell of the range, save the template, close it, and run the report. The columns must be extended to their optimal width. Does it work?

Time to look back - Range-datasets

You already know that:

- XL Report datasets whose corresponding *DataSources* item *Range* property are not left empty are called **Range-datasets**.
- An XL Report range, which serves as a data receiver, must have a special format. Minimal range size is two columns in two rows. The leftmost column and bottom row of a range are treated in a special way as they are used in specifying additional actions over a column or a range that XL Report performs.
- There are two types of ranges in an XL Report: **list range** and **arbitrary range**.
- A list range must meet all the requirements of an Excel list (see [Appendix A](#))
- The leftmost column and bottom row of a range are reserved for range options and column options. Range options should be put into the range's lower left cell, while column options should be put into the column's option cell.

- An option is a string value recognized by XL Report. You can use several options at once if you separate them by a semicolon.
- Arbitrary formatting (including conditional formatting) can be applied to Range-dataset cells. However, you cannot merge cells in list ranges.
- XL Report processes a range according to the schedule:
 - Inserts the required number of rows into the range (not rows into the sheet! Can be easily checked – just insert an arbitrary value in the cell to the right of the lower row of the range).
 - Transfers all visible records (active filters and *SetRange* boundary conditions are taken into account) into the rows added.
 - Applies all your template cells' formats to the range.
 - Deletes the template cells.
 - After processing of the options, deletes the option row, except for the case of options that allow you to insert totals – summary options.
- You can use common Excel formulas that can reference the field formulas. You cannot reference a range's cells in out-of-range formulas.
- You can insert a sum for a column automatically by using the **Sum** option in the column's option cell.
- Simple options can help you to fit your row height (**RowsFit**) and column width (**ColsFit**), to turn off all formulas (**OnlyValues**) and to turn on the AutoFilter (**AutoFilter**) in a range.

Options in a list range

You can use only simple range options in an arbitrary range (excluding an **AutoFilter** that is used in a list range only, as you may recall). This isn't an XL Report restriction – it is an Excel restriction. You need to understand that **all** the Excel capabilities in data grouping, analyzing, and filtering are based on list processing methods. Sorting, totaling, grouping, etc. are done not by XL Report but by Excel itself, so the existing restrictions on the report design are Excel restrictions. Remember the main thing – if you would like to total, filter, sort, group etc. you have to create and use list ranges (or just lists in Excel terminology – see [Appendix A](#)). We refer to a lot of Excel documentation here; that's because we plan to talk about things that are probably well known. Now we'll begin.

Sorting a list range

Where? QDemo.dpr form: frmLists1; unit: fLists1; template: tLists1.xls

Just use the **Sort** option in the option cell of the columns that are needed. We opened the previous form, opened the template, and deleted all options in the *OrdersRange* range. In the option cell of the *PaymentMethod* we put the **Sort** option. There was no previous sorting in this range. The primary key of the *Orders* table provided the default sorting. Now the report is sorted on the *PaymentMethod* column. You can sort the column in descending order by adding the **Desc** option to the **Sort** option and separating them by semicolon.

Order No	Sale date	Ship date	Addr1	Addr2	Payment method	Items total	Tax rate	Amount paid
#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?
					sort;desc			

Sorting in XL Report has some restrictions. First, a list range (a list) can be sorted only on three columns maximum. This is the restriction of the *Sort* method of the *Range* object. Second, XL Report can sort a list range only in left-to-right order. If you set **Sort** option for more then three columns then only the three leftmost columns will be used for sorting.

Totals in a column

Where? QDemo.dpr form: frmLists2; unit: fLists2; template: tLists2.xls

In the previous chapter, we showed you how to get a sum of column values in a list range. The formula generated by XL Report for the column was **=SUBTOTAL(9;...)**, where **9** stands for **Sum** (See Excel Help if you need assistance). XL Report supports all the summary functions of **SUBTOTAL**. Here are the respective options:

- **Sum** – the sum of the values in a column;
- **Count** – the number of items in a column;
- **CountNums** – the number of records or rows in a column that contains numeric data;
- **Avg** – the average of the values in a column;
- **Max** – the largest value in a column;
- **Min** – the smallest value in a column;

- **Product** – the result of multiplying all the values in a column;
- **StDev** – an estimate of the standard deviation of a population, where the column is the sample;
- **StDevP** – the standard deviation of a population, where the column is the entire population;
- **Var** – an estimate of the variance of a population, where the column is the sample
- **VarP** – the variance of a population, where the column is the entire population.

You can get only one summary option for a column. If you provide several options (separated by a semicolon, of course), then XL Report will use only the last one.

Payment method	Items total	Tax rate	Amount paid
#NAME?	#NAME?	#NAME?	#NAME?
Total			sum

We warned you that in the case of summary option the option row isn't destroyed. Notice the **Total** string in the **I13** cell of the template. Its contents confirm our statement that the option row will be deleted. This is the one and only exception.

Subtotals in a list range

Where? QDemo.dpr form: frmLists3; unit: fLists3; template: tLists3.xls

Summary options can be used in combination with the **Group** option. Provide this one for columns where you want to get subtotals. Here is an example. Open the template of the previous example and write down **Sum** in *Items Total* and *Amount Paid* column option cells. In the option cell of the *Payment Method* column write down the **Group** option. Save, close, and run the report. You can see the result – subtotals are inserted in the *Payment Method* column.

Order No	Sale date	Ship date	Addr1	Addr2	Payment method	Items total	Tax rate	Amount paid
#NAME?	#NAME?	#NAME?	####	#####	#NAME?	#NAME?	#####	#NAME?
group						sum	sum	

10										
11		Order No	Sale date	Ship date	Addr1	Addr2	Payment method	Items total	Tax rate	Amount paid
12	•	1060	02.28.89	03.01.89			Check	15 355,00		15 355,00
13	•	1160	06.01.94	06.01.94			Check	2 206,85		2 206,85
14							Check Total	17 561,85		17 561,85
15	•	1102	06.06.92	06.06.92			Credit	2 844,00		2 844,00
16	•	1178	08.02.94	08.02.94			Credit	5 511,75		5 511,75
17	•	1202	10.06.94	10.06.94			Credit	4 205,00		4 205,00
18	•	1278	12.23.94	12.23.94			Credit	11 568,00		11 568,00
19	•	1302	01.16.95	01.16.95			Credit	24 485,00		24 485,00
20							Credit Total	48 613,75		48 613,75
21	•	1073	04.15.89	04.16.89			MC	19 414,00		19 414,00
22	•	1173	07.16.94	07.16.94			MC	54,00		54,00
23							MC Total	19 468,00		19 468,00
24							Grand Total	85 643,60		85 643,60
25										

You can provide up to 16 columns for subtotals. Add the **Group** option to the *Sale Date* column. In the report you see that 1) subtotals are inserted on *Sale Date* and *Payment Method*; 2) the *OrdersRange* range contains both original data and subtotals; 3) grouping is made in left-to-right order – subtotals are calculated first on the leftmost column with **Group** option, then the next grouped column is on its right, and so on. This is another feature of XL Report to remember.

In contrast to the total of a column, you can have several subtotals. If you add the **Avg** option in the *Amount Paid* column and run the report, you will see the sum and the average of this column.

Subtotaling implies previous sorting of data (*Sale Date* and *Payment Method* columns in our case). By default, XL Report does the work itself. But if you have sorted the dataset, you can use the **GroupNoSort** range option. It is just a tip to lessen the time it takes to generate a report. We remind you to put range options in the leftmost bottom cell of a range.

Hiding detail data in an outline

Where? QDemo.dpr form: frmLists4; unit: fLists4; template: tLists4.xls

Add the **COLLAPSE** parameter to the **Group** option in the previous example to hide detail data in a subtotaled report. It can be useful if your customer prefers to see grand totals immediately and then decipher them by using the outline symbols.

	10									
	11	Order No	Sale date	Ship date	Addr1	Addr2	Payment method	Items total	Tax rate	Amount paid
<div><div></div><div>+</div><div>+</div><div>+</div><div>-</div></div>	14						Check Total	17 561,85		17 561,85
	20						Credit Total	48 613,75		48 613,75
	23						MC Total	19 468,00		19 468,00
	24						Grand Total	85 643,60		85 643,60
	25									

XL Report uses only the rightmost option column containing the **Group** option with this parameter.

Time to look back - list range options

- A list is the basis of Excel power, thus giving us the ability to sort, filter, group, etc.
- A Range-list can be sorted via the **Sort** option. A maximum of three columns can be sorted. Default sorting order is ascending. To sort in descending order, use the **Desc** option. In the case of several columns, fields for sorting are taken in left-to-right order.
- For a column in a list range, you can get a total that will be placed directly below the column (only one total value per column). You can make use of any function of the **SUBTOTAL** function using the appropriate options.
- In a list range, you can get subtotals on several columns: subtotaled columns must have any of the summary options and grouped columns must have the **Group** option.
- The maximum number of grouped columns is 16. Fields for grouping are taken in left-to-right order.
- If there are subtotals in a report, you can indicate the level of a subtotals' outline to be hidden via the **Collapse** parameter of the **Group** option.

Pivot tables

Excel PivotTable gives us a really convenient tool to analyze our data. The possibility of quickly changing the data layout and analyzing it without programming, using only mouse (and brains) is a gift for some customers (and a cool tool for programmers, we add). PivotTable is a tool to organize and reorganize our data in order to summarize quickly large amounts of information. It adds multi-dimension features to a relational table. You can rotate its rows and columns to see different summaries of the source data, filter the data by displaying different pages, or display the details of areas of interest. A pivot table consists of the row area, column area, data area, and page area. Fields of the page area add other dimensions to a 2-D table.

Here is an example of a pivot table that is used to analyze orders.

C2		=	Blue Sports				
	A	B	C	D	E	F	G
1							
2		Company	Blue Sports				
3							
4						Data	
5		Payment method	OrderNo	Ship date	Tax rate	Items total	Amount paid
6		Cash	1083	05.10.89	0,00	11 164,80	11 164,80
7			1283	12.30.94	0,00	7 134,00	7 134,00
8		Cash Sum				18 298,80	18 298,80
9		Credit	1012	05.20.88	0,00	5 201,00	5 201,00
10			1057	02.19.89	0,00	1 975,00	1 975,00
11			1061	03.04.89	0,00	24 277,30	24 277,30
12			1091	06.01.89	0,00	1 950,00	1 950,00
13			1161	06.04.94	0,00	102 453,60	102 453,60
14			1212	11.14.94	0,00	3 975,75	3 975,75
15			1261	12.11.94	0,00	1 999,00	1 999,00
16		Credit Sum				141 831,65	141 831,65
17		MC	1168	07.04.94	0,00	104,00	104,00
18		MC Sum				104,00	104,00
19		Visa	1148	03.03.94	0,00	5 011,00	5 011,00
20		Visa Sum				5 011,00	5 011,00
21		Grand Total				165 245,45	165 245,45
22							
23							

To create a pivot table you have to create a list range in the template where **PivotTable** options are used to describe the pivot table fields and their formats. Data will be transferred into the list range and XL Report will create a pivot table. When designing the template, you can designate fields for all the areas of a pivot table, choose summary function for the fields and define their appearance.

Your first pivot table

Where? QDemo.dpr form: frmPivot1; unit: fPivot1; template: tPivot1.xls

We would like to remind you that the Excel PivotTable works with 2-D sources – a list on a sheet, so you have to supply it with a single dataset.

There are two tables in **DBDEMOS – Customer.Db** and **Orders.Db**. We joined these tables on the *CustNo* field and got orders for every customer. Here is the SQL statement:

```
SELECT * FROM Customer, Orders WHERE Customer.CustNo = Orders.CustNo
```

Look at the form. We placed the dataset (*TQuery*) on the form and filled the *SQL* property with the previous SQL statement. We chose the name of the component – *qryOrders*. Then we added the *TDataSource* instance, the grid, and the navigator. Then it was the turn of a *TxlReport* component. We placed it onto the form, named it *xlReport*, added one item in its *DataSources* collection and bound the item with *qryOrders*. We also added the *Report* and *Template* buttons and loaded their *Click* events with calls to *Report* and *Template* methods accordingly. The final step in our preparations – **xlReport.XLSTemplate = “tPivot1.xls”**.

Let's create the template taking the next picture as an example. Put the **PivotName=OrdersPivot\Dst=Pivot!R4C2\ColumnGrand** option in the leftmost cell of the bottom row. This way we tell XL Report to build a pivot table, name it *Orders1*, and place it on the *Pivot* worksheet starting from the *R4C2* cell. The **ColumnGrand** parameter allows totals for columns of the pivot table. Then put the **Row** option in the option cells of *Payment method*, *OrderNo*, *Ship date* and *Tax rate* columns. This way we tell XL Report to use these fields in the Row Area. To get subtotals on payment method add the **Sum** option to the

	A	B	C	D	E	F	G	H
1		First Pivot Table						
2								
3		Company	Payment method	OrderNo	Ship date	Items total	Tax rate	Amount paid
4		#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?
5		pivot\page	row,sum	row	row	data	row	data
6								
7								

existing **Row** option of the *Payment method* field. Use the **Data** option for the *Amount paid* and *Items total* fields (pivot table data area fields). Add the **Page** option to the options of *Company* field, thus referring it to the Page Area. Don't forget to apply appropriate formats to cells. Format the option cells, to make the results of calculations in the pivot table show with this format applied. Apply any special color to the *Payment method* field. That's the template. You saw the report at the beginning of the chapter.

To make the *Pivot* worksheet active after report generation, indicate its name in the *TxlReport.ActiveSheet* option.

Effective cross-tables

Where? QDemo.dpr form: frmPivot2; unit: fPivot2; template: tPivot2.xls

As you see, building a pivot table means using the **Pivot** option in a list range. Then the list range becomes a data source for the pivot table. The **Pivot** option differs from other options in that it can have several arguments separated by a backslash. The syntax is:

PivotName=PivotTableName [**\Dst=Destination**] [**\DataToRows**] [**\RowGrand**] [**\ColumnGrand**] [**\NoPreserveFormatting**]

Where:

Name=PivotTableName – the Excel name for the pivot table.

Dst=Destination – cell in the upper left corner of the pivot table. If you omit the *Destination* then the pivot table will be placed on a new sheet called *PivotTableName* starting from the A1 cell.

DataToRows – allows placing data fields in the pivot table Row Area. By default, data fields are placed in the Column Area.

RowGrand – lets you see grand totals for rows in the pivot table.

ColumnGrand – lets you see grand totals for the columns in the pivot table.

NoPreserveFormatting – allows the pivot table building without preserving the source range's formatting (see "Speeding up the Pivot table generation" in **QDemo**).

Some examples of the **Pivot** option use:

"Pivot\Name=Pivot1\Dst=Totals!A1" – generate Pivot1 pivot table on the Totals worksheet beginning from the A1 cell.

"Pivot\Name=Pivot25\DataToRows" – generate a Pivot125 pivot table on the current worksheet and place data fields in rows.

"Pivot\Name=Pivot25\Dst=Totals!R1C1:R1C1\RowGrand" – add grand totals for the rows of the pivot table.

"Pivot\Name=Pivot25\ColumnGrand" - add grand totals for the columns of the pivot table.

You probably noticed that fields were added to all Pivot table areas in left-to-right order. So you have to place the fields related to each area in the same order you need to display them in the final pivot table.

Note

Field names of a pivot table are taken out of the list range head. Be careful with the field names in pivots – neither Excel 97 nor Excel 2000 (2002) like field names containing a carriage return. The situation has not changed with arrival of Excel 2002.

For the example in this chapter, we took the previous one and changed its template. First, we added a new worksheet – *Sheet2*. Second, we indicated the cell in the left upper corner of the pivot table – **Dst=Sheet2!R8C2:R8C2** (you can also use **Dst=Sheet2!B8**). To get grand totals for rows we added the **RowGrand** option. To do the same thing for columns you can add the **ColumnGrand** option.

As a result, we got: **Pivot\Name=OrdersPivot\Dst=Sheet2!R8C2:R8C2\RowGrand**. Then we added: *Company* field (and indicated that we wanted to get a subtotal on it – the **Sum** option) as well as the *OrderNo*, *Ship date* and *Tax rate* fields to the row area (the **Row** option). We added the *Payment method* field to the

A4		= pivot\name=OrdersPivot\dst=Sheet2!R8C2\columngrand\rowgrand						
	A	B	C	D	E	F	G	H
1								
2		Company	Payment method	OrderNo	Ship date	Tax rate	Items total	Amount paid
3		#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?
4		pivot\row;sum	column	row	row	row	data	data
5								
6								

column area, and we left the *Items total* and *Amount paid* fields in the data area. This report differs from the previous one. Take a look at the *Payment method* field in the generated report – the use of fields with few values leads to the most readable and understandable reports.

We used the **Sum** option here. You can use any appropriate summary option – minimum and maximum values, deviations, and so on. You can also get several subtotals by specifying the options needed.

Formatting a Pivot table

Where? QDemo.dpr form: frmPivot3; unit: fPivot3; template: tPivot3.xls

To format a pivot table in XL Report you have to format a source range. Then XL Report applies the template cell formats to the corresponding cells of a pivot table. The field totals get the formats of corresponding option cells. We used the color in the option cell to print *Company* field totals in yellow. XL Report will transfer only the background color, font, and number format. It is just another Excel restriction.

Now for a more advanced example. The goal of a future report is to show minimum and maximum values of the order sum per customer, and subtotals on payment method. The solution is shown in the figure below. Note the cell coloring. See the result in **Quick Demo XL Report**.

	A	B	C	D	E	F	G	H
1								
2		Company	Payment method	OrderNo	Ship date	Tax rate	Items total	Amount paid
3		#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?
4		row;max;min	row;sum	row	row	row	data	data
5								

You should be aware of Excel pivot table limitations. Some information can be found in Excel Help. More detailed review of the limitations can be found in MSDN:

- XL2000: Limits of PivotTables in Microsoft Excel 2000 (ID: Q211517)
- XL97: Limits of PivotTables in Microsoft Excel 97 (ID: Q157486)

Static Pivot tables

Where? QDemo.dpr form: frmPivot5; unit: fPivot5; template: tPivot5.xls

Starting from build 115 XL Report offers a more advanced technique of Pivot table creation. Now you can place one or more Pivot tables in a report template using comfortable PivotTable Wizard. We used the first Pivot table sample as a base. Its template contains the SourceRange range at the Sheet1 worksheet. For building the pivot table, we needed to select a source area. This area doesn't coincide with the SourceRange. See below how we selected the source area.

B	C	D	E	F	G
Drop Page Fields Here					
			Data		
Company	Payment method	OrderNo	Amount paid	Items Total	Calculated field
#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?
#NAME? Total			#NAME?	#NAME?	#NAME?
#NAME? Total			#NAME?	#NAME?	#NAME?
Grand Total			#NAME?	#NAME?	#NAME?

PivotTable

PivotTable

Company	Payment...	OrderNo	Ship da...	Items t...	▲
Tax rate	Amount ...	Field1			▼

Page 34

A5		= pivot\Refresh=PivotSheet!PivotTable1,Pivot Sheet1!Second PivotTable							
	A	B	C	D	E	F	G	H	I
1		Static Pivot Table				afalinasoft.com XL REPORT with XLOptionsPack™ 4			
2									
3		Company	Payment method	OrderNo	Ship date	Items total	Tax rate	Amount paid	
4		#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	
5		pivot\Refresh=PivotSheet!PivotTable1,Pivot Sheet1!Second PivotTable							
6									

Excel 2000 and Excel XP users can also make use of Pivot charts in a template. Both Pivot tables and Pivot charts will be refreshed after data transfer.

Time to look back - pivot tables

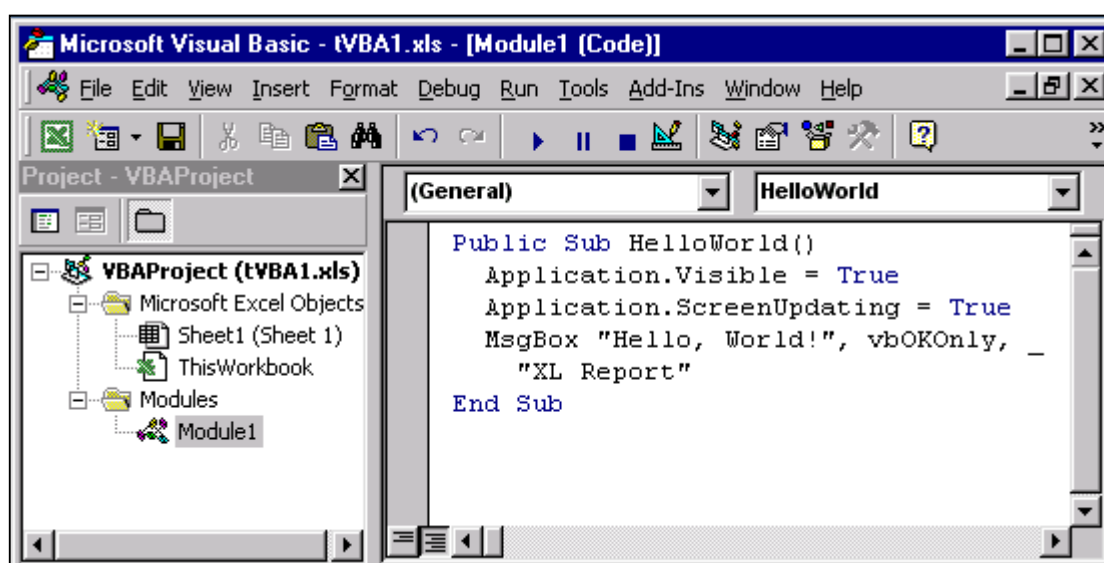
You know that:

- You can automatically build a pivot table based on a list range using appropriate options.
- The **Pivot** option has several arguments allowing you to name a Pivot table, to define its place in a report, to use grand totals, and others.
- If the *Destination* parameter is missing then XL Report sets up a pivot table on a new sheet.
- The **Row** option designates the field in the row area of a pivot table, **Page** – the page area, **Column** – the column area, **Data** – the data area. Along with these options, you can use any summary options.
- XL Report uses the names from the header of the corresponding list range to name pivot table fields.
- XL Report assigns the background color, font, and number format of the list range cells to the appropriate cells of the target pivot table. Option cell formats are transferred into subtotal cells of the pivot table.
- The **ActiveSheet** property allows you to specify the name of the worksheet that will become active after report generation.
- Use the **NoPreserveFormatting** parameter for speeding up pivot table creation.

Using VBA

If you cannot complete your task by the methods described earlier you should consider using VBA. That way you will get all of Excel's power at your fingertips.

The *TxlReport* class as well as the *TxlDataSource* class represents items of the *DataSources* collection. Both have two published properties – *MacroAfter* and *MacroBefore*. These properties are used to call public VBA procedures. So their values (they are strings) must contain the full names of public VBA procedures (they are called **macros** in Excel), saved along with the template workbook. We use macros to build Excel charts in our own work. But you can make anything you want, or rather whatever Excel is capable of. And Excel can make lots of nice things.



To invoke VBA procedures you must add a module in a template workbook and create a public procedure in it. If you specify the module and procedure names in the *MacroBefore* and/or *MacroAfter* properties in *ModuleName.ProcedureName* format then these procedures will be invoked automatically.

Note

Excel may issue warnings about macros with the examples in this section. Don't disable macros when you open these workbooks.

Simple example

Where? QDemo.dpr form: frmVBA1; unit: fVBA1; template: tVBA1.xls

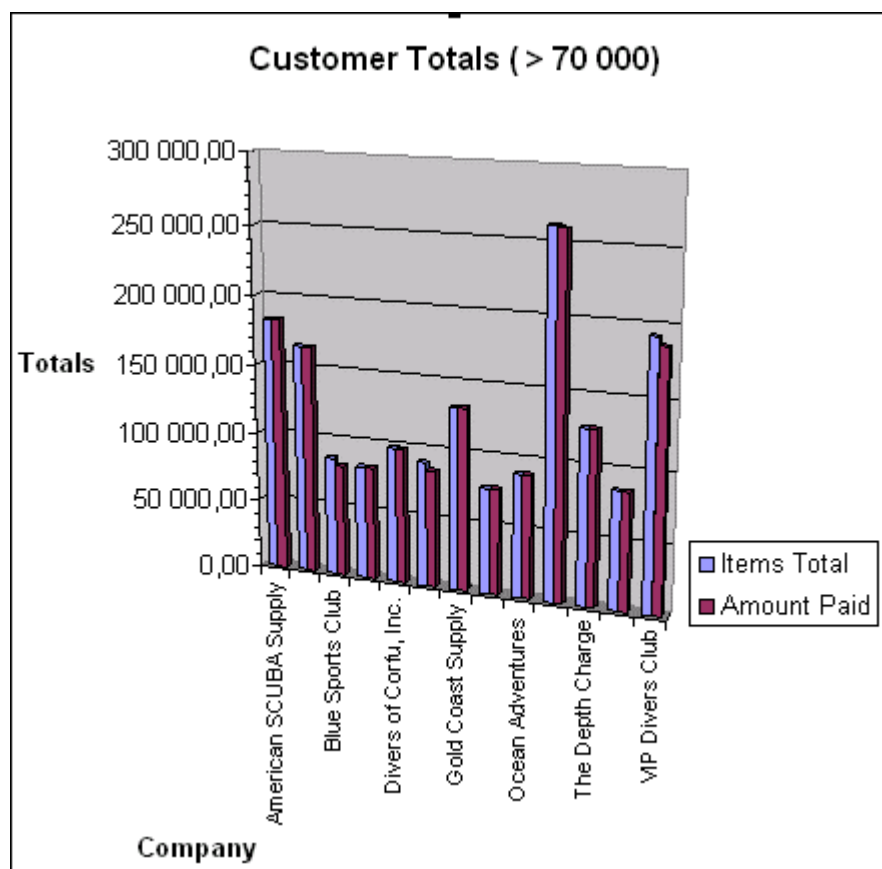


We took **Quick Start** as an example. We opened the template workbook, added a module with the *Module1* name into the workbook and created a public procedure named according to tradition *HelloWorld* – we called *MsgBox* to display the “Hello, world!” message. Then we saved the template and closed it. As we said earlier, we must supply XL Report with the full name of the procedure. This name is *Module1.HelloWorld*. We put it into the *MacroAfter* property of the *TxlReport* instance. Now you can see the result – just run the report.

The main question in using such procedures is how can you access the data in these macros? It's simple. You make use of NoRange-datasources' named cells and Range-datasources' named ranges.

Excel charts

Where? QDemo.dpr form: frmVBA2; unit: fVBA2; template: tVBA2.xls



Here are totals for customers that have orders totaling more than \$70,000. Let's look at how it was done.

First, a dataset: we took the first pivot table example as a base. We wrote the following SQL statement in the SQL property of *qryAll*:

```
SELECT c.Company,
       Sum(o."ItemsTotal") as ItemsTotal,
       Sum(o."AmountPaid") as AmountPaid
FROM Orders o, Customer c
WHERE c.CustNo = o.CustNo
GROUP BY c.Company
ORDER BY c.Company
```

This query returns company names along with the summed values of the *ItemsTotal* and *AmountPaid* fields out of the *Order* table. Since we used a prepared example, we had to change the template and create a macro in it. You can see the template in the figure at your right. The macro (*BuildChart* is its name) creating this chart was recorded in Excel (*Tools|Macro|Record New Macro*), saved in one of *tVBA2.xls* modules and corrected in order to work with the *AllRange*. We think that the most interesting part of this code is the statement:

AllRange		=	
A	B	C	D
1			
2			
3	Company	Items Total	Amount Paid
4	#NAME?	#NAME?	#NAME?
5	AutoFilter		
6			
SourceData			

```
SrcRange.AutoFilter Field:=3, Criteria1:=">70000", Operator:=xlAnd
```

This line sets a condition for the **AutoFilter** created for this range by XL Report.

Passing parameters to VBA procedures

Where?	QDemo.dpr form: frmVBA3; unit: fVBA3; template: tVBA3.xls
--------	---

We used static criteria in the previous example. Now we want to show you how you can make it dynamic. We added the edit field *edCriteria* to the form. Then we corrected the procedure by adding the **Criteria as string** parameter and changing the code to let **AutoFilter** use this parameter. Full code can be found in the template. Here is the problem: how do we supply the *BuildChart* procedure with the value from the *edCriteria* field? You have to use the *OnMacro* event of the *TxlReport* instance. The code:

```
procedure TfrmVBA3.xlReportMacro(Report: TObject; const AMacroName: String;
  var Arg1, Arg2, Arg3, Arg4, Arg5, Arg6, Arg7, Arg8, Arg9, Arg10, Arg11,
  Arg12, Arg13, Arg14, Arg15, Arg16, Arg17, Arg18, Arg19, Arg20, Arg21,
  Arg22, Arg23, Arg24, Arg25, Arg26, Arg27, Arg28, Arg29,
  Arg30: OleVariant);
begin
  if AMacroName = 'Module1.BuildChart' then
    Arg1 := edCriteria.Text;
  end;
```

The astounding number of parameters was dictated by the syntax of the *Application.Run* method of the Excel Type Library in case of early binding. Parameters must be passed by position. We only had to pass one parameter in our example.

Time to look back - XL Report and VBA

Summing up:

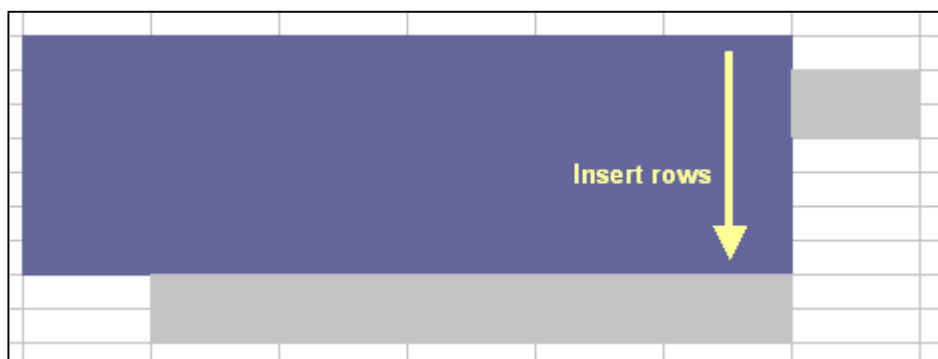
- XL Report can invoke VBA procedures from the template workbook. These procedures must be placed in a standalone module(s) and must be declared *public*.
- *TxlReport* and *TxlDataSource* classes both have *MacroBefore* and *MacroAfter* properties to indicate the full name of the VBA procedures to be invoked.
- *MacroBefore* is triggered before the report generation (*TxlReport*) and before the data transfer (*TxlDataSource*).
- *MacroAfter* is triggered after the report generation (*TxlReport*) and after the data transfer (*TxlDataSource*).
- The *OnMacro* events of these two classes must be used to pass parameters to the called VBA procedures.
- You can automate your routine tasks using Macro Recorder in Excel.

Complex reports

Ranges' placement in XL Report

As we have said previously, the number of ranges (i.e. number of datasets) is not limited, and there is no restriction on the number of datasets that place the only record in a report – **NoRange-Datasets**. Once again, there are no constraints on quantity. There are constraints on range placement. Why?

XL Report sees a range as a single whole, so rows are inserted into range and not into worksheets. It leads to



the simple conclusion – ranges cannot intersect. And what's more, they cannot have common columns if the upper range is narrower than the lower one or ones. The illustration shows you the result of inserting rows in the

blue range. Because this rule was broken, the gray range has lost one of a range's main requirements – continuity.

The next illustration shows you several variants of a ranges' placement in a report. So the conclusion: any underlying range can have common columns with an overlying range only when the columns of the underlying one are a subset of the overlaying range's columns.

Nested ranges - designing master-detail reports

Where? QDemo.dpr form: frmMD1; unit: tMD1; template: tMD1.xls

XL Report allows you to create complex reports by making use of nested ranges. This way you simulate the subordinate relations of your data.

We have created a new form, added three instances of *TTable* and bound them to DBDEMOS (*Customer*, *Orders*, and *Items*). Then we have bound them with each other through the *MasterSource* property. In such a way we have created a data structure that allowed us to look through orders of every customer. To view the data we have added three *DataSources*, several edit fields for *Customer* and two grids (*TDBGrid*) for *Orders*

and *Items*. To get the data in a report we have added an instance of *TxlReport*. Now the template. The *Items* table resides in the lowest level of subordination structure. So its range (we have decided that it should be a list range) must be the innermost one and the first range to start with. We have added field formulas and headers, and gave a name for the range - *ItemsRange*. Note that this range is a list range.

ItemsRange

=

	A	B	C	D	E	F	G	H	I
1	Master-detail report					<div>afalinasoft.com</div> <div>XL REPORT</div> <div>with XLOptionsPack™</div> <div>4</div>			
2									
3				Description		SellPrice	Qty	Discount	ExtPrice
4				#NAME?		#NAME?	#NAME?	#NAME?	#NAME?
5									
6									
7									

Sheet 1

To include orders' data we have added an empty row above the header row for *ItemsRange* and specified the field formulas for Orders datasource, selected the range, which included this row, the header row for *ItemsRange*, *ItemsRange* itself, and one more empty row below it (it would serve as an option row for Orders' range), and named the selected range - *OrdersRange*. Then we added the header row above the *OrdersRange*. The question is: what sort of range is *OrdersRange*? When answering this question you should exclude all inner ranges (the *ItemsRange* in this case) out of consideration. The answer is: the *OrdersRange* is an arbitrary range because of presence of the header row of the *ItemsRange*.

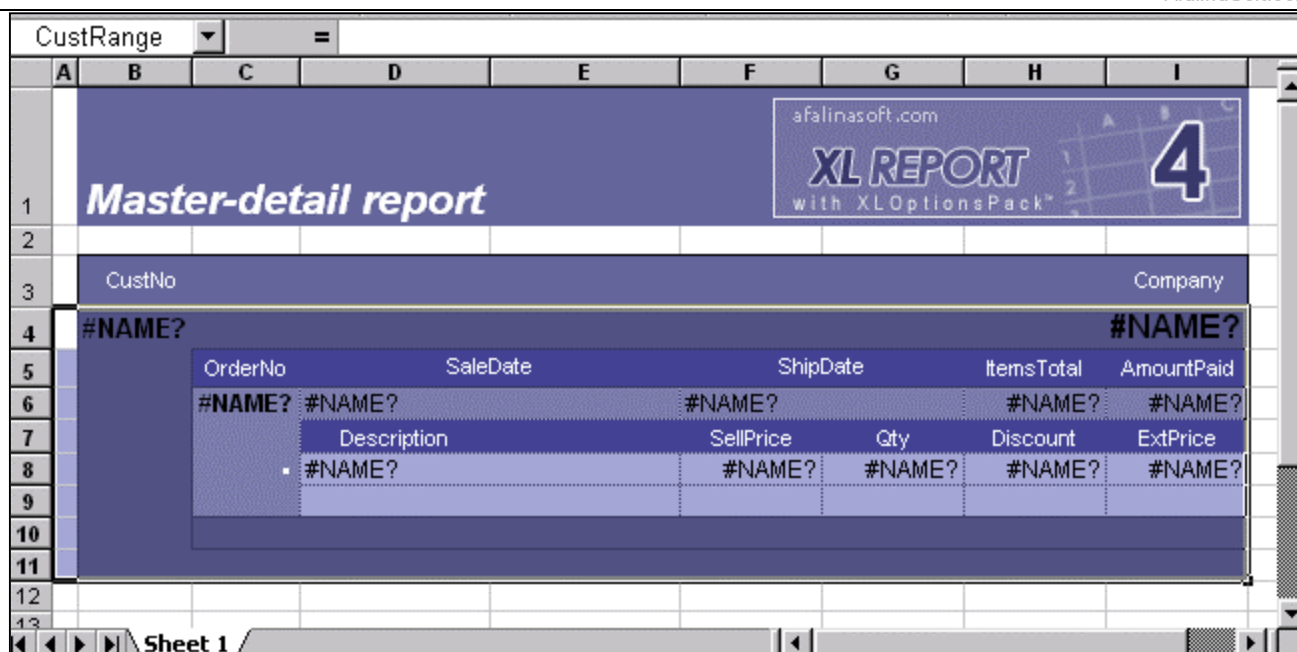
OrdersRange

=

A	B	C	D	E	F	G	H	I	
1	Master-detail report					<div>afalinasoft.com</div> <div>XL REPORT</div> <div>with XLOptionsPack™</div> <div>4</div>			
2									
3		OrderNo	SaleDate		ShipDate		ItemsTotal	AmountPaid	
4		#NAME?	#NAME?		#NAME?		#NAME?	#NAME?	
5			Description		SellPrice	Qty	Discount	ExtPrice	
6			#NAME?		#NAME?	#NAME?	#NAME?	#NAME?	
7									
8									
9									

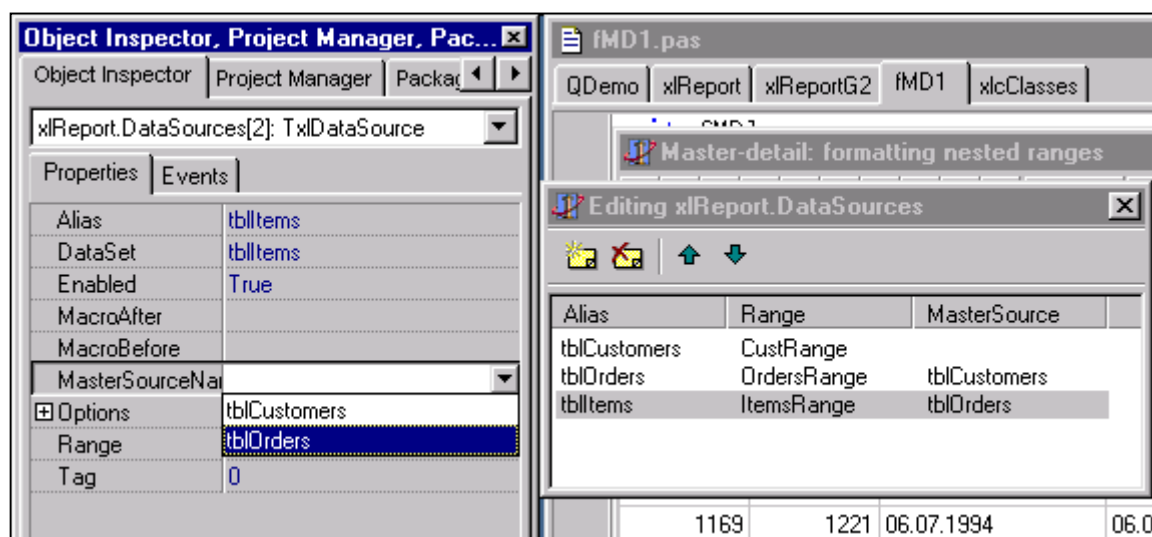
Sheet 1

Then we added another two rows above the header row of the *OrdersRange*: one of them became the header row of the range to be, and field formulas for customer's data (*CustNo* and *Company* fields) were specified in the other one. Then we selected the last one, the header row of the *OrdersRange*, *OrdersRange* itself (recall that it included *ItemsRange* at this moment), and an empty row - it's another option row - and gave the name for the selected range - *CustRange* (see the picture at the following page). Note that the *CustRange* is an arbitrary range because it contains more than one row per record; namely, it contains the header row of the *OrdersRange*. To say truth, we had no special need in so many empty cells at the left of the template. It's just a design solution. But you should remember that a) leftmost columns of all nested ranges must coincide with each other thus forming the common option column for all nested ranges and b) rightmost columns must coincide too.



The template was saved as *tMd1.xls* and bound to the *xlReport* component through the use of the *XLSTemplate* property.

Then three items were created in the *DataSource* collection of the *xlReport* and were bound to corresponding datasets (*DataSet* property) and ranges (*Range* property). To reflect the subordinate structure of our data we used the *MasterSource* property: because the *OrdersRange* was nested in the *CustRange*, the item bound to *Orders* must have a reference to the *CustRange* in its *MasterSourceName* property, and again, because the *ItemsRange* was nested in the *OrdersRange*, the item bound to *Items* must have a reference to the *OrdersRange* in its *MasterSourceName* property.



The following is the full set of rules you will use designing master-detail reports:

- Every range must stick to the above given continuity rules
- Every range must have an option row of its own
- Every range in a nest of ranges must be of the same width and their left (and right) borders must be coincident
- The leftmost column is an option column for every nest of ranges
- There is no restriction on the number of ranges in a nest

- The size of a generated report cannot exceed the size of a worksheet <g>
- An inner range can be placed between any rows of an outer range
- Every range (minus inner ranges) is considered to be a single whole and is either a list range or an arbitrary range whose restrictions were given in corresponding chapters

Totals in a column in master-detail reports

Where? QDemo.dpr form: frmMD2; unit: tMD2; template: tMD2.xls

Now we show you the use of the option row in nested range. We derived this template out of the previous one by making small changes to the option row of the *ItemsRange* range. Both the template and the report are shown below.

CustNo		Company	
#NAME?	#NAME?	#NAME?	#NAME?
OrderNo	SaleDate	ShipDate	ItemsTotal
#NAME?	#NAME?	#NAME?	#NAME?
Description	SellPrice	Qty	Discount
#NAME?	#NAME?	#NAME?	#NAME?
Order totals			sum

CustNo		Company	
1 221	Kauai Dive Shoppe		
OrderNo	SaleDate	ShipDate	ItemsTotal
1 023	7.1.1988	7.2.1988	4 674,00
Description	SellPrice	Qty	Discount
Regulator System	250,00	5,00	1 250,00
Depth/Pressure Gauge	206,00	4,00	824,00
Alternate Inflation	260,00	10,00	2 600,00
Total of order			4 674,00
1 076	12.16.1994	4.26.1989	17 781,00
Description	SellPrice	Qty	Discount
Remotely Operated Vi	2 369,00	4,00	9 476,00
Towable Video Camera	1 999,00	4,00	7 996,00
Second Stage Regulat	309,00	1,00	309,00
Total of order			17 781,00
1 123	8.24.1993	8.24.1993	13 945,00

We remind you that summary options can work in a list range only (the *ItemsRange* is a list range – it contains only one data row, the option row and option column plus it has the header row above). Can you get totals in a column of a range which is torn apart by included ranges? Yes. Look at *OrdersRange* range again. It isn't a list range. But we put the **Sum** option into the option cell of the *AmountPaid* column and get the report.

3	CustNo					Company
4	#NAME?					#NAME?
5	OrderNo	SaleDate	ShipDate	ItemsTotal	AmountPaid	
6	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	
7		Description	SellPrice	Qty	Discount	ExtPrice
8		#NAME?	#NAME?	#NAME?	#NAME?	#NAME?
9		<input type="text" value="sort"/>	Order totals			sum
10		Customer totals				sum
11						

1 280	12.26.1994	12.26.1994	4 317,75	4 317,75	
	Description	SellPrice	Qty	Discount	ExtPrice
	• Dive Computer	650,00	2,00		1 300,00
	• Flashlight (Recharge	169,95	2,00		339,90
	• Halogen Flashlight	59,95	3,00		179,85
	• Marine Super VHS Vid	2 498,00	1,00		2 498,00
	Total of order				4 317,75
	1 305	1.20.1995	1.20.1995	3 065,00	3 065,00
	Description	SellPrice	Qty	Discount	ExtPrice
	• Second Stage Regulat	365,00	1,00		365,00
	• Depth/Pressure Gauge	105,00	22,00		2 310,00
	• 71.4 cu ft Tank	195,00	2,00		390,00
Total of order				3 065,00	
Total of customer				69 364,70	

How does it work? Well, after excluding the *ItemsRange* (the row with field formulas and the option row) out of consideration, you see that the *OrdersRange* contains three rows: the one with field formulas, the header row for now excluded *ItemsRange*, and the option row.

3	CustNo					Company
4	#NAME?					#NAME?
5	OrderNo	SaleDate	ShipDate	ItemsTotal	AmountPaid	
6	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	
7		Description	SellPrice	Qty	Discount	ExtPrice
10		Total of customer				sum
11						

In this case, summary options take only the first row of such a range into account. It contains the field formula for the *AmountPaid* field thus allowing calculating the sum.

Sorting nested ranges

Where? QDemo.dpr form: frmMD3; unit: fMD3; template: tMD3.xls

You can use the **Sort** option at the lowest level of nesting. We remind you that this option work in a list range only. The following template is designed to view the list of items in every order sorted ascending.

Multiple-sheet master-detail reports

Where? QDemo.dpr form: frmMD4; unit: fMD4; template: tMD4.xls

To create a multiple-sheet report you should specify the name of a master dataset in the *MultisheetAlias* property and the name of one of the dataset's fields in the *MultisheetField* property. XL Report, being activated via the *Report* method, will create the report workbook with as many worksheets as there are records in the dataset; each one is named after the corresponding value of the *MultisheetField*.

We have taken the template of the previous example and cleared the *Range* property of the item in the *DataSources* collection that is bound to the *Customer* table. Then we have specified "tblCustomers" in the *MultisheetAlias* and "Company" in the *MultisheetField*. The most significant part of the resulting report follows:



When creating a multiple-sheet report, XL Report loops through records of the *MultisheetAlias* and creates worksheets named after the value of the supplied field (*MultisheetField*). Each record triggers the full set of events for datasources (including macro calls). You are allowed to think about this process as about building several different reports. This mechanism is transparent for all VBA macros and event procedures. Naturally, this transparency has required bringing in some restrictions on design of templates. Here they are:

- A template for a multiple-sheet report must contain only one worksheet;
- If the *MultisheetField* contains identical values, then the names of resulting worksheets has additional numeric identifier added;
- The **OnlyValues** option will be applied to every report worksheet;
- For every report worksheet, every Name object referencing the report worksheet will be destroyed
- The developer is responsible for uniqueness of the names created in macros and event procedures (names of pivots and charts, for instance).

Time to look back - complex reports

Now you know that:

- XL Report imposes some limitations on placement of ranges in a template.
- It allows creating reports based on data linked with one-to-many relation through nesting of ranges and the *MasterSourceName* property.
- You can get totals in a column of a nested range using summary options.
- You can use the **Sort** option in the list ranges only at the lowest level of nesting
- XL Report allows creating multiple-sheet reports. Every record of a master dataset supplied in the *MultisheetAlias* property causes creation of a new (separate) worksheet named after the value of the field supplied in the *MultisheetField* property. Design of such reports has some limitations cited in the "Multiple-sheet master-detail reports" chapter.

Additional features

Other options

In addition to range options and column options, XL Report processes several sheet options and report options. The first ones have an influence upon a worksheet, and the second, accordingly, on a report workbook. Both are used to automate getting the optimal height and width for cells, scaling to the installed printer, hiding some worksheets and protecting them from changes.

Sheet options

Where? QDemo.dpr form: frmOpts1; unit: fOpts1; template: tOpts1.xls

You need to specify the sheet options in the **A2** cell of a given worksheet. When processing every worksheet of the template, XL Report checks that cell's value. When XL Report recognizes any options, it takes some additional actions. Here they are:

- **OnlyValues** – replaces all formulas on a worksheet with their values.
- **Hide** or **SheetHide** – hides a worksheet in a generated report workbook.
- **RowsFit** – makes the range rows fit the contents. It can be done manually – select the rows needed and double-click on the boundary between any of the selected rows.
- **ColsFit** – makes the range columns fit the contents. It can be done manually – select the columns needed and double-click on the boundary between any of the selected columns.
- **AutoScale** – sets up the page option *Fit to 1 page wide by (empty) tall* (this can be found in *File|Page Setup* on the *Page* tab).
- **AutoSafe** – protects a sheet with a random password. This corresponds to *Tools|Protection|Protect Sheet* menu command call and *Contents* box checking.

Note

Using the **AutoSafe** option together with clearing (checking) the "Locked" checkbox in a cell's properties, you can realize a flexible approach to user's modifications of a report. For instance, if you clear the "Locked" checkbox of a given cell in a template workbook and use the **AutoSafe** sheet or report option, then the user still will be able to change the contents of the cell in the report workbook.

The **A2** cell will be cleared only if it contains any XL Report options. So you can use this cell for your own needs. This rule also refers to the **A1** cell that contains report options. Some XL Report users, knowing that report options can be specified on any sheet, create a hidden worksheet with report options.

Report options

Where? QDemo.dpr form: frmOpts2; unit: fOpts2; template: tOpts2.xls

Report options should be placed in the **A1** cell of any sheet (or sheets – in this case all the options supplied will count) of a template workbook. While processing every sheet of a template, XL Report checks this cell. When XL Report recognizes any options it takes some additional actions. Here is the options list:

- **OnlyValues** – replaces every formula in the report workbook with its calculated value.

- **ShowPivotBar** – turns on the *Pivot table* bar. This bar is hidden by default even if the report contains a pivot table. This panel allows the user to edit the Pivot table. See Excel Help for additional information.
- **AutoSafe** – protects a report workbook with a random password. This corresponds to the *Tools|Protection|Protect Workbook* menu command with the *Structure* check box on. The use of **AutoSafe** on report levels completely protects a report workbook because the *Tools|Protection|Protect sheet* menu command is applied to every worksheet in the workbook.

Visualize report generation

Where? QDemo.dpr form: *frmEvents1*; unit: *fEvents1*; template: *tEvents1.xls*

Analytical reports can contain lots of data. So, report generation can take a lot of time. To reassure the user that



the program is progressing appropriately, we recommend you use the *OnProgress* property of the *TxlReport* class declared as follows:

```
property OnProgress: TxlOnProgress;
type
  TxlOnProgress = procedure (Report: TxlReport; Position, Max: integer) of object;
```

Report generation consists of several steps. You get the number of steps in the *Max* parameter. The current step is kept in the *Position* parameter. By using the *TProgressBar* component named *pbReport*, you can process the event this way:

```
procedure TfrmEvents1.xlReportProgress(Report: TxlReport; Position, Max: Integer);
begin
  pbReport.Min := 0;
  pbReport.Max := Max;
  pbReport.Position := Position;
end;
```

Note the assignment statement ***pbReport.Max:=Max***. The report generation algorithm consecutively parses a template. This means that at some stage the total amount of steps needed is unknown. For example, no one can know before list range option processing if there is a need to create a pivot table. That's why the *Max* value can change in run-time.

OnBeforeBuild and OnAfterBuild events

Where? QDemo.dpr form: *frmEvents2*; unit: *fEvents2*; template: *tQuickStart.xls*

XL Report can open and close the datasets engaged in report building automatically. The *TxlReport* instance has the *Options* property which *roAutoOpen* and *roAutoClose* inhabit. If they are **True**, then all unopened datasets will be opened before report building and, accordingly, they will be closed immediately after report building. Since parameterized queries are often used, however, you will need to initialize these parameters. We recommend you use the *OnBeforeBuild* event of *TxlReport* class or the *OnBeforeDataTransfer* event of *DataSources* collection item (*TxlDataSource* class):

```
property OnBeforeBuild: TxlReportHandleEvent;
type
  TxlReportHandleEvent = procedure (xlReport: TObject) of object;

property OnBeforeDataTransfer: TxlDataTransferHandleEvent;
type
  TxlDataTransferHandleEvent = procedure (DataSource: TxlDataSource) of object;
```

The *TxlReport* and *TxlDataSource* classes both have the *Tag* property, which is usual in Delphi and often used to distinguish between components in a common event handler. If using this property the *OnBeforeBuild* handler looks like the following:

```
procedure TfrmEvents2.xlReportBeforeBuild(Report: TObject);
begin
  case TxlReport.Tag of
    10: tblOrders.ParamByName('CustNo').AsInteger :=
        tblCustomers.FieldByName('CustNo').AsInteger;
  end;
end;
```

The form used as example in this chapter shows a list of clients from the **Customer.Db** table. The report goal is to show all orders for a given client. So we created the query whose *CustNo* parameter is initialized in this event handler. The *DataSources* collection item containing *tblOrders* dataset has *rgoAutoOpen* and *rgoAutoClose* flags that were set to **True**. So report building passes through the following stages. First, in the *OnBeforeBuild* handler, the *CustNo* parameter of the query is set equal to the value of the same field in the client table. Then the *rgoAutoOpen* flag is handled – the query is opened and the report is built. After report generation *rgoAutoClose* is handled – the query is closed. We recommend you use this approach to minimize the need for system resources.

TxlReport.Options - options of a report

While working on previous examples, you might run into a property that follows VCL fashion. Some of the Options' flags are complete analogs of sheet and report options of a template; others are accessible in the Delphi Object Inspector only.

```
property Options: TxlReportOptionsSet read FOptions write SetOptions
  default [xroOptimizeLaunch, xroDisplayAlerts, xroAutoOpen];
type
  TxlReportOptions = (xroOptimizeLaunch, xroNewInstance, xroDisplayAlerts,
    xroAddToMRU, xroAutoSave, xroUseTemp, xroAutoOpen, xroAutoClose, xroHideExcel);
  TxlReportOptionsSet = set of TxlReportOptions;
```

Options property flag	Description
xroAddToMRU	True – every template opened at design time, will be added to the Excel list of recently used workbooks. Useful in case of a large number of templates. Default – false.
xroAutoClose	True – automatically closes datasets engaged after report generation. If false – all datasets remain open. Default – false. Attention! This flag change causes the same change in the <i>rgoAutoClose</i> flag for every item in the <i>DataSources</i> collection.
xroAutoOpen	True – automatically opens datasets engaged in report generation if they are closed. If false and a dataset is not open – an exception is raised. Default – true. Attention! This flag change causes the same change of <i>rgoAutoOpen</i> flag for every item in the <i>DataSources</i> collection.
xroAutoSave	True – report workbook is saved via the Workbook.Save method called immediately after generation. If false and <i>roDisplayAlerts</i> is included – Excel raises an exception. Default – false.
xroDisplayAlerts	False – suppresses Excel prompts and alert messages while a macro is running. Can be found in <i>Excel.Application</i> properties. Default – true.
xroHideExcel	True – lets you build a report leaving Excel processes invisible. Can work only if the Excel process is created by XL Report. To determine the path to save the report, use the <i>roUseTemp</i> flag and <i>TempPath</i> property. After being built, the report is closed immediately. Default – false.
xroNewInstance	True – XL Report app creates a separate Excel process. False – XL Report app uses an existing Excel process or launches it anew. Default – false.

xroOptimizeLaunch	To optimize the number of Excel launches. If true, Excel is unloaded only after the application is closed, thus preventing Excel from continually launching while working with XL Report. Default – true . Call the <i>TxlReport.ReleaseExcelApplication</i> method to unload Excel manually.
xroSaveClipboard	XL Report makes use of the Clipboard extensively. This option allows you to preserve its contents. But in some rare cases this can lead to inadequate behavior of Excel objects. It shows itself in wrong formatting of a report. For instance, working on our documentation we had a chance to see that the use of the Clipboard by Adobe PhotoShop causes the <i>PasteSpecial</i> method to work not in a way described in Excel documentation. If you ever come across such situations, please inform us and try to set this option to False (Default).
xroUseTemp	True – a report workbook is generated with the file name in the stencil Template-workbook-file-name & date & time & .xlrtmp . Such files are deleted while the app is closing. The folder name to keep these files is provided in the <i>TxlReport.TempPath</i> property. BTW, this is the only use of the property. If <i>TempPath</i> is empty, the temp files will be created in the application's start folder. False – the temp files will be created in the default working folder (<i>General</i> tab in <i>Tools Options</i> dialog) and will be named according to Excel naming conventions. Default – false .

TxlDataSource.Options

DataSources collection items have their own *Options* property. In contrast to *TxlReport.Options*, this property relates to a given dataset that is bound to the collection item or to the given range into which the data is put. Here they are.

```
property Options: TxlRangeOptionsSet; default [xrgoAutoOpen];
type
  TxlRangeOptions = (xrgoAutoOpen, xrgoAutoClose);
  TxlRangeOptionsSet = set of TxlRangeOptions;
```

Options property flag	Description
xrgoAutoClose	True – automatically closes the dataset associated with the <i>DataSources</i> collection item after report generation. If false – the dataset remains open. Default – false .
xrgoAutoOpen	True – automatically opens the dataset associated with the <i>DataSources</i> collection item if it is closed. If false and a dataset is not open – an exception is raised. Default – true .
xrgoPreserveRowHeight	Keeps the height of rows in a report equals to the height of corresponding rows in a template. Default – True .

Add a report to an existing workbook

Where? QDemo.dpr form: *frmAddF1*; unit: *fAddF1*; template: *tAddF1.xls+tExistingBook.xls*

This example adds the **Sheet1** worksheet to the **tExistBook.xls**. If the **Sheet1** worksheet already exists in the target workbook the added worksheet will be renamed according to Excel naming convention. Thus **Sheet1** will become **Sheet1(1)**. This is done using the following method:

```
procedure ReportTo(const WorkbookName: string; const NewWorkbookName: string = '');
```

The *WorkbookName* parameter allows you to specify the workbook that receives the report's worksheet. The *NewWorkbookName* parameter allows specifying the name of the workbook, which will receive the final report. The following examples demonstrate the logics of the method:

```
xlReport.ReportTo('', ExtractFilePath(ParamsStr(0)) + 'NewBook.xls');
xlReport.ReportTo('..\WB1.xls', ExtractFilePath(ParamsStr(0)) + 'NewBook.xls');
```

The first example causes creation of a report and saving it in **NewBook.xls**, while the second one creates a report, adds it to **WB1.xls**, and saves the resulting report in **NewBook.xls**.

XL Report creates a report going through usual chain of methods and events, and adds all non-hidden worksheets of the report workbook to the workbook specified. The *OnlyValues* option will be applied to every worksheet added. And more, all names referencing the worksheet added will be destroyed. The report worksheets are added after the last existing worksheet.

XL Report fires an exception if the *WorkbookName* file doesn't exist and keeps silence if the *NewWorkbookName* file exists.

Several reports in one workbook

Where? *QDemo.dpr* form: *frmAddF2*; unit: *fAddF2*; template: *tAddF2.xls.xls*

XL Report contains the *MergeReports* class method that allows you to merge several reports in one workbook.

```
class procedure MergeReports(Reports: array of TxlReport;
    SheetPrefixes: array of string);
```

You should pass an array of *TxlReport* instances – the reports that you want to be merged – to the *Reports* parameter. Every process of report generation in this case is isolated from other report generation processes so you will not be bothered with existing event handlers for every report or with macros in their templates. After processing the current report, the *OnlyValues* option will be applied to its visible worksheets and every name (the *Name* object) will be deleted from them (in order to prevent probable name conflicts). All the reports will be added to the first instance of *TxlReport* in the *Reports* array. Every worksheet will be added with its own name prefixed with the prefix taken from the *SheetPrefixes* parameter. You should ensure that both of the arrays have equal number of elements.

Custom options - the XLOptionPack technology

XL Report supports the XLOptionPack technology that allows the developer to create option packages. This technology bases on the architecture of XL Report, which permits expanding the space of options due to specially designed additional components. These components must be descendants of the *TxlOptionPackage* component class that encapsulates the mechanisms required in option package creation. Every option in such a package must descend from the *TxlOption* class. You can find detailed information on creating custom options and option packages in "XLOptionPack Developer's Guide".

Time to look back - other options

Please remember that:

- Along with column and range options, there are sheet and workbook options.
- Sheet options must be placed in the **A2** cell of a given worksheet.
- Report options must be placed in the **A1** cell of any worksheet or worksheets of a workbook.
- Using these options, you can: hide some sheets, protect sheets or the workbook, and substitute formulas with their values.

Unbound data transfer

Working with XL Report you can build your reports using data from different sources and not necessarily datasets. G2 is based on an imported Excel Type Library. It means that you have all the power of Excel's published interfaces. The Excel Type Library will help you to build reports based on sources other than the *TDataSet*. You can use it in event handlers such as *AfterBuild* or *AfterDataTransfer*. But XL Report gives you much more simple mechanisms as well.

Report parameters

Where? *QDemo.dpr* form: *frmUData1*; unit: *fUData1*; template: *tUData1.xls*

Similarly to creating parameterized queries through the use of property, you can create parameterized reports through the use of the *Params* property of *TxlReport*. This property is a collection whose items expose only two properties: *Name* of a string type and *Value* of a Variant type. Having any number of items added to the collection, you are allowed to use a field formula on the *XLRParams_ParamName* pattern anywhere in your report. Here *XLRParams* is a constant part of the formula and *ParamName* is to be changed according to your needs.

We have added the *StartDate* parameter of a Data type to the XL Report component and written the *XLRParams_StarDate* formula in one of the template cells. We initialize the parameter with a value of the *DateTimePicker* component previously added to the form before calling the *Report* method:

```
xlReport.ParamByName['StartDate'].Value := edStartDate.Date;
xlReport.Report;
```

Consider the *Params* to be just another NoRange-dataset whose *Alias* is always "XLRParams" or as unbound NoRange-dataset.

Unbound data in ranges

Where? *QDemo.dpr* form: *frmUData2*; unit: *fUData2*; template: *tUData2.xls*

In order to transfer a lot of unbound data you use another XL Report feature – an unbound Range-dataset.

We have created a new form and added a StringGrid with 4 columns. Let us assume our data of four different types are kept in this grid. The first column will contain strings, the second – integer values, the third – real, and the fourth – DateTime values. The item was added to *xlReport.DataSources* whose *Dataset* property was left empty, *Alias* was named as "Grid", and the "UnboundRange" string was put in the *Range* property. A template workbook was created with the four-column *UnboundRange*; the field formulas were: *Grid_StrValue*, *Grid_IntValue*, *Grid_FloatValue*, *Grid_DateValue*. Here we have assumed that those are the field names (*StrValue*, *IntValue*, and so on). Look, the process is the same as in case of regular Range-dataset – the only difference is emptiness of the *Dataset* property.

When this property is not empty, XL Report has all the metadata needed to create a report. If it is empty, it is you who must supply XL Report with data and their structure: the number of fields, their names, and their types. There are three events that will help you to provide this.

```
property OnGetDataSourceInfo: TxlGetDataSourceInfo;
property OnGetFieldInfo: TxlGetFieldInfo;
```

```

property OnGetRecord: TxlGetRecord;

type
  TxlGetDataSourceInfo = procedure (DataSource: TxlDataSource;
    var FieldCount: integer) of object;
  TxlGetFieldInfo = procedure (DataSource: TxlDataSource;
    const FieldIndex: integer;
    var FieldName: string;
    var FieldType: TxlDataType) of object;
  TxlGetRecord = procedure (DataSource: TxlDataSource; const RecNo: integer;
    var Values: OLEVariant; var EOF: boolean) of object;

```

At first, XL Report triggers the *OnGetDataSourceInfo* event, allowing you to define the number of fields. In our example it was:

```

procedure TfrmUDData2.xlReport1DataSources0GetDataSourceInfo(
  DataSource: TxlDataSource; var FieldCount: Integer);
begin
  FieldCount := strGrid.ColCount;
end;

```

Then you supply XL Report with field names and types:

```

procedure TfrmUDData2.xlReport1DataSources0GetFieldInfo(DataSource: TxlDataSource;
  const FieldIndex: Integer; var FieldName: String; var FieldType: TxlDataType);
begin
  FieldName := strGrid.Cells[FieldIndex, 0];
  case FieldIndex of
    0: FieldType := xdString;
    1: FieldType := xdInteger;
    2: FieldType := xdFloat;
    3: FieldType := xdDateTime;
  end;
end;

```

This event handler is called *FieldCount* times. *FieldIndex* starts from 0. XL Report follows Excel, so the *FieldType* accepts only following values:

```

type
  TxlDataType =
    (xdNotSupported, xdInteger, xdBoolean, xdFloat, xdDateTime, xdString);

```

At last, you supply XL Report with your data in *OnGetRecord* event handler.

```

procedure TfrmUDData2.xlReport1DataSources0GetRecord(DataSource: TxlDataSource;
  const RecNo: Integer; var Values: OleVariant; var EOF: Boolean);
begin
  EOF := RecNo > (strGrid.RowCount - 1); // exclude fixed row
  if not EOF then begin
    Values[0] := strGrid.Cells[0, RecNo];
    Values[1] := StrToInt(strGrid.Cells[1, RecNo]);
    Values[2] := StrToFloat(strGrid.Cells[2, RecNo]);
    Values[3] := StrToDate(strGrid.Cells[3, RecNo]);
  end;
end;

```

RecNo contains the record number of the current record and starts from 1. If your data comes to the end, you set *EOF* to true. *Values* is a 1-D variant array with the number of items equal to the number of fields, each item initialized with *UnAssigned*.

There is one restriction to the use of unbound data: You cannot use them in master-detail and multiple-sheet reports.

XL Report ProOptionPack™

What is it?

In order to fulfill requirements of our most exacting customers, we created a package of extended options for XL Report. It bases on XLOptionPack technology (as a descendant of TxLOptionPackage) and includes the set of extended options' classes.

Almost all the features were developed at XL Report users' desire. Today this package includes the following features.

Data grouping without subtotals – now you can use the **GROUP** option without summary options (**SUM**, **COUNT**, etc.). The **GROUP** option is now extended with parameters allowing cell merging (much as **MergeLabels** for pivot tables), controlling the placement of the group row, and grouping with no group header. You can use Excel formulas as well as XL Report options in the group header. See the **Group** without subtotals branch in QDemoPro.

Subtotals – all the features of the **GROUP** option are available in subtotalling. Moreover, you are able to space groups out to separate printed sheets, suppress totals, and create a group header in the presence of a concluding row. See the Subtotals branch in QDemoPro.

Direct support of all Excel chart types – XL Report has a possibility of placing static pivot tables in a template. It allows using all the pivot table design features, and speeding up the report generation process. We apply much the same machinery when developing the **CHART** option. Now you place a chart straight in a template and XL Report refreshes it. See the Static Charts branch in QDemoPro.

What follow below is a full list of ProOptionPack options and their parameters:

CHART	Refreshes a given chart with list range data.
SUM, AVG, AVERAGE, COUNT, COUNTNUMS, MAX, MIN, PRODUCT, STDEV, STDEVP, VAR, VARP	Now you can get totals in arbitrary ranges through the use of the \TOTALOFROW parameter..
GROUP (with subtotals)	Available parameters: ASC, DESC, COLLAPSE, MERGELABELS, MERGELABELS2, PLACETOCOLUMN, DISABLEOUTLINE.
GROUP (without subtotals)	Available parameters: ASC, DESC, COLLAPSE, MERGELABELS, MERGELABELS2, PLACETOCOLUMN, WITHHEADER, DISABLEOUTLINE, DISABLESUBTOTALS, PAGEBREAKS.
SUMMARYABOVE, SUMMARYBELOW	Range options. Allow placing the summary row below or above the group header (if any).

GROUPWITHHEADER	Allows using all features available for usual cells straight in the group header.
DELETECOLUMN	Deletes a column of a range
OUTLINE	Allows creating the Outline view for nested ranges.

QDemoPro.dpr

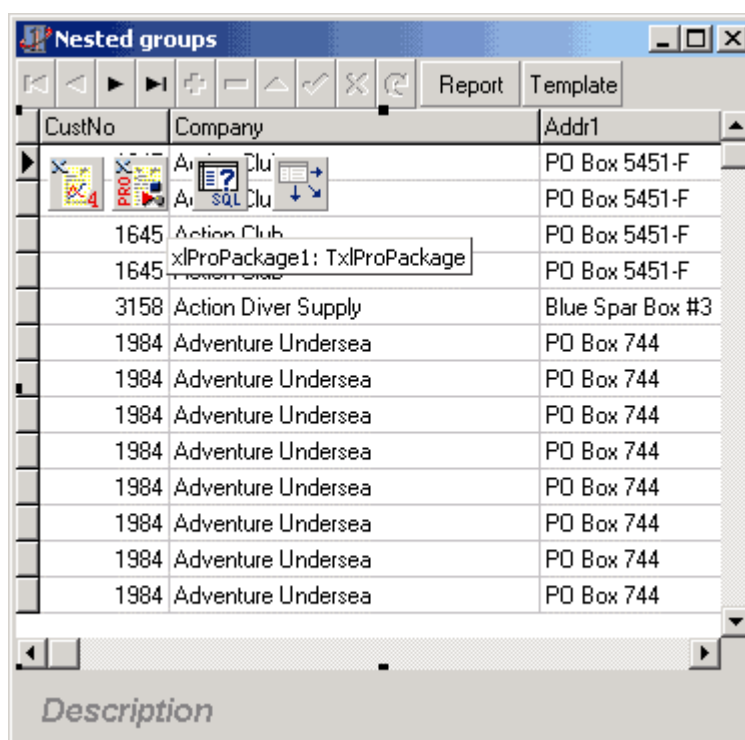
The QDemoPro.dpr demo-project is included in the installation package. It uses DBDEMOS tables as a source of data. It is installed in the <XLReportDir>/QDemo/DelpiPro folder and all the templates (both for XL Report and ProOptionPack) are installed in the <XLReportDir>/QDemo/Templates folder (where <XLReportDir> is the folder where you have XL Report installed).

Using ProOptionPack™

ProOptionPack™ as well as XL Report itself is built using the XLOptionPack™ technology implemented by the classes from xlcOPack unit. All the options included into package are encapsulated into the TxIProPackage component class, which is added to the component palette during installation procedure. In order to access ProOptionPack™ features you have to create at least one instance of the TxIProPackage class. You can choose either placing the component statically on the form or dynamically creating it via the Create method. So, ProOptionPack features will be available while there is at least one instance of the TxIProPackage class.


You can also prefer to create the package dynamically:

```
uses xlProOPack;
var
  xlPackage: TxIProPackage;
begin
  xlPackage := TxIProPackage.Create(Self);
  xlReport4.Report;
  xlPackage.Free;
end;
```




Grouping without subtotals

Excel doesn't have quick grouping features similar to those of the Subtotals method of the *Range* object. ProOptionPack solves this problem allowing using the XL Report **Group** option without any parameters. In this way, for instance:

A	B	C	D	E	F	G	H	I
	Simple data grouping					 ProOptionPack™ ADVANCED XL REPORTS		
	Company	Order No	Payment Method	Ship date	Sale date	Items total	Tax rate	Amount paid
	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?	#NAME?
	group							

This sample template will group the list range by the *Company* column producing all the subtotal attributes without creating subtotals themselves: 1) grouped data will be divided by a group header row containing the grouped value; 2) the outline view will be created (see the figure below).

1	2	A	B	C	D	E	F	G	H	I
			Simple data grouping					 ProOptionPack™ ADVANCED XL REPORTS		
1										
2										
3										
4			Company	Order No	Payment Method	Ship date	Sale date	Items total	Tax rate	Amount paid
5	•		Action Club	1014	Credit	05.26.88	05.25.88	134,85		134,85
6	•		Action Club	1029	MC	07.19.88	07.18.88	20 108,00		20 108,00
7	•		Action Club	1038	Visa	08.27.88	08.26.88	10 152,00		10 152,00
8	•		Action Club	1129	MC	10.19.93	10.19.93	1 004,80		1 004,80
9	•		Action Club							
10	•		Action Diver Supply	1039	Visa	09.01.88	08.29.88	536,80		536,80
11	•		Action Diver Supply							
12	•		Adventure Undersea	1017	Check	06.13.88	06.12.88	10 195,00		
13	•		Adventure Undersea	1037	Credit	08.27.88	08.26.88	3 117,00		3 117,00
14	•		Adventure Undersea	1074	MC	04.20.89	04.19.89	2 195,00		2 195,00
15	•		Adventure Undersea	1099	Credit	06.16.89	06.16.89	859,95		
16	•		Adventure Undersea	1117	Check	04.13.93	04.13.93	6 734,85		
17	•		Adventure Undersea	1137	Credit	11.27.93	11.27.93	6 785,40		6 785,40
18	•		Adventure Undersea	1217	Check	11.22.94	11.22.94	51 730,80		51 730,80
19	•		Adventure Undersea	1294	MC	01.04.95	01.04.95	3 304,85		3 304,85
20	•		Adventure Undersea	1317	Check	02.01.95	02.01.95	7 572,00		7 572,00
21	•		Adventure Undersea							
22	•		American SCUBA Supply	1204	Check	10.18.94	10.18.94	10 263,75		10 263,75
23	•		American SCUBA Supply	1263	Credit	12.14.94	12.14.94	158 922,65		158 922,65
24	•		American SCUBA Supply	1355	Credit	02.05.95	02.05.95	13 908,00		13 908,00
25	•		American SCUBA Supply							
26	•		Aquatic Drama	1065	Visa	03.26.89	03.25.89	17 814,00		17 814,00
27	•		Aquatic Drama							
28	•		Blue Glass Happiness	1042	AmEx	09.25.88	09.24.88	971,70		971,70
29	•		Blue Glass Happiness	1142	AmEx	12.25.93	12.25.93	3 546,00		3 546,00
30	•		Blue Glass Happiness							

Implementation of the **Group** option in ProOptionPack allows several new features available either through the use of the **Group** option parameters or in combination with other options from this package. First, you can place the group header row above grouped data. Just use the **SummaryAbove** range option. Second, you can avoid data repetition in a grouped column and merge the grouped column's cells. This feature is provided through the

MergeLabels parameter of the **Group** option (syntax - "**Group\MergeLabels**"). The result of joint use of the **SummaryAbove** option and **MergeLabels** parameter is shown in the figure below.

3									
4		Company	Order No	Payment Method	Ship date	Sale date	Items total	Tax rate	Amount paid
5	-	Action Club							
6	•		1014	Credit	05.26.88	05.25.88	134,85		134,85
7	•		1029	MC	07.19.88	07.18.88	20 108,00		20 108,00
8	•		1129	MC	10.19.93	10.19.93	1 004,80		1 004,80
9	•		1038	Visa	08.27.88	08.26.88	10 152,00		10 152,00
10	-	Action Diver Supply							
11	•		1039	Visa	09.01.88	08.29.88	536,80		536,80
12	-	Adventure Undersea							
13	•		1117	Check	04.13.93	04.13.93	6 734,85		
14	•		1317	Check	02.01.95	02.01.95	7 572,00		7 572,00
15	•		1217	Check	11.22.94	11.22.94	51 730,80		51 730,80
16	•		1017	Check	06.13.88	06.12.88	10 195,00		
17	•		1037	Credit	08.27.88	08.26.88	3 117,00		3 117,00
18	•		1137	Credit	11.27.93	11.27.93	6 785,40		6 785,40
19	•		1099	Credit	06.16.89	06.16.89	859,95		
20	•		1294	MC	01.04.95	01.04.95	3 304,85		3 304,85
21	•		1074	MC	04.20.89	04.19.89	2 195,00		2 195,00
22	-	American SCUBA Supply							
23	•		1204	Check	10.18.94	10.18.94	10 263,75		10 263,75
24	•		1263	Credit	12.14.94	12.14.94	158 922,65		158 922,65
25	•		1355	Credit	02.05.95	02.05.95	13 908,00		13 908,00
26	-	Aquatic Drama							
27	•		1065	Visa	03.26.89	03.25.89	17 814,00		17 814,00
28	-	Blue Glass Happiness							
29	•		1142	AmEx	12.25.93	12.25.93	3 546,00		3 546,00
30	•		1042	AmEx	09.25.88	09.24.88	971,70		971,70
31	-	Blue Jack Aqua Center							


So, the **/MergeLabels** parameter of the **Group** option:

- Inserts the group header row above (in the presence of the **SummaryAbove** option) or below the grouped values
- Clears the values in the grouped cells'
- Merges the cleared cells
- Form the outline view

There is another variant of merging cells. It is implemented in the **/MergeLabels2** parameter of the **Group** option. This parameter doesn't insert the group header row and doesn't clear the grouped cells. It merges group cells preserving the group value. The sample report follows below. Note that absence of the outline view grouping symbols - Excel is not capable of creating the outline view if there is no header row.

3							
4		Company	Order No	Payment Method	Ship date	Sale date	Items total
5		Action Club	1014	Credit	05.26.88	05.25.88	134,85
6			1029	MC	07.19.88	07.18.88	20 108,00
7			1038	Visa	08.27.88	08.26.88	10 152,00
8			1129	MC	10.19.93	10.19.93	1 004,80
9		Action Diver Supply	1039	Visa	09.01.88	08.29.88	536,80
10		Adventure Undersea	1017	Check	06.13.88	06.12.88	10 195,00
11			1037	Credit	08.27.88	08.26.88	3 117,00
12			1074	MC	04.20.89	04.19.89	2 195,00
13			1099	Credit	06.16.89	06.16.89	859,95
14			1117	Check	04.13.93	04.13.93	6 734,85
15			1137	Credit	11.27.93	11.27.93	6 785,40
16			1217	Check	11.22.94	11.22.94	51 730,80
17			1294	MC	01.04.95	01.04.95	3 304,85
18			1317	Check	02.01.95	02.01.95	7 572,00
19		American SCUBA Supply	1204	Check	10.18.94	10.18.94	10 263,75
20			1263	Credit	12.14.94	12.14.94	158 922,65
21			1355	Credit	02.05.95	02.05.95	13 908,00
22		Aquatic Drama	1065	Visa	03.26.89	03.25.89	17 814,00
23		Blue Glass Happiness	1042	AmEx	09.25.88	09.24.88	971,70
24			1142	AmEx	12.25.93	12.25.93	3 546,00
25		Blue Jack Aqua Center	1006	Visa	11.07.88	11.06.94	31 987,00
26			1079	Credit	05.04.89	05.03.89	4 445,00
27			1106	Visa	09.23.92	09.23.92	3 531,80
28			1153	Credit	04.16.94	04.16.94	3 860,85
29			1253	Credit	11.26.94	11.26.94	4 774,85

Another feature of the **Group** option allows deleting the group column preserving group header row. You place the group value to any other column of the range (the **PlaceToColumn=n** parameter of the **Group** option) and delete the group column (the **DeleteColumn** column option). The sample template in the figure below demonstrates the use of this feature.

B6		=	group\placetocolumn=2;deletecolumn						
A	B	C	D	E	F	G	H	I	
			<div>ProOptionPackTM ADVANCED XL REPORTS</div> <div>Delete column</div>						
				</					

This template contains the **SummaryAbove** range option in the A6 cell. The resulting report is shown below.

	3							
	4	Order No	Payment Method	Ship date	Sale date	Items total	Tax rate	Amount paid
-	5	Action Club						
.	6	1014	Credit	05.26.88	05.25.88	134,85		134,85
.	7	1029	MC	07.19.88	07.18.88	20 108,00		20 108,00
.	8	1129	MC	10.19.93	10.19.93	1 004,80		1 004,80
.	9	1038	Visa	08.27.88	08.26.88	10 152,00		10 152,00
-	10	Action Diver Supply						
.	11	1039	Visa	09.01.88	08.29.88	536,80		536,80
-	12	Adventure Undersea						
.	13	1117	Check	04.13.93	04.13.93	6 734,85		
.	14	1317	Check	02.01.95	02.01.95	7 572,00		7 572,00
.	15	1217	Check	11.22.94	11.22.94	51 730,80		51 730,80
.	16	1017	Check	06.13.88	06.12.88	10 195,00		
.	17	1037	Credit	08.27.88	08.26.88	3 117,00		3 117,00
.	18	1137	Credit	11.27.93	11.27.93	6 785,40		6 785,40
.	19	1099	Credit	06.16.89	06.16.89	859,95		
.	20	1294	MC	01.04.95	01.04.95	3 304,85		3 304,85
.	21	1074	MC	04.20.89	04.19.89	2 195,00		2 195,00
-	22	American SCUBA Supply						
.	23	1204	Check	10.18.94	10.18.94	10 263,75		10 263,75
.	24	1263	Credit	12.14.94	12.14.94	158 922,65		158 922,65
.	25	1355	Credit	02.05.95	02.05.95	13 908,00		13 908,00
-	26	Aquatic Drama						
.	27	1065	Visa	03.26.89	03.25.89	17 814,00		17 814,00

You can see that the *Company* column is deleted because its option cell contained the **DeleteColumn** option. The group value is placed to the *OrderNo* column thus preserving the header row from deleting. This is accomplished via the **PlaceToColumn=2** parameter of the **Group** option. 2 - means the second column of the source range.

You can create several groupings for the same range. In such a case groupings will be created in the left-to-right order. The **DisableOutline** parameter prevents the outline view for a given group. The figure below shows the report whose template contained two groupings - on the *Company* and *Payment Method* columns. The second **Group** option had the **DisableOutline** parameter.

	4	Company	Payment Method	Order No	Ship date	Sale date	Items total	Tax rate	Amount paid
-	77	Catamaran Dive Club							
.	78		Visa						
.	79			1205	10.21.94	10.21.94	4 029,55		4 029,55
.	80			1066	03.27.89	03.26.89	19 812,00		19 812,00
.	81			1166	07.01.94	07.01.94	28 862,00		28 862,00
-	82	Cayman Divers World Unlimited							
.	83		Check						
.	84			1104	07.18.92	07.18.92	51 673,15		51 673,15
.	85		Visa						
.	86			1292	01.01.95	01.01.95	7 986,90		7 986,90
-	87	Central Underwater Supplies							
.	88		Check						
.	89			1134	11.14.93	11.14.93	6 675,95		6 675,95

In addition to the samples, described above, you will find several more samples in QDemoPro.dpr demonstrating data grouping without subtotals. Please, pay attention to those demonstrating the use of conditional formatting and formulas in the option row.

Advanced subtotals

There are certain differences between groupings with and without subtotals. First - a subtotaled group doesn't have a group header row but they have a summary row instead. Second, the above listed parameters work in a slightly different way. We describe the same features anew plus we describe additional range and column options that affect subtotals.

	4	Company	Payment Method	Order No	Ship date	Sale date	Items total	Amount paid
-	5	Grand Total					2 922 666	2 633 460
-	6	Action Club Total					31 400	31 400
•	7	Action Club	Credit	1014	05.26.88	05.25.88	134,85	134,85
•	8	Action Club	MC	1029	07.19.88	07.18.88	20 108,00	20 108,00
•	9	Action Club	MC	1129	10.19.93	10.19.93	1 004,80	1 004,80
•	10	Action Club	Visa	1038	08.27.88	08.26.88	10 152,00	10 152,00
-	11	Action Diver Supply Total					537	537
•	12	Action Diver Supply	Visa	1039	09.01.88	08.29.88	536,80	536,80
-	13	Adventure Undersea Total					92 495	74 705
•	14	Adventure Undersea	Check	1117	04.13.93	04.13.93	6 734,85	
•	15	Adventure Undersea	Check	1317	02.01.95	02.01.95	7 572,00	7 572,00
•	16	Adventure Undersea	Check	1217	11.22.94	11.22.94	51 730,80	51 730,80
•	17	Adventure Undersea	Check	1017	06.13.88	06.12.88	10 195,00	
•	18	Adventure Undersea	Credit	1037	08.27.88	08.26.88	3 117,00	3 117,00
•	19	Adventure Undersea	Credit	1137	11.27.93	11.27.93	6 785,40	6 785,40
•	20	Adventure Undersea	Credit	1099	06.16.89	06.16.89	859,95	
•	21	Adventure Undersea	MC	1294	01.04.95	01.04.95	3 304,85	3 304,85
•	22	Adventure Undersea	MC	1074	04.20.89	04.19.89	2 195,00	2 195,00
-	23	American SCUBA Supply Total					183 094	183 094
•	24	American SCUBA Supply	Check	1204	10.18.94	10.18.94	10 263,75	10 263,75
•	25	American SCUBA Supply	Credit	1263	12.14.94	12.14.94	158 922,65	158 922,65
•	26	American SCUBA Supply	Credit	1355	02.05.95	02.05.95	13 908,00	13 908,00

Just as in the case of grouping without subtotals, the **SummaryAbove** range option places the summary row above grouped values. The figure above shows the sample report that uses this option. Take note of grand totals that are also placed above the range data.

You can suppress the grand totals creation using the **DisableGrandTotals** range option - see the sample report below.

Company	Payment Method	Order No	Ship date	Sale date	Items total	Amount paid
Action Club Total					31 400	31 400
Action Club	Credit	1014	05.26.88	05.25.88	134,85	134,85
Action Club	MC	1029	07.19.88	07.18.88	20 108,00	20 108,00
Action Club	MC	1129	10.19.93	10.19.93	1 004,80	1 004,80
Action Club	Visa	1038	08.27.88	08.26.88	10 152,00	10 152,00
Action Diver Supply Total					537	537
Action Diver Supply	Visa	1039	09.01.88	08.29.88	536,80	536,80

In the case of several different subtotals for the same column, say sum and average, Excel creates several different grand totals (naturally) but displays them in a somewhat peculiar way. This effect lasts out both in programming subtotals via VBA and in creating them manually. So, if your understanding of 'peculiarity' is close to ours, you will avoid using the **SummaryAbove** option in this case. In case of several groupings in the same range, the **SummaryAbove** option can lead to even more peculiar display of data. That's why we don't recommend using this option in such a case. We tried hard to solve as many potential problems as we could but

we'd recommend avoiding using the **SummaryAbove** option together with subtotals, especially if you plan to work with Excel97.

Another feature is creating page breaks on the fly. If you need to place every group at separate pages, use the **/PageBreaks** parameter to the **Group** option.

	Company	Payment Method	Order No.	Ship date	Sale date	Items Total	Amount paid
5	Action Club	Check	1014	05.20.00	05.20.00	194.25	194.25
6		Check Total				194.25	194.25
7	Action Club	M/C	1029	07.19.00	07.19.00	20 100.00	20 100.00
8	Action Club	M/C	1129	10.19.00	10.19.00	1 004.30	1 004.30
9		M/C Total				21 004.30	21 004.30
10	Action Club	Void	1000	03.27.00	03.28.00	10 152.00	10 152.00
11		Void Total				10 152.00	10 152.00
12	Action Club Total					31 350.55	31 350.55
13	Action Dive Supply	Void	1000	03.01.00	03.29.00	508.30	508.30
14		Void Total				508.30	508.30
15	Action Dive Supply Total					508.30	508.30
16	Adventure Unlimited	Check	1117	04.19.00	04.19.00	8 794.25	8 794.25
17	Adventure Unlimited	Check	1217	02.01.00	02.01.00	7 572.00	7 572.00
18	Adventure Unlimited	Check	1217	11.22.04	11.22.04	51 190.30	51 190.30
19	Adventure Unlimited	Check	1017	08.19.00	08.19.00	10 125.00	10 125.00
20		Check Total				77 681.55	77 681.55
21	Adventure Unlimited	Check	1007	03.07.00	03.28.00	9 117.00	9 117.00
22	Adventure Unlimited	Check	1197	11.27.00	11.27.00	8 725.40	8 725.40
23	Adventure Unlimited	Check	1009	08.18.00	08.18.00	350.35	350.35
24		Check Total				26 202.75	26 202.75
25	Adventure Unlimited	M/C	1294	01.04.00	01.04.00	9 904.25	9 904.25
26	Adventure Unlimited	M/C	1074	04.20.00	04.19.00	2 125.00	2 125.00
27		M/C Total				12 029.25	12 029.25
28	Adventure Unlimited Total					109 913.55	109 913.55
29	American SCUBA Supply	Check	1204	10.13.04	10.13.04	10 289.75	10 289.75
30		Check Total				10 289.75	10 289.75
31	American SCUBA Supply	Check	1209	12.14.04	12.14.04	150 922.00	150 922.00
32	American SCUBA Supply	Check	1205	02.05.00	02.05.00	19 903.00	19 903.00
33		Check Total				172 114.75	172 114.75
34	American SCUBA Supply Total					192 406.50	192 406.50
35	Aquatic Drama	Void	1000	03.28.00	03.28.00	17 814.00	17 814.00
36		Void Total				17 814.00	17 814.00
37	Aquatic Drama Total					17 814.00	17 814.00

The **/MergeLabels** parameter merge the grouped cells but, contrary to grouping without subtotals, keeps the group value in the resulting cell. Sample:

			Company	Payment Method	Order No	Ship date	Sale date	Items total	Amount paid
		4							
	•	5	Action Club	Credit	1014	05.26.88	05.25.88	134,85	134,85
	•	6		MC	1029	07.19.88	07.18.88	20 108,00	20 108,00
	•	7		MC	1129	10.19.93	10.19.93	1 004,80	1 004,80
	•	8		Visa	1038	08.27.88	08.26.88	10 152,00	10 152,00
	-	9						31 399,65	31 399,65
	•	10	Action Diver Supply	Visa	1039	09.01.88	08.29.88	536,80	536,80
	-	11						536,80	536,80
	•	12	Adventure Undersea	Check	1117	04.13.93	04.13.93	6 734,85	
	•	13		Check	1317	02.01.95	02.01.95	7 572,00	7 572,00
	•	14		Check	1217	11.22.94	11.22.94	51 730,80	51 730,80
	•	15		Check	1017	06.13.88	06.12.88	10 195,00	
	•	16		Credit	1037	08.27.88	08.26.88	3 117,00	3 117,00
	•	17		Credit	1137	11.27.93	11.27.93	6 785,40	6 785,40
	•	18		Credit	1099	06.16.89	06.16.89	859,95	
	•	19		MC	1294	01.04.95	01.04.95	3 304,85	3 304,85
	•	20		MC	1074	04.20.89	04.19.89	2 195,00	2 195,00
	-	21						92 494,85	74 705,05

Page 60

		Company	Payment Method	Order No	Ship date	Sale date	Items total	Amount paid
4								
5		Action Club						
6			Credit	1014	05.26.88	05.25.88	134,85	134,85
7			MC	1029	07.19.88	07.18.88	20 108,00	20 108,00
8			MC	1129	10.19.93	10.19.93	1 004,80	1 004,80
9			Visa	1038	08.27.88	08.26.88	10 152,00	10 152,00
10		Action Club Total						31 399,65
11		Action Diver Supply						
12			Visa	1039	09.01.88	08.29.88	536,80	536,80
13		Action Diver Supply Total						536,80
14		Adventure Undersea						
15			Check	1117	04.13.93	04.13.93	6 734,85	
16			Check	1317	02.01.95	02.01.95	7 572,00	7 572,00
17			Check	1217	11.22.94	11.22.94	51 730,80	51 730,80
18			Check	1017	06.13.88	06.12.88	10 195,00	
19			Credit	1037	08.27.88	08.26.88	3 117,00	3 117,00
20			Credit	1137	11.27.93	11.27.93	6 785,40	6 785,40
21			Credit	1099	06.16.89	06.16.89	859,95	
22			MC	1294	01.04.95	01.04.95	3 304,85	3 304,85
23			MC	1074	04.20.89	04.19.89	2 195,00	2 195,00
24		Adventure Undersea Total					17 789,80	92 494,85
								74 705,05

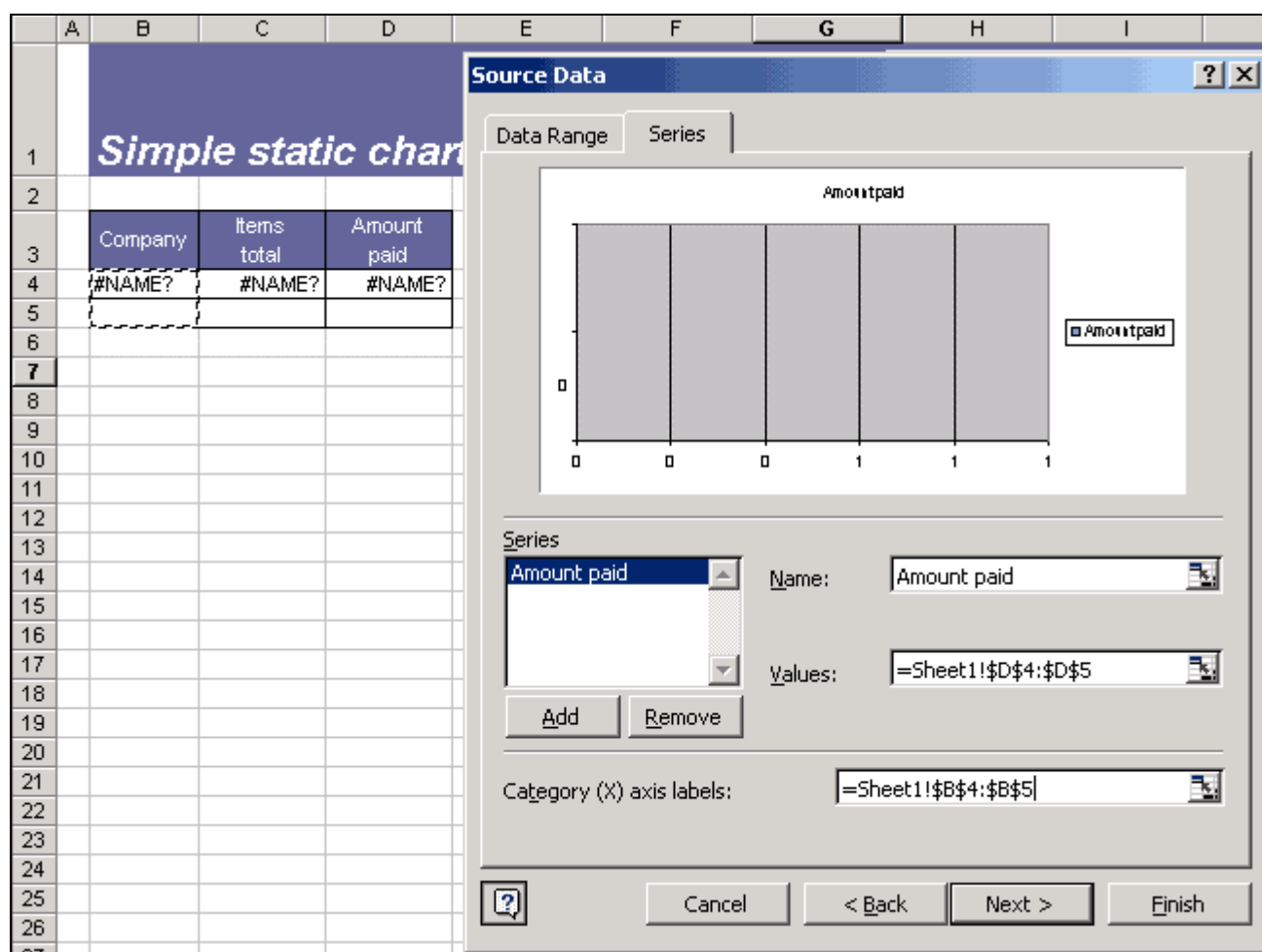
Static charts

One of the main features of ProOptionPack is the possibility to use Excel charts referencing XL Report ranges. This feature is available through the use of the **Chart** option.

Company	Items total	Amount paid
#NAME?	#NAME?	#NAME?

For instance, you had created the *OrdersRange* shown at the figure at your left. Now you invoke Chart Wizard (Insert|Chart...), choose the *Bar* type, leave the *Data Range* box empty, click the *Series* tab and create a new series named *Amount paid*, that takes its values from the *Amount paid*

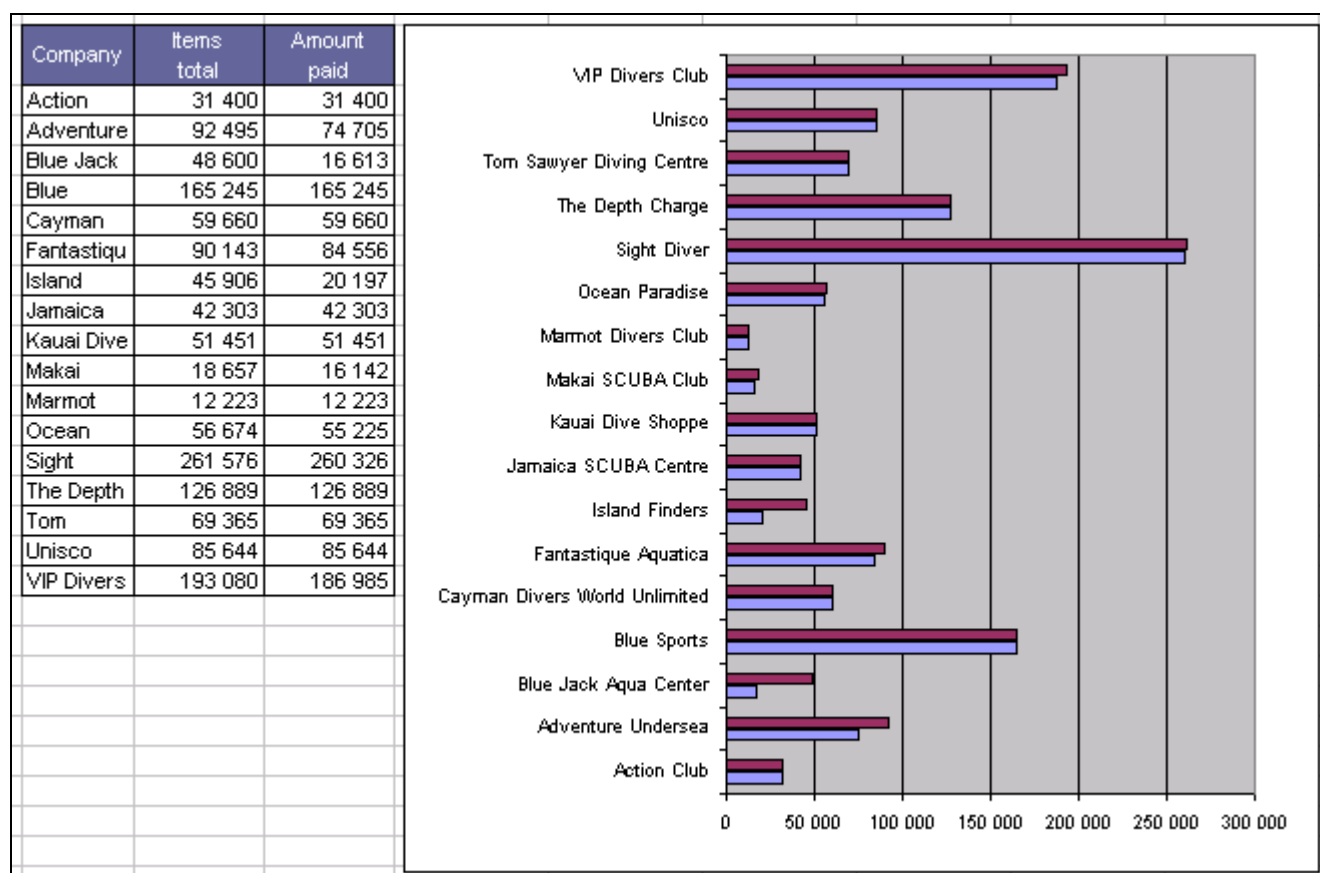
column and labels from the *Company* column.



Do you see that both ranges include the option row? To create another bar (Items total) you add another series, name it and make it take its data from the C4:C5 range. Finish the wizard and place the chart any where on the current worksheet.

In order to let the **Chart** option to work correctly you should give the name to the chart. Press the Ctrl key and click the chart (Ctrl+Click). Then you can change the default chart name in the Name box at the left of the Formula bar. We preferred to name it *SimpleChart*. Nice enough, isn't it? ☺

To refresh the chart with real data you have to apply the **Chart** option to the range: 'Chart\Name=Sheet1!SimpleChart'. Note that we used the full name of the chart, that is, including the sheet name. If you place a chart on a separate sheet, you can put only the name of the sheet in the \Name parameter. And here is the chart with real data:



Microsoft Excel - serious things

We understand that many Delphi developers don't know Microsoft Excel thoroughly. Many of you will run into problems when you start with Excel. We recommend that you read the documents and manuals, especially on Excel formulas, absolute and relative referencing, conditional formatting, filtering, sorting and grouping and, of course, VBA. Write us at support@xl-report.com.

End

There are a lot of other ways to use XL Report. Look at XL Report as at just a simple way to transfer your data to Microsoft Excel. How you will use it in your case you will have to decide for yourself. Good luck!

Appendices

Appendix A: Guidelines for creating a list on a worksheet from Microsoft

Microsoft Excel has a number of features that make it easy to manage and analyze data in a list. To take advantage of these features, enter data in a list according to the following guidelines.

List organization

Use only one list per worksheet Avoid having more than one list on a worksheet. Some list management features, such as filtering, can be used on only one list at a time.

Put similar items in one column Design the list so that all rows have similar items in the same column.

Keep the list separate Leave at least one blank column and one blank row between the list and other data on the worksheet. Excel can then more easily detect and select the list when you sort, filter, or insert automatic subtotals.

Position critical data above or below the list Avoid placing critical data to the left or right of the list; the data might be hidden when you filter the list.

Show rows and columns Make sure any hidden rows or columns are displayed before making changes to the list. When rows and columns in a list are not showing, data can be inadvertently deleted.

List format

Use formatted column labels Create column labels in the first row of the list. Excel uses the labels to create reports and to find and organize data. Use a font, alignment, format, pattern, border, or capitalization style for column labels that is different from the format you assign to the data in the list. Format the cells as text before you type the column labels.

Use cell borders When you want to separate labels from data, use cell borders — not blank rows or dashed lines — to insert lines below the labels.

Avoid blank rows and columns Avoid putting blank rows and columns in the list so that Excel can more easily detect and select the list.

Don't type leading or trailing spaces Extra spaces at the beginning or end of a cell affect sorting and searching. Instead of typing spaces, indent the text within the cell.

Extend list formats and formulas When you add new rows of data to the end of a list, Excel uses consistent formatting and formulas. Three of the five preceding cells must use the same format or formula for this to occur.

Appendix B: Guidelines for naming cells, formulas, and constants in Microsoft Excel

- The first character of a name must be a letter or an underscore character. Remaining characters in the name can be letters, numbers, periods, and underscore characters.
- Names cannot be the same as a cell reference, such as Z\$100 or R1C1.
- Spaces are not allowed. Underscore characters and periods may be used as word separators for example, First.Quarter or Sales_Tax.
- A name can contain up to 255 characters.
- Names can contain uppercase and lowercase letters. Microsoft Excel does not distinguish between uppercase and lowercase characters in names. For example, if you have created the name Sales and then create another name called SALES in the same workbook, the second name will replace the first one.

Appendix C: Standard options

Option	Class	Parameters	xlObjects	RangeType	Description
Unit: xlStdOPack.pas					
OnlyValues	TopOnlyValues		[xoWorkbook, xoWorksheet, xoRange, xoColumn]	[rtRange, rtRangeRoot]	Replaces formulas with their values.
AutoSafe	TopAutoSafe		[xoWorkbook, xoWorksheet]		If used in a workbook, protects the workbook and all of its worksheets. If used in a worksheet, protects the worksheet. Analogous to "Tools Protection Protect Sheet..." and "Tools Protection Protect Workbook..."
RowsFit	TopRowsFit		[xoWorkbook, xoWorksheet, xoRange]	[rtRange, rtRangeRoot]	Makes the row height fit the cell's contents.
ColsFit	TopColsFit		[xoWorkbook, xoWorksheet, xoRange]	[rtRange, rtRangeRoot]	Makes the column width fit the cell's contents.
SheetHide, Hide	TopSheetHide, TopSheetHide2		[xoWorksheet]		Hides the worksheet. If Debug is on, provides "soft" hiding that allows unhiding the worksheet through "Format Sheet Unhide..."
AutoScale	TopAutoScale		[xoWorkbook, xoWorksheet]		If used in a workbook, protects the workbook and all of its worksheets. If used in a worksheet, protects the worksheet. Analogous to "Tools Protection Protect Sheet..." and "Tools Protection Protect Workbook..."
AutoFilter	TopAutoFilter		[xoRange]	[rtRange]	Turn on the Autofilter in the range.
Sort	TopSort	[/Desc] [/ Asc]	[xoColumn]	[rtRange, rtRangeDetail]	Provides sorting for the column. The sort order is either Asc (default) or Desc. Up to three columns can be sorted at once. Columns for sorting are taken in right-to-left order.
Desc	TopDesc		[xoColumn]	[rtRange, rtRangeDetail]	The same as Sort/Desc
Asc	TopAsc		[xoColumn]	[rtRange, rtRangeDetail]	The same as Sort/Asc
Sum Avg or Average Count	TopSum TopAverage, TopAvearge2		[xoColumn]	[rtRange, rtRangeRoot, rtRangeMaster,	Summary options are used in totals and subtotals, and pivot tables. You can get only one total for a column. If you indicate more than one summary option only the last one is used. Totals are inserted into the

Option	Class	Parameters	xlObjects	RangeType	Description
CountNums Min Max Product StDev StDevP Var VarP	TopCount TopCountNums TopMin TopMax TopProduct TopStDev TopStDevP TopVar TopVarP			rtRangeDetail]	option row
Unit: xlPivotOPack.pas					
Group	TopGroup	[/Asc] [/Desc] [/Collapse]	[xoColumn]	[rtRange]	Groups the data in the column where it is used, produces subtotals for columns with any of summary options used, and outlines the range. The range is sorted previously by the columns containing any of the following options: Group, Sort, Desc, and Asc (See also: GroupNoSort and summary options). The sort order is determined by specifying the Desc or Asc parameter (default – Asc). If several Group options are used in a range, then the data are grouped in right-to-left order creating several levels in an outline. The Collapse parameter hides the detail data in the outline up to the level containing the Group option with this parameter. Subtotal rows are formatted according to the option row formatting.
GroupNoSort	TopGroupNoSort		[xoRange]	[rtRange]	Prevents the range to be sorted. Can be used in case of previously sorted data. Speeds up the report creation.
Pivot	TopPivot	/Name=name1 [/Dst=dst1] [/DataToRows] [/RowGrand] [/ColumnGrand] [/MergeLabels] [/NoPreserveFormatting] [/Refresh=PivotTableList]	[xoRange]	[rtRange]	Produces a pivot table basing on the range. In order to describe the structure of the pivot table the pivot description options are used: Page, Row, Column, and Data. These options allow specifying destinations (the pivot table areas) for your data. The fields that are not designated to any of the pivot table's areas are included into the pivot table as hidden ones. The end user is allowed to change the pivot table structure. The Name parameter is required. The name must be allowable in Excel. The Dst parameter allows specifying the placement of the pivot table. You can specify both the worksheet name for the pivot table to be placed on and the top left corner of the pivot table (page fields included). If the worksheet name is omitted, a new worksheet named after the pivot table is inserted. The placement of the pivot table's top left corner can be given in either A1 or R1C1 style. Examples: Dst=Sheet1!D8; Dst=D8. In case of several data fields, the DataToRows parameter places data

Option	Class	Parameters	xlObjects	RangeType	Description
					<p>field labels in rows (they are placed in columns by default). The RowGrand and ColumnGrand parameters show grand totals for rows and columns respectively.</p> <p>The MergeLabels parameter allows using merged cells in outer-row item, column item, subtotal, and grand total labels.</p> <p>The NoPreserveFormatting parameter prevents the source range formats to be applied to the pivot table. Use it in order to speed up the report.</p> <p>The Refresh parameter allows specifying the list of static pivot tables connected with the range. The pivot table names must be preceded with the name of the worksheet containing the pivot table (WorksheetName!PivotTable) and separated with semicolon. If either the worksheet name or pivot table name contain spaces, it shouldn't be enclosed in single quotation marks.</p> <p>The pivot table technical parameters and limitations can be found in MSDN.</p>
Page	TopPage		[xoColumn]	[rtRange]	Produces a pivot table field basing on the column and places it into the Page Area of the pivot table. The name of the field is the label of the column.
Row	TopRow		[xoColumn]	[rtRange]	<p>Produces a pivot table field basing on the column and places it into the Row Area of the pivot table. The name of the field is the label of the column.</p> <p>Excel provides grouping of inner-row items for every outer-row item and inserts subtotals if necessary. The type of subtotals is determined by specifying an additional summary option(s) along with the Row option of the source column. For example, Row;Sum;Count options given in a column produce sum and count subtotals for the corresponding pivot table field.</p>
Column	TopColumn		[xoColumn]	[rtRange]	<p>Produces a pivot table field basing on the column and places it into the Column Area of the pivot table. The name of the field is the label of the column.</p> <p>You can insert one or more subtotals for a pivot table field. The type of subtotals is determined by specifying an additional summary option(s) along with the Column option of the source column. For example, Column;Sum;Count options given in a column produce sum and count subtotals for the corresponding pivot table field.</p>
Data	TopData		[xoColumn]	[rtRange]	Produces a pivot table field basing on the column and places it into the Data Area of the pivot table. The name of the field is the label of the

Option	Class	Parameters	xlObjects	RangeType	Description
					column. The field label are placed in columns (see also the DataToRows parameter of the Pivot option) Data fields are summed by default. In order to apply any different summary function, add an appropriate summary option to the Data option.

Appendix D: ProOptionPack options

Option	Class	Parameters	xlObjects	RangeType	Description
Unit: xlProOPack.pas					
Sum	<i>TopSumEx</i>	<i>TotalOfRow=row</i>	<i>[xoColumn]</i>	<i>[rtRange, rtRangeRoot, rtRangeMaster, rtRangeDetail]</i>	Extended summary options.
Avg или Average	<i>TopAverageEx, TopAvearge2Ex</i>				Extends standard option functionality allowing totaling in arbitrary ranges. This feature is accessed through the use of the TotalOfRow parameter that accepts the row for which you want to get a summary.
Count	<i>TopCountEx</i>				
CountNums	<i>TopCountNumsEx</i>				
Min	<i>TopMinEx</i>				
Max	<i>TopMaxEx</i>				
Product	<i>TopProductEx</i>				
StDev	<i>TopStDevEx</i>				
StDevP	<i>TopStDevPEX</i>				
Var	<i>TopVarEx</i>				
VarP	<i>TopVarPEX</i>				
Chart	<i>TopChart</i>	<i>Name=n</i>	<i>[xoRange]</i>	<i>[rtRange]</i>	Obligatory. Refreshes the chart specified in n with data from the range. N must contain the full name of a chart, including the sheet name.
DeleteColumn	<i>TopDeleteColumn</i>		<i>[xoColumn]</i>	<i>[rtRange, rtRangeRoot]</i>	Deletes the column where the option is used.

Group	<i>TopGroupEx</i>	<i>Collapse</i> <i>Desc</i> <i>Asc</i> <i>MergeLabels</i> <i>MergeLabels2</i> <i>PlaceToColumn=n</i> <i>WithHeader</i> <i>Disablesubtotals</i> <i>DisableOutline</i> <i>PageBreaks</i>	<i>[xoColumn]</i>	<i>[rtRange, rtRangeDetail]</i>	<p>Extends the functionality of the standard Group option. The range where the option is used must be a list (see Appendix A).</p> <p>Allows creating subtotals in the ranges of the lowest level of nesting in master-detail reports.</p> <p>Can be used in absence of the summary options. In such a case, data are grouped without subtotals. The option row formatting is used for formatting subtotals and group headers.</p> <p>In the presence of the SummaryAbove option (see below), subtotals are placed above grouped values.</p> <p>The Collapse, Asc, and Desc parameters are the same as in the standard option pack.</p> <p>MergeLabels and MergeLabels2 cause merging of cells in the grouped column.</p> <p>PlaceToColumn allows placing the group header to a specified column.</p> <p>DisableSubtotals turns off subtotals for the grouped column. Useful in report debugging. Sorts data within the group.</p> <p>DisableOutline turns off an outline for the grouped column.</p> <p>PageBreaks inserts page breaks between groups.</p> <p>WithHeader creates a header in case of subtotals.</p>
GroupNoSort	<i>TopGroupNoSort</i>		<i>[xoRange]</i>	<i>[rtRange, rtRangeDetail]</i>	The same as the standard option but can be used with extended Group option. The range where the option is used must be a list (see Appendix A).
DisableGrandTotals	<i>TopDisableGrandTotals</i>		<i>[xoRange]</i>	<i>[rtRange]</i>	Suppresses grand totals in the case of grouping with subtotals.
SummaryAbove SummaryBelow	<i>TopSummaryAbove</i> <i>TopSummaryBelow</i>		<i>[xoRange]</i>	<i>[rtRange, rtRangeDetail]</i>	<p>Additional options to be used together with the Group option. SummaryAbove allows placing summary above grouped values. SummaryBelow (default) places summary below data.</p> <p>The range where the option is used must be a list (see Appendix A).</p>
GroupWithHeader	<i>TopGroupWithHeader</i>		<i>[xoRange]</i>	<i>[rtRange]</i>	Creates a group header for a subtotaled range. Works with range of a special format only.
Outline	<i>TopOutline</i>	<i>CollapseLevel=level</i>	<i>[xoRange]</i>	<i>[rtRangeRoot]</i>	<p>Allows creating the Outline view corresponding to the structure of nested ranges. Is used at the topmost level of nesting. Can be used together with the SummaryAbove and SummaryBelow options. Outline doesn't cover subtotals. Clears the outline created by the Group option.</p> <p>Use the CollapseLevel parameter in order to show only a specific level in an outline.</p>