

SYBASE®

DataWindow Designer User's Guide

DataWindow .NET™

2.0

DOCUMENT ID: DC00043-01-0200-01

LAST REVISED: March 2006

Copyright © 2004-2006 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase, the Sybase logo, ADA Workbench, Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Server Enterprise Monitor, Adaptive Server Enterprise Replication, Adaptive Server Everywhere, Adaptive Warehouse, Afaia, Answers Anywhere, Anywhere Studio, Application Manager, AppModeler, APT Workbench, APT-Build, APT-Edit, APT-Execute, APT-Translator, APT-Library, AvantGo Mobile Delivery, AvantGo Mobile Inspection, AvantGo Mobile Marketing Channel, AvantGo Mobile Pharma, AvantGo Mobile Sales, AvantGo Pylon, AvantGo Pylon Application Server, AvantGo Pylon Conduit, AvantGo Pylon PIM Server, AvantGo Pylon Pro, Backup Server, BizTracker, ClearConnect, Client-Library, Client Services, Convoy/DM, Copernicus, Data Pipeline, Data Workbench, DataArchitect, Database Analyzer, DataExpress, DataServer, DataWindow, DataWindow .NET, DB-Library, dbQueue, Developers Workbench, DirectConnect, DirectConnect Anywhere, Distribution Director, e-ADK, E-Anywhere, e-Biz Impact, e-Biz Integrator, E-Whatever, EC Gateway, ECMAP, ECRTIP, eFulfillment Accelerator, Embedded SQL, EMS, Enterprise Application Studio, Enterprise Client/Server, Enterprise Connect, Enterprise Data Studio, Enterprise Manager, Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, eProcurement Accelerator, EWA, Financial Fusion, Financial Fusion Server, Gateway Manager, GlobalFIX, iAnywhere, iAnywhere Solutions, ImpactNow, Industry Warehouse Studio, InfoMaker, Information Anywhere, Information Everywhere, InformationConnect, InternetBuilder, iScript, Jaguar CTS, jConnect for JDBC, M2M Anywhere, Mach Desktop, Mail Anywhere Studio, Mainframe Connect, Maintenance Express, Manage Anywhere Studio, M-Business Anywhere, M-Business Channel, M-Business Network, M-Business Suite, MDI Access Server, MDI Database Gateway, media.splash, MetaWorks, mFolio, Mirror Activator, MySupport, Net-Gateway, Net-Library, New Era of Networks, ObjectConnect, ObjectCycle, OmniConnect, OmniSQL Access Module, OmniSQL Toolkit, Open Biz, Open Client, Open ClientConnect, Open Client/Server, Open Client/Server Interfaces, Open Gateway, Open Server, Open ServerConnect, Open Solutions, Optima++, PB-Gen, PC APT Execute, PC DB-Net, PC Net Library, Pharma Anywhere, PocketBuilder, Pocket PowerBuilder, Power++, power.stop, PowerAMC, PowerBuilder, PowerBuilder Foundation Class Library, PowerDesigner, PowerDimensions, PowerDynamo, PowerScript, PowerSite, PowerSocket, Powersoft, PowerStage, PowerStudio, PowerTips, Powersoft Portfolio, Powersoft Professional, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, QAnywhere, Rapport, RemoteWare, RepConnector, Replication Agent, Replication Driver, Replication Server, Replication Server Manager, Replication Toolkit, Report-Execute, Report Workbench, Resource Manager, RFID Anywhere, RW-DisplayLib, RW-Library, Sales Anywhere, SDF, Search Anywhere, Secure SQL Server, Secure SQL Toolset, Security Guardian, SKILLS, smart.partners, smart.parts, smart.script, SOA Anywhere, SQL Advantage, SQL Anywhere, SQL Anywhere Studio, SQL Code Checker, SQL Debug, SQL Edit, SQL Edit/TPU, SQL Everywhere, SQL Modeler, SQL Remote, SQL Server, SQL Server Manager, SQL SMART, SQL Toolset, SQL Server/CFT, SQL Server/DBM, SQL Server SNMP SubAgent, SQL Station, SQLJ, STEP, SupportNow, S.W.I.F.T. Message Format Libraries, Sybase Central, Sybase Client/Server Interfaces, Sybase Financial Server, Sybase Gateways, Sybase IQ, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase User Workbench, SybaseWare, Syber Financial, SyberAssist, SybFlex, SyBooks, System 10, System 11, System XI (logo), SystemTools, Tabular Data Stream, TradeForce, Transact-SQL, Translation Toolkit, UltraLite, UltraLite.NET, UNIBOM, Unilib, Uninull, Unisep, Unistring, URK Runtime Kit for UniCode, VisualWriter, VQL, WarehouseArchitect, Warehouse Control Center, Warehouse Studio, Warehouse WORKS, Watcom, Watcom SQL, Watcom SQL Server, Web Deployment Kit, Web.PB, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, XA-Server, XcelleNet, and XP Server are trademarks of Sybase, Inc. 10/05

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names used herein may be trademarks or registered trademarks of their respective companies.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book	xv
-----------------------	----

PART 1 THE DATAWINDOW DESIGNER ENVIRONMENT

CHAPTER 1	Working with DataWindow Designer	3
	About DataWindow Designer	4
	The DataWindow Designer environment.....	4
	About the PowerBar	5
	Working with projects	6
	Working with DataWindow objects	7
	Creating new DataWindow objects	8
	Opening existing DataWindow objects	8
	Running or previewing DataWindow objects	9
	Working in painters	10
	Using views in painters.....	10
	Using pop-up menus	15
	Defining colors.....	16
	Working with tools	17
	Using the file editor.....	17
	Using the To-Do List.....	18
	Setting properties	20
	Using toolbars	21
	Toolbar basics	21
	Drop-down toolbars	21
	Controlling the display of toolbars	22
	Moving toolbars using the mouse.....	23
	Customizing toolbars	23
	Creating new toolbars	28
	Using online Help	29
	Customizing keyboard shortcuts	30
	Changing fonts	31

How your DataWindow Designer environment is managed..... 32
 About the registry 32
 About the initialization file..... 33

CHAPTER 2 Working with Libraries 35
 About libraries 35
 About the Library painter 36
 Working with libraries 37
 Displaying libraries and objects 37
 Using the pop-up menu 38
 Controlling columns that display in the List view 38
 Selecting objects 39
 Filtering the display of objects 39
 Filtering the display of libraries and folders 40
 Opening and previewing objects 40
 Copying, moving, and deleting objects..... 40
 Setting the root..... 42
 Moving back, forward, and up one level..... 42
 Modifying comments 43
 Deleting libraries..... 44
 Optimizing libraries..... 44
 Regenerating library entries 45
 Migrating libraries 45
 Exporting and importing entries 46
 Creating a deployment library (PBD file)..... 48
 Including additional resources in the PBD file 48
 What happens at runtime 49
 Creating a library directory report..... 49

CHAPTER 3 Managing the Database..... 51
 Working with database components 51
 Managing databases 54
 Using the Database painter..... 55
 Modifying database preferences 58
 Logging your work 59
 Creating and deleting an Adaptive Server Anywhere database..... 60
 Working with tables 62
 Creating a new table from scratch..... 62
 Creating a new table from an existing table 63
 Specifying column definitions 64
 Specifying table and column properties 64
 Altering a table 68
 Cutting, copying, and pasting columns..... 70

Closing a table.....	70
Dropping a table	70
Viewing pending SQL changes	71
Printing the table definition	72
Exporting table syntax	72
About system tables	73
Creating and editing Sybase Adaptive Server temporary tables	74
Working with keys	75
Working with indexes	79
Working with database views	81
Manipulating data	86
Retrieving data	86
Modifying data	87
Sorting rows	88
Filtering rows	89
Viewing row information	90
Importing data	90
Printing data	91
Saving data	91
Creating and executing SQL statements	92
Building and executing SQL statements	92
Customizing the editor.....	95
Controlling access to the current database.....	96

PART 2

DATAWINDOWS

CHAPTER 4

Defining DataWindow Objects.....	99
About DataWindow objects	99
DataWindow object examples	100
Choosing a presentation style.....	101
Using the Tabular style.....	102
Using the Freeform style	102
Using the Grid style	103
Using the Label style	103
Using the N-Up style	105
Using the Group style.....	106
Using the Composite style.....	107
Using the Graph and Crosstab styles.....	108
Using the TreeView style.....	108
Building a DataWindow object	109
Selecting a data source.....	110

- Using Quick Select..... 112
 - Selecting a table..... 113
 - Selecting columns..... 114
 - Specifying sorting criteria..... 115
 - Specifying selection criteria..... 115
- Using SQL Select..... 121
 - Selecting tables and views..... 122
 - Selecting columns..... 124
 - Displaying the underlying SQL statement..... 125
 - Joining tables..... 126
 - Using retrieval arguments..... 129
 - Specifying selection, sorting, and grouping criteria..... 130
- Using Query..... 136
- Using External..... 136
- Using Stored Procedure..... 137
- Using ADO DataSet..... 139
- Choosing DataWindow object-wide options..... 141
- Generating and saving a DataWindow object..... 142
 - About the extended attribute system tables and DataWindow objects..... 143
 - Saving the DataWindow object..... 144
 - Modifying an existing DataWindow object..... 144
- Defining queries..... 145
 - Previewing the query..... 146
 - Saving the query..... 146
 - Modifying a query..... 147
- What's next..... 147

CHAPTER 5 Enhancing DataWindow Objects..... 149

- Working in the DataWindow painter..... 150
 - Understanding the DataWindow painter Design view..... 152
 - Using the DataWindow painter toolbars..... 154
 - Using the Properties view in the DataWindow painter..... 155
 - Selecting controls in the DataWindow painter..... 156
 - Resizing bands in the DataWindow painter Design view..... 157
 - Using zoom in the DataWindow painter..... 157
 - Undoing changes in the DataWindow painter..... 158
- Using the Preview view of a DataWindow object..... 158
 - Retrieving data..... 159
 - Modifying data..... 161
 - Viewing row information..... 162
 - Importing data into a DataWindow object..... 163
 - Using print preview..... 163
 - Printing data..... 165

Working in a grid DataWindow object	166
Saving data in an external file	168
Saving the data as PDF	168
Saving the data in HTML Table format.....	169
Working with PSR files	170
Modifying general DataWindow object properties	171
Changing the DataWindow object style.....	171
Setting colors in a DataWindow object.....	172
Specifying properties of a grid DataWindow object.....	173
Specifying pointers for a DataWindow object.....	174
Defining print specifications for a DataWindow object	174
Modifying text in a DataWindow object	178
Defining the tab order in a DataWindow object.....	179
Naming controls in a DataWindow object.....	180
Using borders in a DataWindow object	181
Specifying variable-height bands in a DataWindow object....	182
Modifying the data source of a DataWindow object	183
Storing data in a DataWindow object using the Data view.....	185
What happens at runtime	187
Retrieving data	187
Prompting for retrieval criteria in a DataWindow object	187
Retrieving rows as needed.....	188
Saving retrieved rows to disk	189

CHAPTER 6 Controlling Updates in DataWindow Objects..... 191

About controlling updates.....	191
What you can do	192
Specifying the table to update.....	193
Specifying the unique key columns.....	193
Specifying an identity column.....	194
Specifying updatable columns	194
Specifying the WHERE clause for update/delete	195
Specifying update when key is modified	197
Using stored procedures to update the database	198

CHAPTER 7 Working with Controls in DataWindow Objects..... 201

Adding controls to a DataWindow object	201
Adding columns to a DataWindow object.....	201
Adding text to a DataWindow object	202
Adding drawing controls to a DataWindow object	203
Adding a group box to a DataWindow object	203
Adding pictures to a DataWindow object.....	204
Adding computed fields to a DataWindow object	205

- Adding buttons to a DataWindow object 210
- Adding graphs to a DataWindow object 213
- Adding InkPicture controls to a DataWindow object..... 213
- Adding reports to a DataWindow object 214
- Reorganizing controls in a DataWindow object..... 214
- Displaying boundaries for controls in a DataWindow object . 215
- Using the grid and the ruler in a DataWindow object 215
- Deleting controls in a DataWindow object..... 216
- Moving controls in a DataWindow object 216
- Copying controls in a DataWindow object..... 216
- Resizing controls in a DataWindow object 217
- Aligning controls in a DataWindow object 218
- Equalizing the space between controls in a DataWindow
 object 218
- Equalizing the size of controls in a DataWindow object 219
- Sliding controls to remove blank space in a DataWindow
 object 219
- Positioning controls in a DataWindow object 221
- Rotating controls in a DataWindow object 222

CHAPTER 8 Displaying and Validating Data 225

- About displaying and validating data 225
- Presenting the data 226
- Validating data..... 227
- About display formats..... 227
- Working with display formats 228
- Working with display formats in the Database painter 228
- Working with display formats in the DataWindow painter 230
- Defining display formats..... 231
- Number display formats 233
- String display formats..... 235
- Date display formats..... 235
- Time display formats 237
- About edit styles..... 238
- Working with edit styles..... 240
- Working with edit styles in the Database painter..... 240
- Working with edit styles in the DataWindow painter..... 242
- Defining edit styles 242
- The Edit edit style..... 242
- The DropDownList edit style 243
- The CheckBox edit style..... 244
- The RadioButtons edit style 245
- The EditMask edit style 246

	The DropDownDataWindow edit style.....	249
	The InkEdit edit style	251
	Defining a code table	251
	How code tables are implemented	252
	How code tables are processed	253
	Validating user input.....	254
	About validation rules.....	255
	Understanding validation rules	255
	Working with validation rules.....	256
	Defining validation rules	257
	Defining a validation rule in the Database painter	257
	Defining a validation rule in the DataWindow painter.....	260
	How to maintain extended attributes.....	263
CHAPTER 9	Filtering, Sorting, and Grouping Rows	265
	Filtering rows.....	265
	Sorting rows	267
	Suppressing repeating values	269
	Grouping rows.....	270
	Using the Group presentation style	272
	Defining groups in an existing DataWindow object	276
CHAPTER 10	Highlighting Information in DataWindow Objects	285
	Highlighting information.....	285
	Modifying properties when designing	285
	Modifying properties at runtime	286
	Modifying properties conditionally at runtime	289
	Example 1: creating a gray bar effect.....	290
	Example 2: rotating controls.....	291
	Example 3: highlighting rows of data.....	292
	Example 4: changing the size and location of controls	294
	Supplying property values.....	295
	Background.Color.....	297
	Border.....	298
	Brush.Color	299
	Brush.Hatch.....	300
	Color.....	301
	Font.Escapement (for rotating controls)	302
	Font.Height.....	303
	Font.Italic.....	304
	Font.Strikethrough.....	305
	Font.Underline	306
	Font.Weight	306

- Format 307
- Height 307
- Pen.Color 308
- Pen.Style 308
- Pen.Width 310
- Pointer 310
- Protect 311
- Timer_Interval 311
- Visible 311
- Width 312
- X 312
- X1, X2 313
- Y 313
- Y1, Y2 314
- Specifying colors 314

- CHAPTER 11**
- Using Nested Reports 317**
 - About nested reports 317
 - Creating a report using the Composite presentation style 319
 - Placing a nested report in another report 321
 - Placing a related nested report in another report 321
 - Placing an unrelated nested report in another report 324
 - Working with nested reports 325
 - Adjusting nested report width and height 325
 - Changing a nested report from one report to another 326
 - Modifying the definition of a nested report 326
 - Adding another nested report to a composite report 327
 - Supplying retrieval arguments to relate a nested report to its base report 327
 - Specifying criteria to relate a nested report to its base report 329
 - Using options for nested reports 330

- CHAPTER 12**
- Working with Crosstabs 333**
 - About crosstabs 333
 - Two types of crosstabs 336
 - Creating crosstabs 337
 - Associating data with a crosstab 338
 - Specifying the information 338
 - Viewing the crosstab 341
 - Specifying more than one row or column 343
 - Previewing crosstabs 344

	Enhancing crosstabs.....	344
	Specifying basic properties	345
	Modifying the data associated with the crosstab.....	346
	Changing the names used for the columns and rows	346
	Defining summary statistics.....	347
	Cross-tabulating ranges of values.....	350
	Creating static crosstabs	353
	Using property conditional expressions.....	354
CHAPTER 13	Working with TreeViews.....	357
	TreeView presentation style.....	357
	Creating a new TreeView DataWindow	359
	TreeView creation process.....	359
	Creating a TreeView DataWindow	359
	Adding and deleting TreeView levels.....	363
	Selecting a tree node and navigating the tree.....	364
	Sorting rows in a TreeView DataWindow.....	365
	TreeView DataWindow Design view	366
	Setting properties for the TreeView DataWindow	367
	Setting general TreeView properties.....	368
	Setting TreeView level properties.....	369
	Setting detail band properties.....	371
	TreeView DataWindow samples	371
	Data Explorer sample.....	372
	Data Linker sample	375
CHAPTER 14	Exporting and Importing XML Data.....	379
	About XML	379
	Valid and well-formed XML documents.....	380
	XML syntax.....	381
	XML parsing	382
	XML support in the DataWindow painter.....	383
	The Export/Import Template view for XML.....	384
	Creating templates	385
	Saving templates.....	387
	Header and Detail sections	387
	Editing XML templates	390
	XML declaration	391
	Document type declaration.....	392
	Root element	393
	Controls	394
	DataWindow expressions.....	394
	Attributes	395

- Composite and nested reports 395
- CDATA sections 397
- Comments 397
- Processing instructions 398
- Exporting to XML..... 398
 - Setting properties on the Data Export page 399
 - Selecting templates at runtime 408
- Importing XML..... 408
 - Importing with a template 409
 - Default data import..... 413
 - Tracing import 416

CHAPTER 15

Working with Graphs..... 419

- About graphs 419
 - Parts of a graph..... 420
 - Types of graphs..... 421
- Using graphs in DataWindow objects 425
 - Placing a graph in a DataWindow object..... 425
 - Using the graph's Properties view 426
 - Changing a graph's position and size..... 427
 - Associating data with a graph 428
 - Using overlays..... 437
- Using the Graph presentation style..... 439
- Defining a graph's properties 440
 - Using the General property page in the graph's Properties view 440
 - Sorting data for series and categories..... 442
 - Specifying text properties for titles, labels, axes, and legends 442
 - Specifying overlap and spacing..... 445
 - Specifying axis properties 446
 - Specifying a pointer..... 448

PART 3

APPENDIXES

APPENDIX A

- Identifiers 451**
 - Rules 451
 - Reserved words 452

APPENDIX B	The Extended Attribute System Tables	453
	About the extended attribute system tables	453
	The extended attribute system tables	454
	Edit style types for the PBCatEdt table	457
	CheckBox edit style (code 85).....	457
	RadioButton edit style (code 86)	458
	DropDownListBox edit style (code 87)	459
	DropDownDataWindow edit style (code 88).....	460
	Edit edit style (code 89).....	461
	Edit Mask edit style (code 90)	463
APPENDIX C	Starting DataWindow Designer from the Command Line	465
	Starting DataWindow Designer from the command line	465
Index		467

About This Book

Audience	This book is written for programmers who need to design DataWindow® objects. DataWindow objects are used to retrieve, present, and manipulate data from a relational database or other source. You can use the DataWindow objects that you build in DataWindow Designer to create applications for the Microsoft .NET Framework using Sybase DataWindow .NET™.
How to use this book	This book guides you through the process of using DataWindow Designer to create DataWindow objects.
Related documents	<p>For information on using DataWindow objects with the Microsoft .NET Framework, see the <i>DataWindow .NET Programmer's Guide</i>.</p> <p>For a complete list of books and online documentation, see the preface in the <i>DataWindow .NET Programmer's Guide</i>.</p>
Other sources of information	<p>Use the Sybase Getting Started CD, the SyBooks CD, and the Sybase Product Manuals Web site to learn more about your product:</p> <ul style="list-style-type: none">• The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.• The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format. <p>Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.</p> <p>Refer to the <i>SyBooks Installation Guide</i> on the Getting Started CD, or the <i>README.txt</i> file on the SyBooks CD for instructions on installing and starting SyBooks.</p>

-
- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

To access the Sybase Product Manuals Web site, go to Product Manuals at <http://www.sybase.com/support/manuals/>.

Sybase EBFs and software maintenance

❖ Finding the latest information on EBFs and software maintenance

- 1 Point your Web browser to the Sybase Support Page at <http://www.sybase.com/support>.
- 2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.
- 3 Select a product.
- 4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the “Technical Support Contact” role to your MySybase profile.

- 5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

PART 1

The DataWindow Designer Environment

This part introduces you to DataWindow Designer and describes how to work in and manage its environment. It also describes how to work with databases, tables, views, and extended attributes.

Working with DataWindow Designer

About this chapter

This chapter describes the basics of working with DataWindow Designer and its painters.

Contents

Topic	Page
About DataWindow Designer	4
The DataWindow Designer environment	4
Working with projects	6
Working with DataWindow objects	7
Working in painters	10
Working with tools	17
Setting properties	20
Using toolbars	21
Using online Help	29
Customizing keyboard shortcuts	30
Changing fonts	31
How your DataWindow Designer environment is managed	32

About DataWindow Designer

DataWindow Designer is a tool that creates DataWindow objects for use in applications. A DataWindow object is an object that you use to retrieve and manipulate data from a relational database or other data source. The following DataWindow object retrieves and updates employee data.

The screenshot shows a preview window titled "Preview - d_employee_details". The main content is a form for "Employee Data for Fran Whitney" with "Employee ID: 102" and a date of "27/1/2006". The form is organized into two columns. The left column contains: Manager ID: 501, Department ID: 100, Salary: \$45,700.00, Street: 49 East Washington Street, City: Needham, State: MA, Zip Code: 02192-1505, and Phone: (617) 555-3985. The right column contains: Social Security Number: 017-34-9033, Start Date: 02/26/2004, Termination Date: 00/00/0000, Birth Date: 06/05/1962, Sex: Female (selected with a radio button), Health Insurance: checked (checkbox), and Life Insurance: checked (checkbox).

DataWindow Designer provides built-in connectivity to a broad range of desktop and server-based databases. It includes the EAS Demo DB (a Sybase Adaptive Server® Anywhere database) to create reports and other DataWindow Designer objects.

When you work in DataWindow Designer, you work in a graphical environment—and working with data in this environment means you do not need to understand SQL, the standard programming language for talking to databases. DataWindow Designer creates all SQL statements behind the scenes as you build your DataWindow objects and other objects graphically.

Painters

In DataWindow Designer, you do most of your work in the DataWindow painter, where you “paint” your DataWindow objects. DataWindow Designer also has painters where you work with databases, libraries, and SQL statements.

The DataWindow Designer environment

When you start DataWindow Designer the first time

In DataWindow Designer, you always work within the context of a DataWindow project. Information about the project is stored in a text file with the extension *.dwp*. The DataWindow objects you create are stored in binary files called libraries. Library files have the extension *.pbl*.

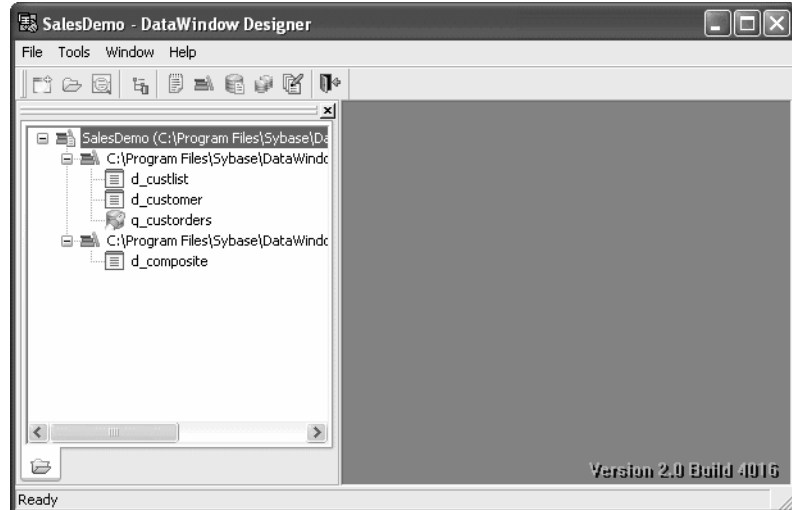
To create a new project and a library to hold your DataWindow objects, click the New button on the PowerBar and select Project from the Project tab page.

If you have used DataWindow Designer before

You can open an existing project by selecting Open Project or Recent Projects from the File menu, or Open Project from the pop-up menu in the System Tree.

When DataWindow Designer starts

When DataWindow Designer starts, it opens in a window that contains the System Tree on the left, a menu bar at the top, and the PowerBar below the menu bar. The following figure shows an open project in the System Tree:

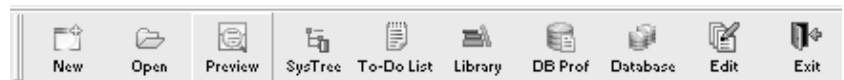


You can create new objects, open existing objects, change libraries, access the database, and perform other tasks by using pop-up menus in the System Tree, menus on the menu bar, or buttons in the PowerBar.

About the PowerBar

What it is

From the PowerBar you can create new projects and objects and open existing objects. You can display text on each button to help you identify them, as shown in the following illustration:



Buttons on the PowerBar

From left to right on the PowerBar, here are the buttons and what you can do after you click a button:

This PowerBar button	Lets you do this
New	Create new projects and DataWindow objects and open tools
Open	Open existing objects
Preview	Preview DataWindow objects
System Tree	Show or hide the System Tree
To-Do List	Keep track of object creation tasks and use links to quickly get you to the place where you complete the tasks
Library	Manage your libraries using the Library painter
DB Profile	Define and use named sets of parameters to connect to a particular database
Database	Maintain databases and database tables, control user access to databases, and manipulate data in databases using the Database painter
File Editor	Edit text files
Exit	Close DataWindow Designer

Customizing the PowerBar

You can customize the PowerBar. For example, you can move it to different locations and add buttons for operations you perform frequently. For more information, see “Using toolbars” on page 21.

About PowerTips

In the PowerBar, when you leave the mouse pointer over a button for a second or two, DataWindow Designer displays a brief description of the button, called a **PowerTip**. PowerTips display in DataWindow Designer wherever there are toolbar buttons.

Working with projects

You can create a new project, add existing libraries to a project’s library list, create a new library, and open a different project to work on objects in a library in that project. The libraries in your current library list display in the System Tree. You can open the DataWindow objects in any library on your current list.

❖ To create a new project:

- 1 Click the New button in the PowerBar, or select File>New from the menu bar.
- 2 On the Project page in the New dialog box, select the project icon and click OK.
- 3 Browse to the location where you want to store the project and type a name for the project.
- 4 Do the same for the library, or simply click Finish to create the library in the same folder as the project and with the same name.

❖ To add existing libraries to a project:

- 1 Right-click the project icon in the System Tree and select Properties.
- 2 Click the Browse button to navigate to the folder where the libraries reside.
- 3 Select one or more libraries, click Open, then click OK in the Library List dialog box.

❖ To create a new library:

- 1 Right-click the project icon in the System Tree and select New Library.
- 2 Browse to the location where you want to create the library, type a name for the library, and click OK.

❖ To open a different project:

- Right-click the project icon in the System Tree and select Open Project, or select File>Open Project from the File menu.

Working with DataWindow objects

In DataWindow Designer, you can:

- Create new DataWindow objects
- Open existing DataWindow objects
- Run or preview DataWindow objects

After you create or open a DataWindow object, it displays in the DataWindow painter and you work on it there.

Creating new DataWindow objects

To create new DataWindow objects, use the New button in the PowerBar.

❖ **To create a new object:**

- 1 Click the New button in the PowerBar, or select File>New from the menu bar.
- 2 In the New dialog box, select the DataWindow tab page.
- 3 Select an icon and click OK.

The DataWindow wizard helps you create a new DataWindow object. For more information about using the wizard, see Chapter 4, “Defining DataWindow Objects.”

Opening existing DataWindow objects

There are several ways to open existing objects. The quickest way is to double-click its icon in the System Tree, or select it from the list of recent objects on the File menu. The recent objects list includes the eight most recently opened objects, but you can include up to 36 objects on the list.

❖ **To modify the number of recent objects:**

- 1 Select Tools>System Options from the menu bar.
- 2 In the System Options dialog box (General tab page), modify the number for the recent objects list.

You can also use the Open button in the PowerBar to open a DataWindow object. The DataWindow object must be in a library on your current library list.

❖ **To open existing objects:**

- 1 Click the Open button in the PowerBar, or select File>Open from the menu bar.

- In the Open dialog box, select one or more libraries in the Library pane and then select the object you want to open.



- Click OK.

The object opens in the DataWindow painter.

Running or previewing DataWindow objects

To preview a DataWindow object, use the Preview button in the PowerBar.

❖ To run or preview an object:

- Click the Preview button in the PowerBar, or select File>Run/Preview from the menu bar.
- In the Run/Preview dialog box, select one or more libraries in the Library pane and then select the DataWindow object you want to preview.
- Click OK.

The DataWindow object displays in DataWindow Preview mode. This mode has a Painterbar from which you can sort, filter, import, and retrieve rows, insert and delete rows, get information on row activity, and zoom in and out. To return to the DataWindow painter, click the X in the Painterbar.

Working in painters

The DataWindow Designer painters are listed in Table 1-1.

Table 1-1: DataWindow Designer painters

Painter	What you do
Database painter	Maintain databases, control user access to databases, manipulate data in databases, and create tables
DataWindow painter	Build intelligent objects called DataWindow objects that present information from the database
Library painter	Manage libraries
Query painter	Graphically define and save SQL SELECT statements for reuse with DataWindow objects
Select painter	Graphically define SQL SELECT statements for DataWindow objects

Using views in painters

Most of the DataWindow Designer painters have views. Each view provides a specific way of viewing or modifying the object you are working on or a specific kind of information related to that object. Having all the views available in a painter window means you can work on more than one task at a time.

Views are displayed in panes in the painter window. Some views are stacked in a single pane. At the bottom of the pane there is a tab for each view in the stack. Clicking the tab for a view pops that view to the top of the stack.

Each painter has a default layout, but you can display the views you choose in as many panes as you want to and save the layouts you like to work with. For some painters, all available views are included in the default layout; for others, only a few views are included.

Each pane has:

- A title bar you can display temporarily or permanently
- A handle in the top-left corner you can use to drag the pane to a new location
- Splitter bars between the pane and each adjacent pane

Displaying the title bar

For most views a title bar does not permanently display at the top of a pane (because it is often unnecessary). But you can display a title bar for any pane either temporarily or permanently.

❖ To display a title bar:

- 1 Place the pointer on the splitter bar at the top of the pane.

The title bar displays.

- 2 To display the title bar permanently, click the pushpin at the left of the title bar or select Pinned from its pop-up menu.

Click the pushpin again or select Pinned again on the pop-up menu to hide the title bar.

After you display a title bar either temporarily or permanently, you can use the title bar's pop-up menu.

❖ To maximize a pane to fill the workspace:

- Select Maximize from the title bar's pop-up menu or click the Maximize button on the title bar.

❖ To restore a pane to its original size:

- Select Restore from the title bar's pop-up menu or click the Restore button on the title bar.

Moving and resizing panes and views

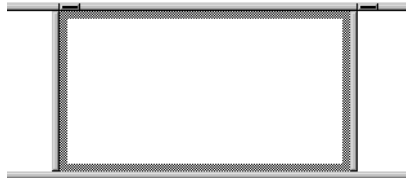
You can move a pane or a view to any location in the painter window. You might find it takes a while to get used to moving panes and views around, but if you do not like a layout, you can always revert to the default layout and start again. To restore the default layout, select View>Layouts>Default.

To move a pane, you select and drag the title bar of the view that is at the top of the stack. If the pane contains stacked views, *all* views in the stack move together. To move one of the views out of the stack, you drag the tab for the view you want to move.

❖ To move a pane:

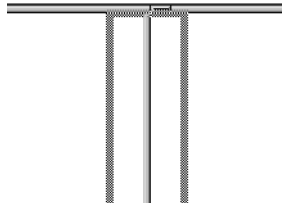
- 1 Place the pointer anywhere on the title bar of the view at the top of the stack, hold down the left mouse button, and start moving the pane.

A gray outline appears in the pane:



- 2 Drag the outline to the new location.

The outline changes size as you drag it. When the pointer is over the middle of a pane, the outline fills the pane. As you drag the pointer toward any border, the outline becomes a narrow rectangle adjacent to that border. When the pointer is over a splitter bar between two panes, rows, or columns, the outline straddles the splitter bar:



When you move the pointer to a corner

When you move the pointer to a corner, you will find that you have many places where you can drop the outline. To see your options, move the pointer around in all directions in the corner and see where the outline displays as you move it.

- 3 Release the mouse button to drop the outline in the new location:

To move a pane here	Drop the outline here
Between two panes	On the splitter bar between the panes
Between a border and a pane	At the side of the pane nearest the border
Into a new row	On the splitter bar between two rows or at the top or bottom of the painter window
Into a new column	On the splitter bar between two columns or at the left or right edge of the painter window
Onto a stack of panes	On the middle of the pane (if the pane was not already tabbed, tabs are created)

❖ To move a view in a stacked pane:

- Place the pointer anywhere on the view's tab, hold down the left mouse button, and start moving the view.

You can now move the view as in the previous procedure. If you want to rearrange the views in a pane, you can drag the view to the left or right within the same pane.

❖ To resize a pane:

- Drag the splitter bars between panes.

Floating and docking views

Panes are docked by default within a painter window, but some tasks may be easier if you float a pane. A floating pane can be moved outside the painter's window or even outside the DataWindow Designer window.

When you open another painter

If you have a floating pane in a painter and then open another painter, the floating pane temporarily disappears. It reappears when the original painter is selected.

❖ To float a view in its own pane:

- Select Float from the title bar's pop-up menu.

❖ To float a view in a stacked pane:

- Select Float from the tab's pop-up menu.

❖ To dock a floating view:

- Select Dock from the title bar's pop-up menu.

Adding and removing views

You may want to add additional views to the painter window. If there are some views you rarely use, you can move them into a stacked pane or remove them. When removing a view in a stacked pane, make sure you remove the view and not the pane.

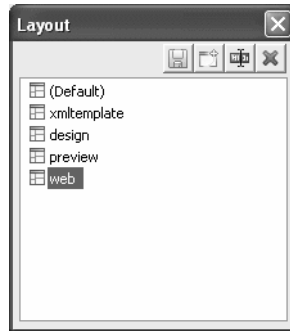
- ❖ **To add a new view to the painter window:**
 - 1 Select View from the menu bar and then select the view you want to add.
The view displays in a new pane in a new row.
 - 2 Move the pane where you want it.
For how to move panes, see “Moving and resizing panes and views” on page 11.
- ❖ **To remove a view in its own pane from the painter window:**
 - 1 If the view’s title bar is not displayed, display it by placing the pointer on the splitter bar at the top of the pane.
 - 2 Click the Close button on the title bar.
- ❖ **To remove a view in a stacked pane from the painter window:**
 - Select the tab for the view and select Close from its pop-up menu.
- ❖ **To remove a stacked pane from the painter window:**
 - 1 If the title bar of the top view in the stack is not displayed, display it by placing the pointer on the splitter bar at the top of the pane.
 - 2 Click the Close button on the title bar.

Saving a layout

When you have rearranged panes in the painter window, DataWindow Designer saves the layout in the registry. The next time you open the painter window, your last layout displays. You can also save customized layouts so that you can switch from one to another for different kinds of activities.

- ❖ **To save customized layouts for a painter window:**
 - 1 Select View>Layouts>Manage from the menu bar.

- Click the New Layout button (second from the left at the top of the dialog box).



- Type an appropriate name in the text box and click OK.

Restoring the default layout

You can restore the default layout at any time by selecting Views>Layout>Default.

Using pop-up menus

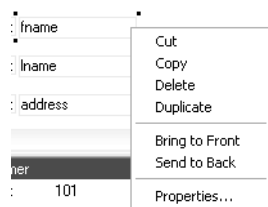
DataWindow Designer provides a context-sensitive pop-up menu that lists:

- Actions appropriate to the currently selected object or the current position of the pointer
- Where appropriate, a Properties menu item for accessing the Properties view or the Properties dialog box associated with the current object or the current position of the pointer

The pop-up menu is available almost everywhere in DataWindow Designer when you position the pointer over an object and click the right mouse button.

Example

For example, the following screen shows the pop-up menu for a column in a DataWindow:



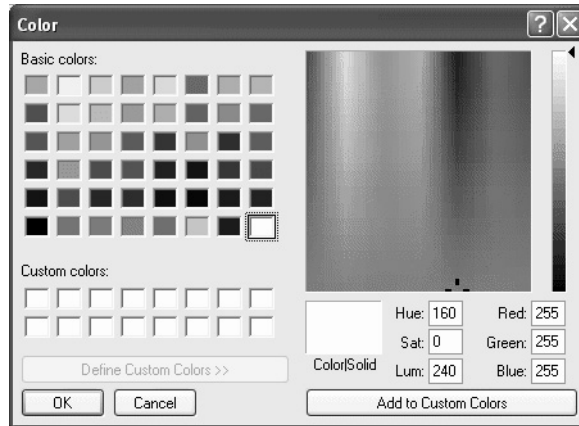
Defining colors

You can define custom colors to use in most painters and in objects you create.

❖ **To define custom colors:**

- 1 In a painter that uses custom colors, select Design>Custom Colors from the menu bar.

The Color dialog box displays:



- 2 Define your custom colors:

Area of the Color dialog box	What you do
Basic colors	Click the basic color closest to the color you want to define to move the pointer in the color matrix and slider on the right
Custom colors palette	Modify an existing color—click a custom color, then modify the color matrix and slider. Define a new color—click an empty box, define the color, and click Add to Custom Colors
Color matrix	Click in the color matrix to pick a color
Color slider	Move the slider on the right to adjust the color's attributes
Add to Custom Colors button	After you have designed the color, click this button to add the custom color to the Custom colors palette on the left

Working with tools

In addition to painters, DataWindow Designer provides you with some other tools to help you with your work. You can open most tools from the Tools menu, the PowerBar, or the Tools page in the New dialog box.

Tool summary

Table 1-2 lists the tools.

Table 1-2: DataWindow Designer tools

Tool	What you use the tool for
File editor	Edit text files. For more information, see "Using the file editor" next.
To-Do List	Keep track of object creation tasks and create links to quickly get you to the place where you need to complete the tasks. For information, see "Using the To-Do List" on page 18.
Database profile	Define and use named sets of parameters to connect to a particular database. For information, see <i>Connecting to Your Database</i> .

Using the file editor

DataWindow Designer provides a text editor that is always available. Using the editor, you can view and modify text files (such as initialization files and tab-separated files with data) without leaving DataWindow Designer.

❖ **To open the file editor:**

- 1 Press Shift+F6 anywhere in DataWindow Designer.
- 2 Select File>Open File or click the Open icon (an open folder) on the Painter bar to open the file you want to edit.

Setting file editing properties

The file editor has font properties and an indentation property that you can change to make files easier to read. If you do not change any properties, files have black text on a white background and a tab stop setting of 3 for indentation.

❖ **To specify File Editor properties:**

- 1 Select Design>Options to display the property page.
- 2 Choose the tab appropriate to the property you want to specify.

Editing activities

The file editor provides a full set of basic editing facilities including:

- Opening, saving, and printing files
- Cutting, copying, pasting, and clearing selected text
- Finding and replacing text
- Undoing changes
- Commenting and uncommenting lines
- Importing and exporting text files
- Dragging and dropping text

The file editor has a PainterBar that provides a shortcut for performing frequently used activities. There is also a corresponding menu item (and often a shortcut key) for each activity.

To see the shortcut keys, select Tools>Keyboard Shortcuts from the menu bar and use the Keyboard Shortcuts dialog box.

To move text, simply select it, drag it to its new location, and drop it. To copy text, press the Ctrl key while you drag and drop the text.

Using the To-Do List

Opening the To-Do List

The To-Do List displays a list of tasks you want to do in the current project.

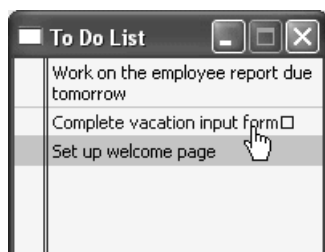
❖ To open the To-Do List:

- Click the To-Do List button in the PowerBar, or select Tools>To-Do List from the menu bar.

To-Do List entries

You can create an entry in the To-Do List at any time to remind you about any task you need to complete. You can create entries that are hot-linked to quickly get you from the To-Do List to the DataWindow painter and the specific object you need.

When you move the pointer over entries on the To-Do list, the pointer changes to a hand if it is over a linked entry.



You can export or import a To-Do List by selecting Import or Export from the pop-up menu. Doing this is useful if you want to move from one computer to another or you need to work with To-Do Lists as part of some other system such as a project management system.

Working with entries on the To-Do List

Table 1-3 tells you how to work with entries on the To-Do List.

Table 1-3: The To-Do List

To do this	Do this
See linked entries	Move the pointer over the entries. A hand displays when the entry you are over is linked.
Use a linked entry to get to a painter or wizard	Double-click the linked entry or select it and then select Go To Link from the pop-up menu.
Add an entry with no link	Select Add from the pop-up menu.
Add a linked entry to the DataWindow painter	With the painter open, select Add Linked from the pop-up menu.
Change an entry's position on the list	Drag the entry to the position you want.
Edit or delete an entry	Select Edit or Delete from the pop-up menu.
Delete checked entries or all entries	Select Delete Checked or Delete All from the pop-up menu.
Check or uncheck an entry	Select an entry and then select Check/Uncheck from the pop-up menu.
Export a To-Do List	Select Export from the pop-up menu, name the To-Do List text file, and click Save.
Import a To-Do List	Select Import from the pop-up menu, navigate to an exported To-Do List text file, and click Open.

Setting properties

In DataWindow Designer, you set properties on pages that display in a tabbed dialog box or in a view in a painter. In the DataWindow painter, you set object properties in the Properties view. In the Database painter, you set properties in the Object Details view.

For example, for a column in a DataWindow object, you can set several different kinds of properties (general, pointer, HTML, position, edit style, font, and display format) by clicking appropriate tabs in the Properties view.

How property pages work in the Properties view

The Properties view is dynamically updated when you select another object or control. If you select more than one object or control, `group selected` displays in the title bar, the properties common to them display, and you can set the properties for more than one control at a time.



In the Properties view, you can use a pop-up menu to specify where the labels for the properties display and to get Help on the properties.

The selections you make or the information you type into a box in the Properties view is saved when you tab to another field or open a different property page.

How other property pages work

Properties dialog boxes that do not display in the Properties view have OK, Cancel, Apply, and Help buttons. The Apply button is enabled when you make a change on one tab:

Use this button	To do this
OK	Apply the properties you have set on all tabs and close the property page
Cancel	Close the window and apply no new changes
Apply	Apply the properties you have set on all tabs immediately without closing the property page
Help	Get Help on setting properties for the tab that displays

Displaying properties pages

You can display properties pages in a few ways:

- Select View>Properties from the menu bar in the DataWindow painter or View>Object Details in the Database painter
- If the Properties view is open, select an object or control in the Layout or Control List views to display the properties of the object or control in the Properties view

- Select Properties from the pop-up menu of an object, control, library name, or table or column name
- Select Object>Properties or Entry>Properties from the menu bar (depending on the painter you are working in)
- Click the Properties button in the PainterBar

Using toolbars

Toolbars provide buttons for the most common tasks in DataWindow Designer. You can move and dock or float toolbars, customize them, and create your own.

Toolbar basics

DataWindow Designer uses three toolbars: the PowerBar, PainterBar, and StyleBar. You can hide a toolbar by right-clicking in the toolbar area and clearing the check mark text to its name. If a toolbar is not hidden, it displays as shown in Table 1-4.

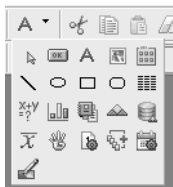
Table 1-4: DataWindow Designer toolbars

Toolbar	Purpose	Availability
PowerBar	Opening painters and tools	Always.
PainterBar	Performing tasks in the current painter	When a painter is open. Some painters have more than one PainterBar.
StyleBar	Changing the properties of text, such as font and alignment	In appropriate painters.

Drop-down toolbars

To reduce the size of toolbars, some toolbar buttons have a down arrow on the right that you can click to display a drop-down toolbar containing related buttons.

For example, the down arrow next to the Text button in the DataWindow painter displays the Controls drop-down toolbar, which has a button for each control you can place on a DataWindow object:



Default button replaced

The button you select from a drop-down toolbar replaces the default button on the main toolbar. For example, if you select the Picture button from the Controls drop-down toolbar, it replaces the Command button in the PainterBar.

Controlling the display of toolbars

You can control:

- Whether to display individual toolbars and where
- Whether to display text on the buttons
- Whether to display PowerTips

Choosing to display text and PowerTips affects all toolbars.

❖ To control a toolbar using the pop-up menu:

- 1 Position the pointer on a toolbar and display the pop-up menu.
- 2 Click the items you want.

A check mark means the item is currently selected.

❖ To control a toolbar using the Toolbars dialog box:

- 1 Select Tools>Toolbars from the menu bar.

The Toolbars dialog box displays.

- 2 Click the toolbar you want to work with (the current toolbar is highlighted) and the options you want.

DataWindow Designer saves your toolbar preferences in the registry and the DataWindow Designer initialization file.

Moving toolbars using the mouse

You can use the mouse to move a toolbar.

❖ **To move a toolbar with the mouse:**

- 1 Position the pointer on the grab bar at the left of the toolbar or on any vertical line separating groups of buttons.
- 2 Press and hold the left mouse button.
- 3 Drag the toolbar and drop it where you want it.

As you move the mouse, an outlined box shows how the toolbar will display when you drop it. You can line it up along any frame edge or float it in the middle of the frame.

Docking toolbars

When you first start DataWindow Designer, all the toolbars display one above another at the top left of the workspace. When you move a toolbar, you can dock (position) it:

- At the top or bottom of the workspace, at any point from the left edge to the right edge
- At the left or right of the workspace, at any point from the top edge to the bottom edge
- To the left or right of, or above or below, another toolbar

Customizing toolbars

You can customize toolbars with DataWindow Designer buttons and with buttons that invoke other applications, such as a clock or text processor.

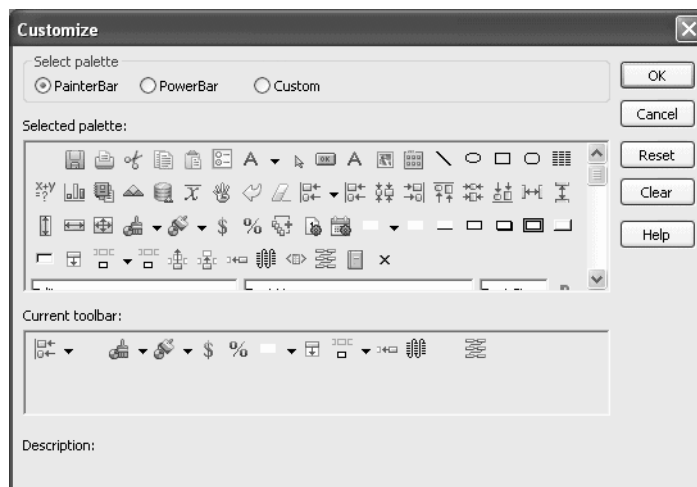
Adding, moving, and deleting buttons

You can add, move, and delete buttons in any toolbar.

❖ **To add a button to a toolbar:**

- 1 Position the pointer on the toolbar and display the pop-up menu.
- 2 Select Customize.

The Customize dialog box displays.



- 3 Click the palette of buttons you want to use in the Select palette group.
- 4 Choose a button from the Selected palette box and drag it to the position you want in the Current toolbar box.

If you choose a button from the Custom palette, another dialog box displays so you can define the button.

For more information, see “Adding a custom button” on page 25.

Seeing what's available in the PowerBar

DataWindow Designer provides several buttons that do not display by default in the PowerBar, but which you can add. To see what is available, scroll the list of buttons and select one. DataWindow Designer lists the description for the selected button.

❖ **To move a button on a toolbar:**

- 1 Position the pointer on the toolbar, display the pop-up menu, and select Customize.
- 2 In the Current toolbar box, select the button and drag it to its new position.

❖ **To delete a button from a toolbar:**

- 1 Position the pointer on the toolbar, display the pop-up menu, and select Customize.
- 2 In the Current toolbar box, select the button and drag it outside the Current toolbar box.

Resetting a toolbar

You can restore the original setup of buttons on a toolbar at any time.

❖ **To reset a toolbar:**

- 1 Position the pointer on the toolbar, display the pop-up menu, and select Customize.
- 2 Click the Reset button, then Yes to confirm, then OK.

Clearing or deleting a toolbar

Whenever you want, you can remove all buttons from a toolbar. If you do not add new buttons to the empty toolbar, the toolbar is deleted. You can delete both built-in toolbars and toolbars you have created.

To recreate a toolbar

If you delete one of DataWindow Designer's built-in toolbars, you can recreate it easily. For example, to recreate the PowerBar, display the pop-up menu, select New, and then select PowerBar1 in the New Toolbar dialog box. For information about creating new toolbars and about the meaning of PowerBar1, see "Creating new toolbars" on page 28.

❖ **To clear or delete a toolbar:**

- 1 Position the pointer on the toolbar, display the pop-up menu, and select Customize.
- 2 Click the Clear button, then Yes to confirm.
The Current toolbar box in the Customize dialog box is emptied.
- 3 Select new buttons for the current toolbar and click OK, or click OK to delete the toolbar.

Adding a custom button

You can add a custom button to a toolbar. A custom button can:

- Invoke a DataWindow Designer menu item
- Run an executable (application) outside DataWindow Designer
- Run a query or preview a DataWindow object
- Assign a display format or create a computed field in a DataWindow object

❖ **To add a custom button:**

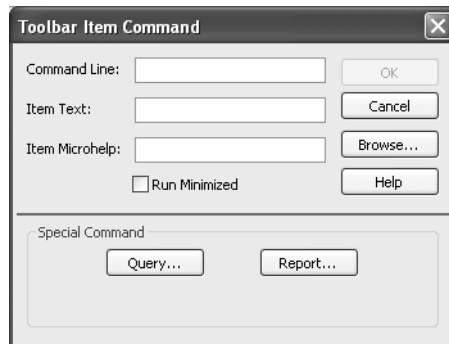
1 Position the pointer on the toolbar, display the pop-up menu, and select Customize.

2 Select Custom in the Select Palette group.

The custom buttons display in the Selected Palette box.

3 Select a custom button and drag it to where you want it in the Current toolbar box.

The Toolbar Item Command dialog box displays. Different buttons display in the dialog box depending on which toolbar you are customizing:



4 Fill in the dialog box as shown in Table 1-5.

Table 1-5: Toolbar Item Command dialog box

Button purpose	Action in Toolbar Item Command dialog box
Invoke a DataWindow Designer menu item	Type <code>@MenuBarItem.MenuItem</code> in the Command Line box. For example, to make the button mimic the Open item on the File menu, type <code>@File.Open</code> You can also use a number to refer to a menu item. The first item in a drop-down/cascading menu is 1, the second item is 2, and so on. Separator lines in the menu count as items. Example: <code>@Edit.Align.4</code>
Run an executable outside DataWindow Designer	Type the name of the executable in the Command Line box. Specify the full path name if the executable is not in the current search path. To search for the file name, click the Browse button.

Button purpose	Action in Toolbar Item Command dialog box
Run a query	Click the Query button and select the query from the displayed list.
Preview a DataWindow object	Click the Report button and select a DataWindow object from the displayed list. You can then specify command-line arguments in the Command Line box, as described below.
Assign a display format to a column in a DataWindow object	(DataWindow painter only) Click the Format button to display the Display Formats dialog box. Select a data type, then choose an existing display format from the list or define your own in the Format box. For more about specifying display formats, see Chapter 8, “Displaying and Validating Data.”
Create a computed field in a DataWindow object	(DataWindow painter only) Click the Function button to display the Function for Toolbar dialog box. Select the function from the list.

- In the Item Text box, specify the text associated with the button in two parts separated by a comma—the text that displays on the button and text for the button's PowerTip:

ButtonText, PowerTip

For example:

Save, Save File

If you specify only one piece of text, it is used for both the button text and the PowerTip.

- In the Item MicroHelp box, specify the text to appear as MicroHelp when the pointer is on the button.

Supplying arguments with DataWindow objects

If you define a custom button to run a DataWindow object, you can specify arguments in the command line in the Toolbar Item Command dialog box.

Table 1-6: Arguments for running reports

Argument	Meaning
<i>/l LibraryName</i>	Specifies the library containing the DataWindow object
<i>/o DataWindowName</i>	Specifies the DataWindow object
<i>/r</i>	Previews the DataWindow object
<i>/ro</i>	Previews the DataWindow object but does not provide design mode for modifying the DataWindow object
<i>/a "Arguments"</i>	Specifies arguments to pass to the DataWindow object

The default command line is:

`DataWindow /o DataWindowName /ro`

Modifying a custom button

❖ **To modify a custom button:**

- 1 Position the pointer on the toolbar, display the pop-up menu, and select Customize.
- 2 Double-click the button in the Current toolbar box.
- 3 Make your changes, as described in “Adding a custom button” on page 25.

Creating new toolbars

DataWindow Designer has built-in toolbars. When you start DataWindow Designer, you see what is called the PowerBar. In each painter, you also see one or more PainterBars. But PowerBar and PainterBar are actually types of toolbars you can create to make working in DataWindow Designer easier.

PowerBars and
PainterBars

A PowerBar is a toolbar that always displays in DataWindow Designer, unless you hide it. A PainterBar is a toolbar that always displays in the specific painter for which it was defined, unless you hide it. You can have up to four PowerBars, named PowerBar1, PowerBar2, and so on. You can have up to eight PainterBars in each painter, named PainterBar1, PainterBar2, and so on.

Where you create
them

You can create a new PowerBar anywhere in DataWindow Designer, but to create a new PainterBar, you must be in the workspace of the painter for which you want to define the PainterBar.

❖ **To create a new toolbar:**

- 1 Position the pointer on any toolbar, display the pop-up menu, and select New.

About the StyleBar

In painters that do not have a StyleBar, StyleBar is on the list in the New Toolbar dialog box. You can define a toolbar with the name StyleBar, but you can add only painter-specific buttons, not style buttons, to it.

- 2 Select a PowerBar name or a PainterBar name and click OK.
The Customize dialog box displays with the Current toolbar box empty.
- 3 One at a time, drag the toolbar buttons you want from the Selected palette box to the Current toolbar box and then click OK.

Using online Help

DataWindow Designer has online Help that provides both reference and task-oriented information. Context-sensitive and reference information is provided in Windows Help format. Task-oriented information is provided in compiled HTML Help (CHM) format.

How to access Help

You can get Help in any of the ways listed in Table 1-7.

Table 1-7: Accessing Help

Approach	What it does
Use the Help menu on the menu bar or press F1	Displays the Help contents.
Add a Help button to the PowerBar and click it	Displays the Help contents.
In a wizard, click the Help button [?] in the upper right corner of the window	The pointer displays with a question mark so that you can get context-sensitive Help. Point and click in a field you need Help on.
In a wizard, press F1	Context-sensitive Help for the current field displays.
In the Properties view in the DataWindow painter, select Help from the pop-up menu on any tab page	Displays a Help topic from which you can get Help on the properties for the object or control whose properties are displaying in the Properties view.
Click the Help button in a dialog box	Displays information about that dialog box.
Click the Online Books button on the Help window	Open the compiled HTML books

Learning to use online Help

To get information on using Help, press F1 anywhere within online Help.

Using the pop-up menu

DataWindow Designer online Help provides a pop-up menu with shortcuts to features available on the Help menu bar. To display the pop-up menu in online Help, click the right mouse button.

Sybooks CD and Web site

DataWindow .NET books are also provided on the Sybooks CD and the Sybase Product Manuals Web site. For more information, see “Other sources of information” on page xv.

Customizing keyboard shortcuts

You can associate your own keyboard shortcuts with DataWindow Designer menu items.

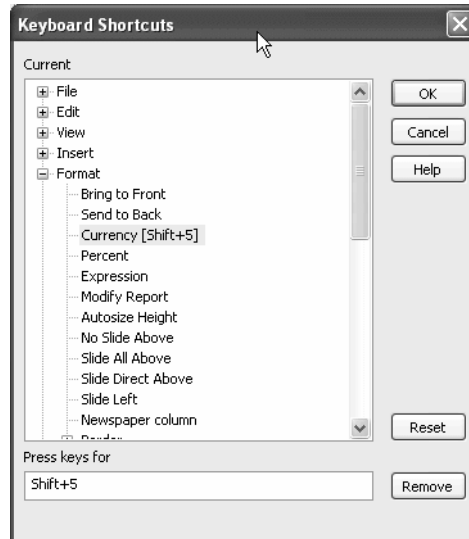
Tip

Creating keyboard shortcuts means you can use the keyboard instead of the mouse in many situations, such as changing libraries, objects, or connections, by creating shortcuts for the File>Recent menu items.

❖ To associate a keyboard shortcut with a menu item:

- 1 Select Tools>Keyboard Shortcuts from the menu bar.

The keyboard shortcuts for the current menu bar display.



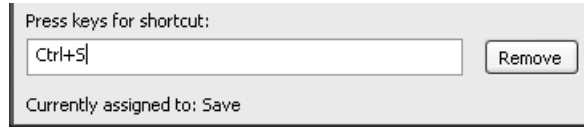
Keyboard shortcuts in a painter or tool

In a painter, the Keyboard Shortcuts dialog box includes both menu items as well as additional actions that apply to the current painter or tool. These nonmenu actions are listed under Additional Global Shortcuts and Additional Painter Shortcuts.

- 2 Select a menu item with no shortcut or a menu item with a default shortcut that you want to change and then put the cursor in the Press Keys For Shortcut textbox.

- 3 Press the keys you want for the shortcut; the new shortcut displays in the textbox.

If you type a shortcut that is already being used, a message notifies you so that you can type a different shortcut or change the existing shortcut.



❖ **To remove a keyboard shortcut associated with a menu item:**

- 1 Select Tools>Keyboard Shortcuts from the menu bar.
- 2 Select the menu item with the shortcut you want to remove.
- 3 Click Remove.

You can reset keyboard shortcuts to the default shortcuts globally or only for the current painter.

❖ **To reset keyboard shortcuts to the default:**

- Click the Reset button and respond to the prompt.

Changing fonts

Table 1-8 summarizes the various ways you can change the fonts used in DataWindow Designer.

Table 1-8: Changing fonts

For this object or painter	Do this
A table's data, headings, and labels	In the Database painter, display the table's property page, and change the font properties on the Data, Heading, and Label Font tabs.
Objects in the DataWindow painter	Select objects and then modify settings in the StyleBar, or, in the Properties view for one or more objects, change the font properties on the Font tab.
Library painter and MicroHelp	Select Tools>System Options from the menu bar and change the font properties on the System Font and Printer Font tab pages.

For this object or painter	Do this
Interactive SQL view in the Database painter and the file editor (changes made for one of these apply to both)	Select Design>Options from the menu bar to display the editor's property page and change the font properties on the Editor Font and Printer Font tab pages.

Use the Printer font tab to set fonts specifically for printing. If you need to print multilanguage characters, make sure you use a font that is installed on your printer. Changes you make in the Tools>System Options dialog box and from the Design>Options menu selection are used the next time you open DataWindow Designer.

How your DataWindow Designer environment is managed

Your DataWindow Designer configuration information is stored in both the DataWindow Designer initialization file (*DW.INI*) file and the registry. When you start DataWindow Designer, it looks in the registry and the *DW.INI* file to set up your environment.

About the registry

Some DataWindow Designer features require the use of the *DW.INI* file, but many features use the registry to get and store configuration information. Normally, you should not need to access or modify items in the registry.

Information related to your preferences (such as the way you have arranged your views in the painters and the shortcut keys you have defined for DataWindow Designer menu items) is stored in *HKEY_CURRENT_USER/Software/Sybase/DataWindowDesigner/2.0*.

Installation-related information is stored in *HKEY_LOCAL_MACHINE/Software/Sybase/DataWindowDesigner/2.0*.

About the initialization file

The initialization file is a text file that contains variables that specify your DataWindow Designer preferences. These preferences include information such as the last database you connected to and the PBL you are using. When you perform certain actions in DataWindow Designer, DataWindow Designer writes your preferences to the initialization file automatically.

Specifying preferences

Normally, you do not need to edit the initialization file. You can specify all your preferences by taking an action, such as resizing a window or opening a new application, or by selecting Design>Options from one of the painters. But sometimes a variable does not appear by default in the options sheet for the painter. In this case, you can use a text editor to modify the variable in the appropriate section of the initialization file.

Do not use a text editor to edit the DataWindow Designer initialization file while DataWindow Designer is running. DataWindow Designer caches the contents of initialization files in memory and overwrites your edited DataWindow Designer initialization file when it exits, ignoring changes.

Format of INI files

The DataWindow Designer initialization file uses the Windows INI file format. It has three types of elements:

- Section names, which are enclosed in square brackets
- Keywords, which are the names of preference settings
- Values, which are numeric or text strings, assigned as the value of the associated keyword

A variable can be listed with no value specified, in which case the default is used.

Some sections are always present by default, but others are created only when you specify different preferences. If you specify preferences for another painter or tool, DataWindow Designer creates a new section for it at the end of the file.

Where the initialization file is kept

DW.INI is installed in the same directory as the DataWindow Designer executable file.

You can keep your initialization file in another location and tell DataWindow Designer where it can find it by specifying the location in the System Options dialog box. You might want to do this if you use more than one version of DataWindow Designer or if you are running DataWindow Designer over a network.

❖ **To specify a different location for DW.INI:**

- 1 Select Tools>System Options from the menu bar.
- 2 On the General tab page, enter the path of your initialization file in the Initialization Path textbox.

DataWindow Designer records the path in the Windows registry.

DataWindow Designer looks in the Windows Registry for a path to the file, and then looks for the file in the directory where DataWindow Designer is installed. If DataWindow Designer cannot find the initialization file using the path in the Registry, it clears the path value.

If DataWindow Designer does not find the initialization file when it starts up, it recreates it. However, if you want to retain any preferences you have set, such as database profiles, keep a backup copy of your initialization file. The recreated file has the default preferences.

About this chapter

DataWindow Designer stores all the objects you create in libraries. This chapter describes how to work with your libraries.

Contents

Topic	Page
About libraries	35
About the Library painter	36
Working with libraries	37
Deleting libraries	44
Optimizing libraries	44
Regenerating library entries	45
Exporting and importing entries	46
Creating a deployment library (PBD file)	48
Creating a library directory report	49

About libraries

DataWindow Designer uses libraries to hold objects. A library is a file with the file extension PBL (pronounced *pibble*):

filename.pbl

In DataWindow Designer, you use the Library painter to work with the objects and libraries you create.

What you can do in the Library painter

The Library painter workspace displays objects in all libraries on your computer, but you can set the Library painter to display only objects in the current library. In the Library painter, you can:

- Open objects in the current library to edit them in the appropriate painters
- Copy, move, and delete objects in any library
- Optimize libraries

What you cannot do in the Library painter

You cannot create a library or rename a library in the Library painter. For information about creating a library, see “Working with projects” on page 6.

You cannot create new DataWindow objects or queries in the Library painter.

You cannot open objects that are *not* in the current project.

About the Library painter

❖ To open the Library painter:

- Click the Library button in the PowerBar.

Views in the Library painter

The Library painter has two views that you use for displaying library files (PBLs) and the objects they contain. The two views, which are available from the View menu, are Tree and List.

By default, the Library painter displays one Tree view (on the left) and one List view (on the right). When the Library painter opens, both the Tree view and the List view display the drives that are on your computer or mapped to it.

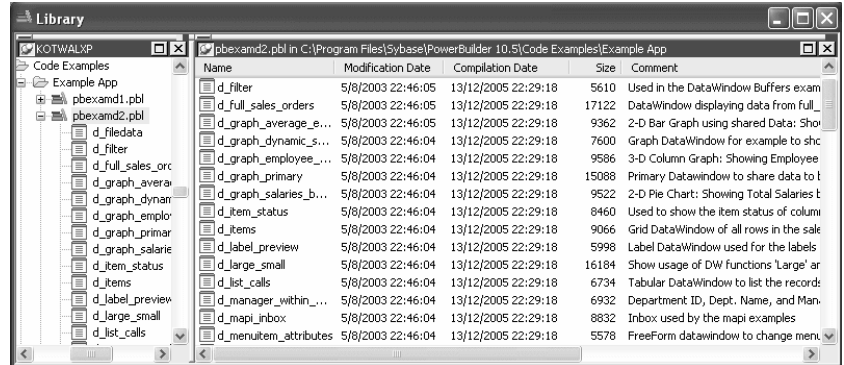
About the Tree view and the List view

Tree view The Tree view displays the drives and folders on the computer and the libraries and objects they contain.

List view The List view also displays the drives and folders on the computer and the libraries and objects they contain, but the view has columns with headers. For libraries, the comment column displays any comment associated with the library. For objects, the columns display the object name, modification date, size, and any comment associated with the object. You can resize columns by moving the splitter bar between columns. You can sort a column’s contents by Clicking the column header.

Displaying items in the Tree view and the List view

Most of the time, you select a library in the Tree view and display the objects in that library in the List view. You can set a new root or move back and forward in the history of your actions in the List view and the Tree view so libraries or other items display. For information about setting the root, see “Setting the root” on page 42. For information about moving back or forward, see “Moving back, forward, and up one level” on page 42.



About sorting the Name column

When you Click the Name column header repeatedly to sort, the sort happens in four ways: by object type and then name in both ascending and descending order and by object name in both ascending and descending order. You may not easily observe the four ways of sorting if all objects of the same type have names that begin with the same character or set of characters.

Working with libraries

You work with libraries in the Library painter.

Displaying libraries and objects

What you see in the views

In the Tree view, you can expand items and see the folders, libraries, or objects they contain. In the List view, you do not see expansions and contractions of contents. The List view displays only contents. You cannot see any files that are not in libraries.

❖ **To expand or collapse an item in the Tree view:**

- Double-click the item.

If the item contains libraries or objects, they display in the List view.

❖ **To display the contents of an item in the List view:**

- Select the item in the Tree view, or double-Click the item in the List view.

Using drag and drop to expand items

You can drag and drop items to expand them and see the contents:

If you drag an item from a	And drop it in a	This happens
Tree view or List view	List view	The List view sets the item as the root and displays the contents
Tree view or List view	Tree view	The Tree view expands to that item

For example, you can drag a library from the Tree view and drop it in the List view to quickly display the objects the library contains in the List view. For information about using drag and drop to copy or move items, see “Copying, moving, and deleting objects” on page 40.

Using the pop-up menu

Like the other painters, the Library painter has a pop-up menu that provides menu items that apply to the selected item in the Tree view or the List view. For example, from a library’s pop-up menu, you can delete, search, optimize, print the directory, or display library properties. From an object’s pop-up menu, you can edit (go to the painter), copy, move, or delete the object.

Controlling columns that display in the List view

You can control whether to display the last modification date, compilation date, size, and comments (if a comment was created when an object or library was created) in the List view.

❖ **To control the display of columns in the List view:**

- 1 Select Design>Options from the menu bar.
- 2 On the General tab page, select or clear these display items: Modification Date, Compilation Date, Sizes, and Comments.

Selecting objects

In the List view, you can select one or more libraries or objects to act on.

❖ **To select multiple entries:**

- In the List view, use Ctrl+Click (for individual entries) and Shift+Click (for a group of entries).

❖ **To select all entries:**

- In the List view, select an object and then Click the Select All button on the PainterBar.

Filtering the display of objects

You can change which objects display in expanded libraries.

Settings are remembered

DataWindow Designer records your preferences in the Library section of the DataWindow Designer initialization file so that the next time you open the Library painter, the same objects and information are displayed.

Specifying which objects display in all libraries

Initially in the Tree and List views, the Library painter displays all objects in libraries that you expand. You can specify that the Library painter display in all libraries only objects whose names match a specific pattern. For example, you can limit the display to DataWindow objects that begin with emp_.

❖ **To restrict which objects are displayed:**

- 1 Select Design>Options from the menu bar and select the Include tab.
- 2 In the Options dialog box, specify the display criteria:
 - To limit the display to entries that contain specific text in their names, enter the text in the Name box. You can use the wildcard characters question mark (?) and asterisk (*) in the string: ? represents one character, * represents any string of characters. The default is all entries.

- 3 Click OK.

The Options dialog box closes.

- 4 In the Tree view, expand libraries or select a library to display the objects that meet the criteria.

Overriding the choices you made for a specific view

In either the Tree view or the List view, you can override your choice of objects that display in all libraries by selecting a library, displaying the library's pop-up menu, and then clearing or selecting items on the list of objects.

Filtering the display of libraries and folders

In either the Tree view or the List view, you can control what displays when you expand a drive or folder. An expanded drive or folder can display only libraries, only folders, or both.

- ❖ **To control the display of libraries and folders:**
 - In either view, select a drive or folder and then select or clear Libraries and/or Folders from the pop-up menu.

Opening and previewing objects

You can open and preview objects in the current project.

- ❖ **To open an object:**
 - In either the Tree view or the List view, double-click the object, or select Edit from the object's pop-up menu.

DataWindow Designer takes you to the painter for that object and opens the object. You can work on the object and save it as you work. When you close it, you return to the Library painter.

You can preview DataWindow objects from the Library painter.

- ❖ **To preview an object in the Library painter:**
 - Select Run/Preview from the object's pop-up menu.

Copying, moving, and deleting objects

As your needs change, you may want to rearrange the objects in libraries. To do that, you need to be able to copy and move objects between libraries or delete objects that you no longer need.

❖ To copy or move objects using drag and drop:

- 1 In the Tree view or the List view, select the objects you want to copy or move.
- 2 Press Shift to move the entries.
Otherwise, the objects will be copied.
- 3 Drag the objects to a library in either view (or if the contents of a library are displayed in the List view, you can drop the objects there).

If copying, DataWindow Designer replicates the objects. If moving, DataWindow Designer moves the objects and deletes them from the source library. If an object with the same name already exists, DataWindow Designer prompts you and, if you allow it, replaces the existing object with the copied or moved object.

❖ To copy or move objects using a button or menu item:

- 1 Select the objects you want to copy or move to another library.
- 2 Click the Copy button or the Move button on the Painterbar, or select Entry>Copy or Entry>Move from the menu bar.
The Select Library dialog box displays.
- 3 Select the library to which you want to copy or move the objects and Click OK.

❖ To delete objects:

- 1 Select the objects you want to delete.
- 2 Click the Delete button, or select Entry>Delete from the menu bar.
You are prompted to confirm the first deletion.

Being asked for confirmation

By default, DataWindow Designer asks you to confirm each deletion. If you do not want to have to confirm deletions, select Design>Options to open the Options dialog box for the Library painter and clear the Confirm on Delete check box in the General tab page.

DataWindow Designer records this preference as the DeletePrompt variable in the Library section of the *DW.INI* file.

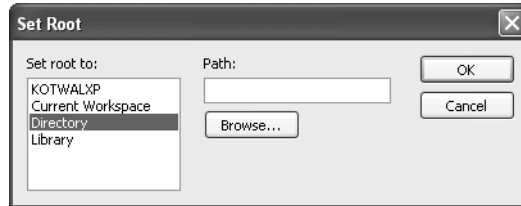
- 3 Click Yes to delete the entry or Yes To All to delete all entries. Click No to skip the current entry and go on to the next selected entry.

Setting the root

In either the Tree view or the List view, you can set the root location of the view.

❖ **To set the root of the current view:**

- 1 In either view, select View>Set Root from the menu bar, or select Set Root from the pop-up menu.



- 2 If you want the root to be a directory or library, type the path or browse to the path.

If you set the root to the Library List in both the Tree view and the List view, only the names of the libraries in the current project display in the Tree view and the List view.

Moving back, forward, and up one level

You can also set a new root by moving back to where you were before, moving forward to where you just were, or for the List view, moving up a level.

❖ **To move back, forward, or up one level:**

- Select View>Back, View>Forward, or View>Up One Level from the menu bar, or select Back, Forward, or Up One Level from the pop-up menu.

The name of the location you are moving back to or forward to is appended to Back and Forward.

Modifying comments

You can use comments to document your objects and libraries. You can associate comments with an object when you first save it in a painter, and you can use the Library painter to:

- Modify existing comments for objects
- Add comments to objects that do not currently have any comments
- Modify the comments for a single object
- Modify comments for multiple objects using a shortcut
- Modify comments for a library

❖ To modify comments for objects:

- 1 In the List view, select the objects you want.
- 2 Select Entry>Properties from the menu bar, or select Properties from the pop-up menu.

DataWindow Designer displays the Properties dialog box. The information that displays is for the first object you selected. You can change existing comments, or, if there are no comments, you can enter new descriptive text.

- 3 Click OK when you have finished with the first object.

If you do not want to change the comments for an object, Click OK. The next object displays.

- 4 Enter comments and Click OK for each object until you have finished.

If you want to stop working on comments before you finish with the objects you selected, Click Cancel. The comments you have entered until the most recent OK are retained. The new comments display in the Library painter workspace.

❖ To modify comments for a library:

- 1 Select the library you want.
- 2 Click the Properties button, or select Library from the pop-up menu.
- 3 Add or modify the comments.

Deleting libraries

❖ **To delete a library:**

- 1 In either the Tree view or the List view, select the library you want to delete.
- 2 Select Library>Delete from the menu bar.

Restriction

You cannot delete libraries in the current library list.

The Delete Library dialog box displays, showing the library you selected.

- 3 Click Yes to delete the library.

The library and all its entries are deleted. You cannot get them back.

Optimizing libraries

You might need to optimize your libraries occasionally. Optimizing removes gaps in libraries and defragments the storage of objects, thus improving performance.

Optimizing affects only layout on disk; it does not affect the contents of the objects. Objects are not recompiled when you optimize a library.

❖ **To optimize a library:**

- 1 In either Tree view or List view, choose the library you want to optimize.
- 2 Select Library>Optimize from the menu bar, or select Optimize from the library's pop-up menu.

DataWindow Designer reorganizes the library structure to optimize object and data storage and index locations. Note that DataWindow Designer does not change the modification date for the library entries. DataWindow Designer saves the unoptimized version as a backup file in the same directory.

The optimized file is created with the default permissions for the drive where it is stored. On some systems new files are not shareable by default. If you see “save of object failed” or “link error” messages after optimizing, check the permissions assigned to the PBL.

If you do not want a backup file

If you do not want to save a backup copy of the library, clear the Save Optimized Backups check box in the Library painter's Design>Options tab dialog box. If you clear this option, the new setting remains in effect until you change it.

Regenerating library entries

When you regenerate an entry, DataWindow Designer recompiles the source form stored in the library and replaces the existing compiled form with the recompiled form.

❖ **To regenerate library entries:**

- 1 Select the entries you want to regenerate.
- 2 Click the Regen button, or select Entry>Regenerate from the menu bar.

DataWindow Designer uses the source to regenerate the library entry and replaces the current compiled object with the regenerated object. The compilation date and size are updated.

Migrating libraries

When you upgrade to a new version of DataWindow Designer, your existing libraries need to be migrated to the new version.

Make sure PBLs are writable

If you make copies of your libraries before you migrate to a new version of DataWindow Designer, make sure that the libraries you will migrate are writable.

The library that is created when you create a new project contains the default application object, which is required for migration. If you add other libraries to the project and delete the default application library, DataWindow Designer cannot migrate the other libraries in the project to the current version.

DataWindow Designer uses project (.dwp) files to manage libraries. The following sample has three libraries: *lib1.pbl*, *lib2.pbl*, and *lib3.pbl*. If you delete the library listed on the applib line, *lib1.pbl*, from the library list, you cannot migrate the project to the current version:

```
Save Format v3.0(19990112)
appname "Default";
applib "lib1.pbl";
LibList "lib1.pbl;lib2.pbl;lib3.pbl";
type "DataWindow Designer";
```

A library must be in the current library list for you to migrate it.

❖ **To migrate libraries:**

- 1 Select Migrate from the pop-up menu for the project in the System Tree.
The Migrate Application dialog box displays.
- 2 Click OK.
DataWindow Designer migrates all objects in the library to the current version.

Exporting and importing entries

You can export DataWindow object definitions to text files. The text files contain all the information that defines the DataWindow objects. The files are virtually identical syntactically to the source forms that are stored in libraries for all objects.

You may want to export object definitions in the following situations:

- You want to store the objects as text files.
- You want to move objects to another computer as text files.

Caution

The primary use of the Export feature is to export source code, not to modify the source. Modifying source in a text file is not recommended.

Later you can import the files back into DataWindow Designer for storage in a library.

❖ To export entries to text files:

- 1 Select the Library entries you want to export.
- 2 Click the Export button.
or
Select Entry>Export from the menu bar.

The Export Library Entry dialog box displays showing the name of the first entry selected for export in the File Name box and the name of the current directory. The current directory is the last directory you selected for saving exported entries or saving a file using the file editor.

DataWindow Designer appends the file extension *.srd*.

- 3 Specify the filename and directory for the export file. Do not change the file extension.
- 4 Specify the encoding type you want the file to use.
- 5 Click OK.

DataWindow Designer converts the entry to the requested file format, stores it with the specified name, then displays the next entry you selected for export.

If a file already exists with the same name, DataWindow Designer displays a message asking whether you want to replace the file. If you say no, you can change the name of the file and then export it, skip the file, or cancel the export of the current file and any selected files that have not been exported.

- 6 Repeat steps 3 and 4 until you have processed all the selected entries.

You cannot see export files in the Library painter

Since export files are text files, the Library painter does not show them; it shows only libraries and directories.

❖ To import text files to library entries:

- 1 Click the Import button.
or
Select Entry>Import from the menu bar.

The Select Import Files dialog box displays showing the current directory and a list of files with the extension *srd* in that directory.

- 2 Select the files you want to import. Use Shift+Click or Ctrl+Click to select multiple files.

- 3 Click Open.

The Import File Into Library Entry dialog box displays listing the libraries in the library search path.

- 4 Select the library you want to import the text files to.
- 5 Click OK.

DataWindow Designer converts the specified text files to DataWindow Designer format, regenerates (recompiles) the objects, stores the entries in the specified library, and updates the entries' timestamps.

If a library entry with the same name already exists, DataWindow Designer replaces it with the imported entry.

Caution

When you import an entry with the same name as an existing entry, the old entry is deleted before the import takes place. If an import fails, the old object will already be deleted.

Creating a deployment library (PBD file)

The DataWindow objects you build in DataWindow Designer can be distributed in a deployment library (PBD). Other DataWindow Designer, PowerBuilder, or InfoMaker users can open a PBL and copy or modify the DataWindow objects in it. They cannot open and modify objects in a PBD.

To create PBDs for all the PBLs in your project, select Build Deployment from the pop-up menu for the project in the System Tree. PBDs have the same name as PBLs with the extension .PBD.

Including additional resources in the PBD file

When building a runtime library (PBD file), DataWindow Designer does not inspect the objects; it simply removes the source form of the objects from the library (leaving only the runtime version). Therefore, if any of the objects in the library use resources (pictures and pointers) and you do not want to provide these resources separately, you must list the resources in a resource file (PBR file). Doing so enables DataWindow Designer to include the resources in the runtime library when it builds it.

A PBR file is an ASCII text file in which you list resource names (such as BMP, CUR, RLE, and WMF files) and DataWindow objects.

❖ **To create a PowerBuilder resource file (PBR file):**

- Using a text editor, create a text file that lists all resource files referenced in your DataWindow objects. List the names of each resource, one resource on each line, then save the list as a file with the extension PBR.

Naming resources

If the resource file is in the current folder, you can simply list the file, such as:

```
FROWN.BMP
```

If the resource file is in a different folder, include the path to the file, such as:

```
C:\BITMAPS\FROWN.BMP
```

What happens at runtime

When a resource such as a bitmap is referenced at runtime, the DataWindow server first looks in the PBDs that are defined for the application. Failing that, it looks in folders in the search path for the file.

Creating a library directory report

A library directory report lists all entries in the current library, showing the following information for all objects in the library, ordered by object type:

- Name of object
- Modification date and time
- Size (of compiled object)
- Comments

❖ **To create the library directory report:**

- 1 Select the current library.
- 2 Select Library>Print Directory from the menu bar.

DataWindow Designer sends the library directory report to the printer specified under File>Printer Setup in the menu bar.

Managing the Database

About this chapter

This chapter describes how to manage the database from within DataWindow Designer.

Contents

Topic	Page
Working with database components	51
Managing databases	54
Using the Database painter	55
Creating and deleting an Adaptive Server Anywhere database	60
Working with tables	62
Working with keys	75
Working with indexes	79
Working with database views	81
Manipulating data	86
Creating and executing SQL statements	92
Controlling access to the current database	96

Before you begin

You work with relational databases in DataWindow Designer. If you are not familiar with relational databases, you might want to consult an introductory text, such as *A Database Primer* by C. J. Date.

Working with database components

A database is an electronic storage place for data. Databases are designed to ensure that data is valid and consistent and that it can be accessed, modified, and shared.

A database management system (DBMS) governs the activities of a database and enforces rules that ensure data integrity. A *relational* DBMS stores and organizes data in tables.

How you work with databases in DataWindow Designer

You can use DataWindow Designer to work with the following database components:

- Tables and columns
- Keys
- Indexes
- Database views
- Extended attributes
- Additional database components

Tables and columns

A database usually has many tables, each of which contains rows and columns of data. Each row in a table has the same columns, but a column's value for a particular row could be empty or NULL if the column's definition allows it.

Tables often have relationships with other tables. For example, in the EAS Demo DB included with DataWindow Designer, the Department table has a Dept_id column, and the Employee table also has a Dept_id column that identifies the department in which the employee works. When you work with the Department table and the Employee table, the relationship between them is specified by a join of the two tables.

Keys

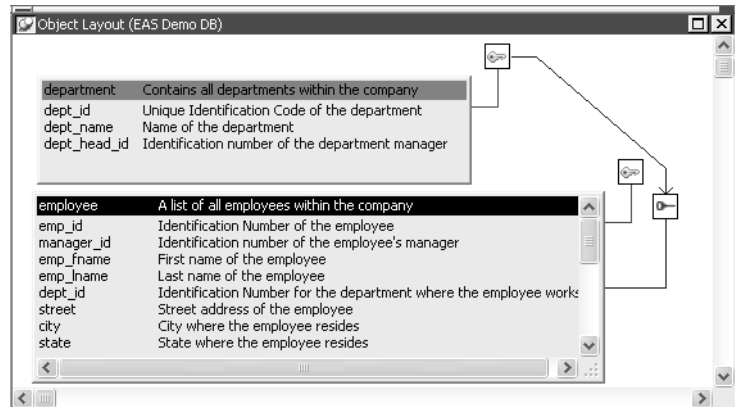
Relational databases use keys to ensure database integrity.

Primary keys A primary key is a column or set of columns that uniquely identifies each row in a table. For example, two employees may have the same first and last names, but they have unique ID numbers. The Emp_id column in the Employee table is the primary key column.

Foreign keys A foreign key is a column or set of columns that contains primary key values from another table. For example, the Dept_id column is the primary key column in the Department table and a foreign key in the Employee table.

Key icons In DataWindow Designer, columns defined as keys are displayed with key icons that use different shapes and colors for primary and foreign. DataWindow Designer automatically joins tables that have a primary/foreign key relationship, with the join on the key columns.

In the following illustration there is a join on the dept_id column, which is a primary key for the department table and a foreign key for the employee table:



For more information, see “Working with keys” on page 75.

Indexes

An index is a column or set of columns you identify to improve database performance when searching for data specified by the index. You index a column that contains information you will need frequently. Primary and foreign keys are special examples of indexes.

You specify a column or set of columns with unique values as a unique index, represented by an icon with a single key.

You specify a column or set of columns that has values that are not unique as a duplicate index, represented by an icon with two file cabinets.

For more information, see “Working with indexes” on page 79.

Database views

If you often select data from the same tables and columns, you can create a database view of the tables. You give the database view a name, and each time you refer to it the associated SELECT command executes to find the data.

Database views are listed in the Objects view of the Database painter and can be displayed in the Object Layout view, but a database view does not physically exist in the database in the same way that a table does. Only its definition is stored in the database, and the view is re-created whenever the definition is used.

Database administrators often create database views for security purposes. For example, a database view of an Employee table that is available to users who are not in Human Resources might show all columns except Salary.

For more information, see “Working with database views” on page 81.

Extended attributes Extended attributes enable you to store information about a table's columns in special system tables. Unlike tables, keys, indexes, and database views (which are DBMS-specific), extended attributes are DataWindow Designer-specific. The most powerful extended attributes determine the edit style, display format, and validation rules for the column.

For more information about extended attributes, see "Specifying column extended attributes" on page 66. For more information about the extended attribute system tables, see Appendix B, "The Extended Attribute System Tables."

Additional database components Depending on the database to which you are connected and on your user privileges, you might be able to view or work with a variety of additional database components through DataWindow Designer. These components might include:

- Driver information
- Groups
- Metadata types
- Procedures and functions
- Users
- Logins

For example, driver information is relevant to ODBC connections. It lists all the ODBC options associated with the ODBC driver, allowing you to determine how the ODBC interface will behave for a given connection. Login information is listed for Adaptive Server Enterprise (SYC) database connections. Information about groups and users is listed for several of the databases and allows you to add new users and groups and maintain passwords for existing users.

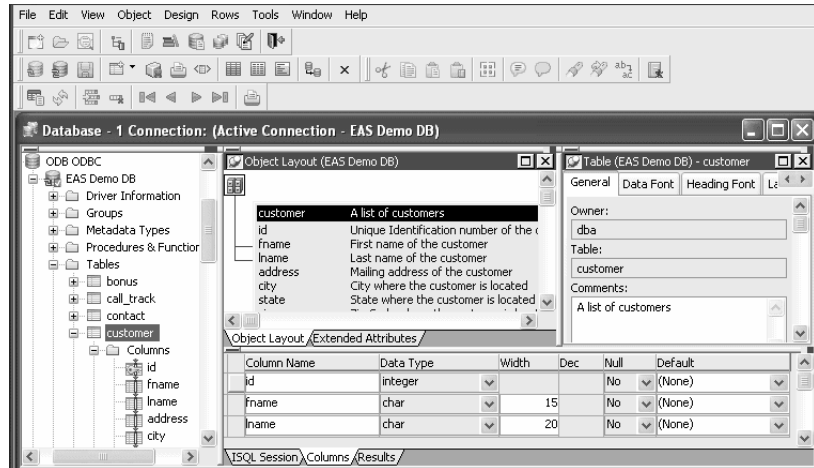
Managing databases

DataWindow Designer supports many database management systems (DBMSs). For the most part, you work the same way in DataWindow Designer for each DBMS, but because each DBMS provides some unique features (which DataWindow Designer makes use of), there are some issues that are specific to a particular DBMS. For complete information about using your DBMS, see *Connecting to Your Database*.

What you can do	<p>Using the Database painter, you can do the following in any DBMS to which you have been given access by the database administrator:</p> <ul style="list-style-type: none">• Modify local table and column properties• Retrieve, change, and insert data• Create new local tables or modify existing tables
Setting the database connection	<p>When you open a painter that communicates with the database (such as the Database painter or DataWindow painter), DataWindow Designer connects you to the database you used last if you are not already connected. If the connection to the default database fails, the painter still opens.</p> <p>If you do not want to connect to the database you used last, you can deselect the Connect to Default Profile option in the Database Preferences dialog box.</p>
Changing the database connection	<p>You can change to a different database at any time. You can have several database connections open at a time, although only one connection can be active. The database components for each open connection are listed in the Objects view.</p> <p>The Database painter title bar displays the number of open connections and shows which is active. The title bar for each view displays the connection with which it is currently associated. You can change the connection associated with a view by dragging the profile name for a different connection onto the view.</p> <p>For more about changing the database you are connected to, see <i>Connecting to Your Database</i>.</p>
Creating and deleting databases	<p>When you are connected to Adaptive Server Anywhere, you can create a new database or delete an existing database using the Database painter.</p> <p>For all other DBMSs, creating and deleting a database is an administrative task that you cannot do within DataWindow Designer.</p>

Using the Database painter

About the painter	<p>To open the Database painter, click the Database button in the PowerBar.</p> <p>Like the other DataWindow Designer painters, the Database painter contains a menu bar, customizable PainterBars, and several views. All database-related tasks that you can do in DataWindow Designer can be done in the Database painter.</p>
-------------------	---



Views in the Database painter

Table 3-1 lists the views available in the Database painter.

Table 3-1: Database painter views

View	Description
Activity Log	Displays the SQL syntax generated by the actions you execute.
Columns	Used to create and/or modify a table's columns (formerly done in the Table painter).
Extended Attributes	Lists the display formats, edit styles, and validation rules defined for the selected database connection.
Interactive SQL	Used to build, execute or explain SQL. (In DataWindow Designer 6 and earlier, some of this functionality was in the Database Administration painter.)
Object Details	Displays an object's properties. For some objects, properties are read-only; for others, properties can be modified. This view is analogous to the Properties view in other painters.
Object Layout	Displays a graphical representation of tables.
Objects	Lists the database interfaces and profiles. For an active database connection, might also list all or some of the following objects associated with that database: groups, metadata types, procedures and functions, tables, columns, primary and foreign keys, indexes, users, views, driver information, and utilities (the database components listed depend on the database and your user privileges).
Results	Displays data in a grid, table, or freeform format.

Dragging and dropping

You can select certain database objects from the Objects view and drag them to the Object Details, Object Layout, Columns, and/or ISQL views. Just position the pointer on the database object's icon and drag it to the appropriate view.

Table 3-2: Using drag and drop in the Database painter

Object	Can be dragged to
Driver, group, metadata type, procedure or function, table, column, user, primary or foreign key, index	Object Details view
Table or view	Object Layout view
Table or column	Columns view
Procedure or view	ISQL view

Database painter tasks

Table 3-3 describes how to do some basic tasks in the Database painter. Most of these tasks begin in the Objects view. Many can be accomplished by dragging and dropping objects into different views. If you prefer, you can use buttons or menu selections from the main bar or from pop-up menus.

Table 3-3: Common tasks in the Database painter

To	Do this
Modify a database profile	Highlight a database profile and select Properties from the Object or pop-up menu or use the Properties button. Alternatively, use the Import and Export Profiles menu selections to copy profiles. For more information, see the section on importing and exporting database profiles in <i>Connecting to Your Database</i> .
Connect to a database	Highlight a database profile and then select Connect from the File or pop-up menu or use the Connect button. With File>Recent Connections, you can review and return to earlier connections. You can also make database connections using the Database Profile button.
Create new profiles, tables, views, columns, keys, indexes, or groups	Highlight the database object and select New from the Object or pop-up menu or use the Create button.
Modify database objects	Drag the object to the Object Details view.
Graphically display tables	Drag the table icon from the list in the Objects view to the Object Layout view, or highlight the table and select Add To Layout from the Object or pop-up menu.
Manipulate data	Highlight the table and select Grid, Tabular, or Freeform from the Object>Data menu or the pop-up menu Edit Data item, or use the appropriate Data Manipulation button.

To	Do this
Build, execute or explain SQL	Use the ISQL view to build SQL statements. Use the Paste SQL button to paste SELECT, INSERT, UPDATE, and DELETE statements or type them directly into the view's workspace. To execute or explain SQL, select Execute SQL and Explain SQL from the Design or pop-up menu.
Define or modify extended attributes	Select from the Object>Insert menu the type of extended attribute you want to define or modify, or highlight the extended attribute from the list in the Extended Attributes view and select New or Properties from the pop-up menu.
Specify extended attributes for a column	Drag the column to the Object Details view and select the Extended Attributes tab.
Access database utilities	Double-click a utility in the Objects view to launch it.
Log your work	Select Design>Start Log from the menu bar. To see the SQL syntax generated, display the Activity Log view.

Modifying database preferences

To modify database preferences, select Design>Options from the menu bar. Some preferences are specific to the database connection; others are specific to the Database painter.

Preferences on the General property page

The Connect To Default Profile, Shared Database Profiles, Keep Connection Open, Use Extended Attributes, and Read Only preferences are specific to the database connection.

The remaining preferences are specific to the Database painter. For information about modifying these preferences, see *Connecting to Your Database*.

Table 3-4: Database painter preferences

Database preference	What DataWindow Designer does with the specified preference
Columns in the Table List	When DataWindow Designer displays tables graphically, eight table columns display unless you change the number of columns.
SQL Terminator Character	DataWindow Designer uses the semicolon as the SQL statement terminator unless you enter a different terminator character in the box. Make sure that the character you choose is not reserved for another use by your database vendor. For example, using the slash character (/) causes compilation errors with some DBMSs.

Database preference	What DataWindow Designer does with the specified preference
Refresh Table List	When DataWindow Designer first displays a table list, DataWindow Designer retrieves the table list from the database and displays it. To save time, DataWindow Designer saves this list internally for reuse to avoid regeneration of very large table lists. The table list is refreshed every 30 minutes (1800 seconds) unless you specify a different refresh rate.

Preferences on the Object Colors property page

You can set colors separately for each component of the Database painter's graphical table representation: the table header, columns, indexes, primary key, foreign keys, and joins. Set a color preference by selecting a color from a drop-down list.

You can design custom colors that you can use when you select color preferences. To design custom colors, select Design>Custom Colors from the menu bar and work in the Custom Colors dialog box.

Logging your work

As you work with your database, you generate SQL statements. As you define a new table, for example, DataWindow Designer builds a SQL CREATE TABLE statement internally. When you click the Create button, DataWindow Designer sends the SQL statement to the DBMS to create the table. Similarly, when you add an index, DataWindow Designer builds a CREATE INDEX statement.

You can see all SQL generated in a Database painter session in the Activity Log view. You can also save this information to a file. This allows you to have a record of your work and makes it easy to duplicate the work if you need to create the same or similar tables in another database.

❖ To start logging your work:

- 1 Open the Database painter.
- 2 Select Start Log from the Design menu or the pop-up menu in the Activity Log view.

DataWindow Designer begins sending all generated syntax to the Activity Log view.

❖ **To stop the log:**

- Select Stop Log from the Design menu or the pop-up menu in the Activity Log view.

DataWindow Designer stops sending the generated syntax to the Activity Log view. Your work is no longer logged.

❖ **To save the log to a permanent text file:**

- 1 Select Save or Save As from the File menu.
- 2 Name the file and click Save. The default file extension is *SQL*, but you can change that if you want to.

Submitting the log to your DBMS

You can open a saved log file and submit it to your DBMS in the ISQL view. For more information, see “Building and executing SQL statements” on page 92.

Creating and deleting an Adaptive Server Anywhere database

In DataWindow Designer you work within an existing database. With one exception, creating or deleting a database is an administrative task that is not performed directly in DataWindow Designer. The one exception is that you can create and delete a local Adaptive Server Anywhere database from within DataWindow Designer.

For information about creating and deleting other databases, see your DBMS documentation.

❖ **To create a local Adaptive Server Anywhere database:**

- 1 From the Objects view, launch the Create ASA Database utility included with the ODBC interface.

The Create Adaptive Server Anywhere Database dialog box displays.

- 2 In the Database Name box, specify the file name and path of the database you are creating.

If you do not provide a file extension, the database file name is given the extension *DB*.

- 3 Define other properties of the database as needed.

If you are using a non-English database, you can specify a code page in the Collation Sequence box.

For complete information about filling in the dialog box, click the Help button in the dialog box.

- 4 Click OK.

When you click OK, DataWindow Designer does the following:

- Creates a database with the specified name in the specified directory or folder. If a database with the same name exists, you are asked whether you want to replace it.
- Adds a data source to the *ODBC.INI* key in the registry. The data source has the same name as the database unless one with the same name already exists, in which case a suffix is appended.
- Creates a database profile and adds it to the registry. The profile has the same name as the database unless one with the same name already exists, in which case a suffix is appended.
- Connects to the new database.

❖ **To delete a local Adaptive Server Anywhere database:**

- 1 Open the Database painter.
- 2 From the Objects view, launch the Delete ASA Database utility included with the ODBC interface.
- 3 Select the database you want to delete and select Open.
- 4 Click Yes to delete the database.

When you click Yes, DataWindow Designer deletes the specified database.

Working with tables

When you open the Database painter, the Object view lists all tables in the current database that you have access to (including tables that were not created using DataWindow Designer). You can create a new table or alter an existing table. You can also modify table properties and work with indexes and keys.

Creating a new table from scratch

In DataWindow Designer, you can create a new table in any database to which DataWindow Designer is connected.

❖ **To create a table in the current database:**

1 Do one of the following:

- Click the Create Table button.
- Right-click in the Columns view and select New Table from the pop-up menu.
- Right-click Tables in the Objects view and select New Table from the pop-up menu.
- Select Insert>Table from the Object menu.

The new table template displays in the Columns view. What you see in the view is DBMS-dependent. You use this template to specify each column in the table. The insertion point is in the Column Name box for the first column.

2 Enter the required information for this column.

For what to enter in each field, see “Specifying column definitions” on page 64.

As you enter information, use the Tab key to move from place to place in the column definition. After defining the last item in the column definition, press the Tab key to display the work area for the next column.

3 Repeat step 2 for each additional column in your table.

4 (Optional) Select Object>Pending SQL from the menu bar or select Pending SQL from the pop-up menu to see the pending SQL syntax.

If you have not already named the table, you must provide a name in the dialog box that displays. To hide the SQL syntax and return to the table columns, select Object>Pending Syntax from the menu bar.

- 5 Click the Save button or select Save from the File or pop-up menu, then enter a name for the table in the Create New Table dialog box.

DataWindow Designer submits the pending SQL syntax statements it generated to the DBMS, and the table is created. The new table is displayed in the Object Layout view.

About saving the table

If you make changes after you save the table and before you close it, you see the pending changes when you select Pending SQL again. When you click Save again, DataWindow Designer submits a DROP TABLE statement to the DBMS, recreates the table, and applies all changes that are pending. Clicking Save many times can be time consuming when you are working with large tables, so you might want to save only when you have finished.

- 6 Specify extended attributes for the columns.

For what to enter in each field, see “Specifying column extended attributes” on page 66.

Creating a new table from an existing table

You can create a new table that is similar to an existing table very quickly by using the Save Table As menu option.

❖ **To create a new table from an existing table:**

- 1 Open the existing table in the Columns view by dragging and dropping it or selecting Alter Table from the pop-up menu.
- 2 Right-click in the Columns view and select Save Table As from the pop-up menu.
- 3 Enter a name for the new table and then the owner’s name, and click OK. The new table appears in the Object Layout view and the Columns view.
- 4 Make whatever changes you want to the table definition.
- 5 Save the table.
- 6 Make changes to the table’s properties in the Object Details view.

For more information about modifying table properties, see “Specifying table and column properties” on page 64.

Specifying column definitions

When you create a new table, you must specify a definition for each column. The fields that display for each column in the Columns view depend on your DBMS. You might not see all of the following fields, and the values that you can enter are dependent on the DBMS.

For more information, see your DBMS documentation.

Table 3-5: Defining columns in the Columns view in the Database painter

Field	What you enter
Column Name	(Required) The name by which the column will be identified.
Data Type	(Required) Select a datatype from the drop-down list. All datatypes supported by the current DBMS are displayed in the list.
Width	For datatypes with variable widths, the number of characters in the field.
Dec	For numeric datatypes, the number of decimal places to display.
Null	Select Yes or No from the Null drop-down list to specify whether NULLs are allowed in the column. Specifying No means the column cannot have NULL values; users must supply a value. No is the default in a new table.
Default	The value that will be placed in a column in a row that you insert into a DataWindow object. The drop-down list has built-in choices, but you can type any other value. For an explanation of the built-in choices, see your DBMS documentation.

Specifying table and column properties

After you create and save a table, you can specify the properties of the table and of any or its columns. Table properties include the fonts used for headers, labels, and data, and a comment that you can associate with the table. Column properties include the text used for headers and labels, display formats, validation rules, and edit styles used for data (also known as a column's extended attributes), and a comment you can associate with the column.

Specifying table properties

In addition to adding a comment to associate with the table, you can choose the fonts that will be used to display information from the table in a DataWindow object. You can specify the font, point size, color, and style.

❖ To specify table properties:

1 Do one of the following:

- Highlight the table in either the Objects view or the Object Layout view and select Properties from the Object or pop-up menu.
- Click the Properties button.
- Drag and drop the table to the Object Details view.

The properties for the table display in the Object Details view.

2 Select a tab and specify properties:

Select this tab	To modify this property
General	Comments associated with the table
Data Font	Font for data retrieved from the database and displayed in the Results view by clicking a Data Manipulation button
Heading Font	Font for column identifiers used in grid, tabular, and n-up DataWindow objects displayed in the Results view by clicking a Data Manipulation button
Label Font	Font for column identifiers used in freeform DataWindow objects displayed in the Results view by clicking a Data Manipulation button

3 Right-click on the Object Details view and select Save Changes from the pop-up menu.

Any changes you made in the Object Details view are immediately saved to the table definition.

Specifying column extended attributes

In addition to adding a comment to associate with a column, you can specify extended attributes for each column. An extended attribute is information specific to DataWindow Designer that enhances the definition of the column.

❖ To specify extended attributes:

1 Do one of the following:

- Highlight the column in either the Objects view or the Object Layout view and select Properties from the Object or pop-up menu.
- Click the Properties button.
- Drag and drop the column to the Object Details view.

2 Select a tab and specify extended attribute values:

Select this tab	To modify these extended attributes
General	Column comments.
Headers	Label text used in free-form DataWindow objects Header text used in tabular, grid, or n-up DataWindow objects
Display	How the data is formatted in a DataWindow object as well as display height, width, and position. For example, you can associate a display format with a Revenue column so that its data displays with a leading dollar sign and negative numbers display in parentheses.
Validation	Criteria that a value must pass to be accepted in a DataWindow object. For example, you can associate a validation rule with a Salary column so that you can enter a value only within a particular range. The initial value for the column. You can select a value from the drop-down list. The initial value must be the same datatype as the column, must pass validation, and can be NULL only if NULL is allowed for the column.
Edit Style	How the column is presented in a DataWindow object. For example, you can display column values as radio buttons or in a drop-down list.

3 Right-click on the Column property sheet and select Save Changes from the pop-up menu.

Any changes you made in the property sheet are immediately saved to the table definition.

Overriding definitions

In the DataWindow painter, you can override the extended attributes specified in the Database painter for a particular DataWindow object.

How the information is stored

Extended attributes are stored in the DataWindow Designer system tables in the database. DataWindow Designer uses the information to display, present, and validate data in the Database painter and in DataWindow objects. When you create a view in the Database painter, the extended attributes of the table columns used in the view are used by default.

About display formats, edit styles, and validation rules

In the Database painter, you create display formats, edit styles, and validation rules. Whatever you create is then available for use with columns in tables in the database. You can see all the display formats, edit styles, and validation rules defined for the database in the Extended Attributes view.

For more information about defining, maintaining, and using these extended attributes, see Chapter 8, “Displaying and Validating Data.”

About headings and labels

By default, DataWindow Designer uses the column names as labels and headings, replacing any underscore characters with spaces and capitalizing each word in the name. For example, the default heading for the column Dept_name is Dept Name. To define multiple-line headings, press Ctrl+Enter to begin a new line.

Specifying additional properties for character columns

You can also set two additional properties for character columns on the Display property page: Case and Picture.

Specifying the displayed case

You can specify whether DataWindow Designer converts the case of characters for a column in a DataWindow object.

❖ **To specify how character data should be displayed:**

- On the Display property page, select a value in the Case drop-down list:

Value	Meaning
Any	Characters are displayed as they are entered
UPPER	Characters are converted to uppercase
lower	Characters are converted to lowercase

Specifying a column as a picture

You can specify that a character column can contain names of picture files.

❖ **To specify that column values are names of picture files:**

- 1 On the Display property page, select the Picture check box.

When the Picture check box is selected, DataWindow Designer expects to find picture file names in the column and displays the contents of the picture file—not the name of the file—in reports and DataWindow objects.

Because DataWindow Designer cannot determine the size of the image until runtime, it sets both display height and display width to 0 when you select the Picture check box.

- 2 Enter the size and the justification for the picture (optional).

Altering a table

After a table is created, how you can alter the table depends on your DBMS.

You can always:

- Add or modify DataWindow Designer-specific extended attributes for columns
- Delete an index and create a new index

You can never:

- Insert a column between two existing columns
- Prohibit null values for an appended column
- Alter an existing index

Some DBMSs let you do the following, but others do not:

- Append columns that allow null values
- Increase or decrease the number of characters allowed for data in an existing column
- Allow null values
- Prohibit null values in a column that allowed null values

Database painter is DBMS aware

The Database painter grays out or notifies you about actions that your DBMS prohibits.

For complete information about what you can and cannot do when you modify a table in your DBMS, see your DBMS documentation.

❖ **To alter a table:**

- 1 Highlight the table and select Alter Table from the pop-up menu.

Opening multiple instances of tables

You can open another instance of a table by selecting Columns from the View menu. Doing this is helpful when you want to use the Database painter's cut, copy, and paste features to cut or copy and paste between tables.

The table definition displays in the Columns view (this screen shows the Employee table).

Column Name	Data Type	Width	Dec	Null	Default
emp_id	integer			No	(None)
manager_id	integer			Yes	(None)
emp_fname	char	20		No	(None)
emp_lname	char	20		No	(None)
dept_id	integer			No	(None)
street	char	40		No	(None)
city	char	20		No	(None)
state	char	4		No	(None)
zip_code	char	9		No	(None)

- 2 Make the changes you want in the Columns view or in the Object Details view.
- 3 Select Save Table or Save Changes.

DataWindow Designer submits the pending SQL syntax statements it generated to the DBMS, and the table is modified.

Cutting, copying, and pasting columns

In the Database painter, you can use the Cut, Copy, and Paste buttons in the PainterBar (or Cut, Copy, and Paste from the Edit or pop-up menu) to cut, copy, and paste one column at a time within a table or between tables.

❖ **To cut or copy a column within a table:**

- 1 Put the insertion point anywhere in the column you want to cut or copy.
- 2 Click the Cut or Copy button in the PainterBar.

❖ **To paste a column within a table:**

- 1 Put the insertion point in the column you want to paste to.

If you are changing an existing table, put the insertion point in the last column of the table. If you try to insert a column between two columns, you get an error message. To an existing table, you can only append a column. If you are defining a new table, you can paste a column anywhere.

- 2 Click the Paste button in the PainterBar.

❖ **To paste a column to a different table:**

- 1 Open another instance of the Columns view and use Alter Table to display an existing table or click New to create a new table.
- 2 Put the insertion point in the column you want to paste to.
- 3 Click the Paste button in the PainterBar.

Closing a table

You can remove a table from a view by selecting Close or Reset View from its pop-up menu. This action only removes the table from the Database painter view. It does not drop (remove) the table from the database.

Dropping a table

Dropping removes the table from the database.

❖ **To drop a table:**

- 1 Select Drop Table from the table's pop-up menu or select Object>Delete from the menu bar.
- 2 Click Yes.

Deleting orphaned table information

If you drop a table outside DataWindow Designer, information remains in the system tables about the table, including extended attributes for the columns.

❖ **To delete orphaned table information from the extended attribute system tables:**

- Select Design>Synch Extended Attributes from the menu bar and click Yes.

If you try to delete orphaned table information and there is none, a message tells you that synchronization is not necessary.

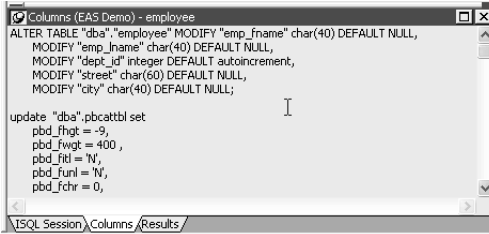
Viewing pending SQL changes

As you create or alter a table definition, you can view the pending SQL syntax changes that will be made when you save the table definition.

❖ **To view pending SQL syntax changes:**

- Right-click the table definition in the Columns view and select Pending Syntax from the pop-up menu.

DataWindow Designer displays the pending changes to the table definition in SQL syntax:



```
Columns (EAS Demo) - employee
ALTER TABLE "dba"."employee" MODIFY "emp_fname" char(40) DEFAULT NULL,
MODIFY "emp_lname" char(40) DEFAULT NULL,
MODIFY "dept_id" integer DEFAULT autoincrement,
MODIFY "street" char(60) DEFAULT NULL,
MODIFY "city" char(40) DEFAULT NULL;

update "dba".pbcattbl set
  pbd_fhgt = -9,
  pbd_fwgt = 400,
  pbd_fitl = 'N',
  pbd_furl = 'N',
  pbd_fchr = 0,
```

The SQL statements execute only when you save the table definition or reset the view and then tell DataWindow Designer to save changes.

Copying, saving, and printing pending SQL changes

When you are viewing pending SQL changes, you can:

- Copy pending changes to the clipboard
- Save pending changes to a file
- Print pending changes

To copy, save, or print only part of the SQL syntax

Select the part of the SQL syntax you want before you copy, save, or print.

- ❖ **To copy the SQL syntax to the clipboard:**
 - In the Pending Syntax view, click the Copy button or select Copy from the pop-up menu.

- ❖ **To save SQL syntax for execution at a later time:**
 - 1 In the Pending Syntax view, Select File>Save As.
The Save Syntax to File dialog box displays.
 - 2 Navigate to the folder where you want to save SQL, name the file, and then click the Save button.

At a later time, you can import the SQL file into the Database painter and execute it.

- ❖ **To print pending table changes:**
 - While viewing the pending SQL syntax, click the Print button or select Print from the File menu.

- ❖ **To display columns in the Columns view:**
 - Select Object>Pending Syntax from the menu bar.

Printing the table definition

You can print a report of the table's definition at any time, whether or not the table has been saved. The Table Definition Report contains information about the table and each column in the table, including the extended attributes for each column.

- ❖ **To print the table definition:**
 - Select Print or Print Definition from the File or pop-up menu or click the Print button.

Exporting table syntax

You can export the syntax for a table to the log. This feature is useful when you want to create a backup definition of the table before you alter it or when you want to create the same table in another DBMS.

To export to another DBMS, you must have the DataWindow Designer interface for that DBMS.

❖ **To export the syntax of an existing table to a log:**

- 1 Select the table in the painter workspace.
- 2 Select Export Syntax from the Object menu or the pop-up menu.

If you selected a table and have more than one DBMS interface installed, the DBMS dialog box displays. If you selected a view, DataWindow Designer immediately exports the syntax to the log.

- 3 Select the DBMS to which you want to export the syntax.
- 4 If you selected ODBC, specify a data source in the Data Sources dialog box.
- 5 Supply any information you are prompted for.

DataWindow Designer exports the syntax to the log. Extended attribute information (such as validation rules used) for the selected table is also exported. The syntax is in the format required by the DBMS you selected.

For more information about the log, see “Logging your work” on page 59.

About system tables

Two kinds of system tables exist in the database:

- System tables provided by your DBMS (for more information, see your DBMS documentation)
- DataWindow Designer extended attribute system tables

About DataWindow Designer system tables

DataWindow Designer stores extended attribute information you provide when you create or modify a table (such as the text to use for labels and headings for the columns, validation rules, display formats, and edit styles) in system tables. These system tables contain information about database tables and columns. Extended attribute information extends database definitions.

In the Employee table, for example, one column name is Emp_name. A label and a heading for the column are defined for DataWindow Designer to use in DataWindow objects. The column label is defined as Last Name:. The column heading is defined as Last Name. The label and heading are stored in the PBCatCol table in the extended attribute system tables.

The extended attribute system tables are maintained by DataWindow Designer, and only DataWindow Designer users can enter information into them. Table 3-6 lists the extended attribute system tables. For more information, see Appendix B, “The Extended Attribute System Tables.”

Table 3-6: Extended attribute system tables

This system table	Stores this extended attribute information
PBCatCol	Column data such as name, header and label for reports and DataWindow objects, and header and label positions
PBCatEdt	Edit style names and definitions
PBCatFmt	Display format names and definitions
PBCatTbl	Table data such as name, fonts, and comments
PBCatVld	Validation rule names and definitions

Opening and displaying system tables

You can open system tables like other tables, in the Database painter.

By default, DataWindow Designer shows only user-created tables in the Objects view. If you highlight Tables and select Show System Tables from the pop-up menu, DataWindow Designer also displays system tables.

Creating and editing Sybase Adaptive Server temporary tables

You can create and edit temporary tables in the Database painter, SQL painter, or DataWindow painter when you use the PowerBuilder SYC native driver to connect to an ASE database. Temporary tables persist for the duration of a database connection, residing in a special database called “tempdb”.

You add a temporary table to the tempdb database by assigning a name that starts with the # character when you create a new table in a PowerBuilder painter. (Temporary tables must start with the # character.)

After you create a temporary table, you can create indexes and a primary key for the table. If you define a unique index or primary key, you can execute INSERT, UPDATE, and DELETE statements for a temporary table. Selecting Edit Data from the pop-up menu of a temporary table retrieves data that you store in that table.

Standard catalog query limitations

When you click Refresh from the pop-up menu for the Tables node in the Database painter or the Objects view of the DataWindow painter, the list of tables displays temporary tables even though they exist only in the tempdb database. However, once you refresh table definitions from the database, the Objects view can no longer list the index or primary key information of the temporary tables, and the Layout view can no longer display that information graphically.

Working with keys

Why you should use keys

If your DBMS supports primary and foreign keys, you can work with the keys in DataWindow Designer.

If your DBMS supports them, you should use primary and foreign keys to enforce the referential integrity of your database. That way you can rely on the DBMS to make sure that only valid values are entered for certain columns instead of having to write code to enforce valid values.

For example, say you have two tables called Department and Employee. The Department table contains the column Dept_Head_ID, which holds the ID of the department's manager. You want to make sure that only valid employee IDs are entered in this column. The only valid values for Dept_Head_ID in the Department table are values for Emp_ID in the Employee table.

To enforce this kind of relationship, you define a foreign key for Dept_Head_ID that points to the Employee table. With this key in place, the DBMS disallows any value for Dept_Head_ID that does not match an Emp_ID in the Employee table.

For more about primary and foreign keys, consult a book about relational database design or your DBMS documentation.

What you can do in the Database painter

You can work with keys in the following ways:

- Look at existing primary and foreign keys
- Open all tables that depend on a particular primary key
- Open the table containing the primary key used by a particular foreign key
- Create, alter, and drop keys

For the most part, you work with keys the same way for each DBMS that supports keys, but there are some DBMS-specific issues. For complete information about using keys with your DBMS, see your DBMS documentation.

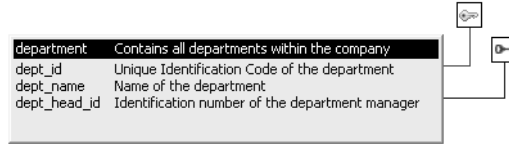
Viewing keys

Keys can be viewed in several ways:

- In the expanded tree view of a table in the Objects view
- As icons connected by lines to a table in the Object Layout view

In the following picture, the Department table has two keys:

- A primary key (on dept_id)
- A foreign key (on dept_head_id)



If you cannot see the lines

If the color of your window background makes it hard to see the lines for the keys and indexes, you can set the colors for each component of the Database painter's graphical table representation, including keys and indexes. For information, see “Modifying database preferences” on page 58.

Opening related tables

When working with tables containing keys, you can easily open related tables.

❖ **To open the table that a particular foreign key references:**

- 1 Display the foreign key pop-up menu.
- 2 Select Open Referenced Table.

❖ **To open all tables referencing a particular primary key:**

- 1 Display the primary key pop-up menu.
- 2 Select Open Dependent Table(s).

DataWindow Designer opens and expands all tables in the database containing foreign keys that reference the selected primary key.

Defining primary keys

If your DBMS supports primary keys, you can define them in DataWindow Designer.

❖ **To create a primary key:**

- 1 Do one of the following:
 - Highlight the table for which you want to create a primary key and click the Create Primary Key drop-down toolbar button in PainterBar1.
 - Select Object>Insert>Primary Key from the main menu or New>Primary Key from the pop-up menu.

- Expand the table's tree view, right-click Primary Key, and select New Primary Key from the pop-up menu.

The Primary Key properties display in the Object Details view.

- 2 Select one or more columns for the primary key.

Columns that are allowed in a primary key

Only a column that does not allow null values can be included as a column in a primary key definition. If you choose a column that allows null values, you get a DBMS error when you save the table. In DBMSs that allow rollback for Data Definition Language (DDL), the table definition is rolled back. In DBMSs that do not allow rollback for DDL, the painter is refreshed with the current definition of the table.

- 3 Specify any information required by your DBMS.

Naming a primary key

Some DBMSs allow you to name a primary key and specify whether it is clustered or not clustered. For these DBMSs, the Primary Key property page has a way to specify these properties.

For DBMS-specific information, see your DBMS documentation.

- 4 Right-click on the Object Details view and select Save Changes from the pop-up menu.

Any changes you made in the view are immediately saved to the table definition.

Completing the primary key

Some DBMSs automatically create a unique index when you define a primary key so that you can immediately begin to add data to the table. Others require you to create a unique index separately to support the primary key before populating the table with data.

To find out what your DBMS does, see your DBMS documentation.

Defining foreign keys If your DBMS supports foreign keys, you can define them in DataWindow Designer.

❖ **To create a foreign key:**

1 Do one of the following:

- Highlight the table and click the Create Foreign Key drop-down toolbar button in PainterBar1.
- Select Object>Insert>Foreign Key from the main menu or New>Foreign Key from the pop-up menu.
- Expand the table's tree view and right-click on Foreign Keys and select New Foreign Key from the pop-up menu.

The Foreign Key properties display in the Object Details view. Some of the information is DBMS-specific.

2 Name the foreign key in the Foreign Key Name box.

3 Select the columns for the foreign key.

4 On the Primary Key tab page, select the table and column containing the Primary key referenced by the foreign key you are defining.

Key definitions must match exactly

The definition of the foreign key columns must match the primary key columns, including datatype, precision (width), and scale (decimal specification).

5 On the Rules tab page, specify any information required by your DBMS.

For example, you might need to specify a delete rule by selecting one of the rules listed for On Delete of Primary Table Row.

For DBMS-specific information, see your DBMS documentation.

6 Right-click on the Object Details view and select Save Changes from the pop-up menu.

Any changes you make in the view are immediately saved to the table definition.

Modifying keys

You can modify a primary key in DataWindow Designer.

❖ To modify a primary key:

- 1 Do one of the following:
 - Highlight the primary key listed in the table's expanded tree view and click the Properties button.
 - Select Properties from the Object or pop-up menu.
 - Drag the primary key icon and drop it in the Object Details view.
- 2 Select one or more columns for the primary key.
- 3 Right-click on the Object Details view and select Save Changes from the pop-up menu.

Any changes you make in the view are immediately saved to the table definition.

Dropping a key

You can drop keys (remove them from the database) from within DataWindow Designer.

❖ To drop a key:

- 1 Highlight the key in the expanded tree view for the table in the Objects view or right-click the key icon for the table in the Object Layout view.
- 2 Select Drop Primary Key or Drop Foreign Key from the key's pop-up menu.
- 3 Click Yes.

Working with indexes

You can create as many single- or multi-valued indexes for a database table as you need, and you can drop indexes that are no longer needed.

Update limitation

You can update a table in a DataWindow object only if it has a unique index or primary key.

Creating an index

In Adaptive Server Anywhere databases

In Adaptive Server Anywhere databases, you should not define an index on a column that is defined as a foreign key, because foreign keys are already optimized for quick reference.

❖ **To create an index:**

1 Do one of the following:

- Highlight the table for which you want to create an index and click the Create Index drop-down toolbar button in PainterBar1.
- Select Object>Insert>Index from the main menu or New>Index from the pop-up menu.
- Expand the table's tree view, right-click on Indexes, and select New Index from the pop-up menu.

The Index's properties display in the Object Details view.

2 Enter a name for the index in the Index box.

3 Select whether or not to allow duplicate values for the index.

4 Specify any other information required for your database

For example, in Adaptive Server Enterprise, specify whether the index is clustered, and in Adaptive Server Anywhere, specify the order of the index.

5 Click the names of the columns that make up the index.

6 Select Save Changes from the pop-up menu.

7 Right-click on the Object Details view and select Save Changes from the pop-up menu.

Any changes you made in the view are immediately saved to the table definition.

Modifying an index

You can modify an index.

❖ **To modify an index:**

1 Do one of the following:

- Highlight the index listed in the table's expanded tree view and click the Properties button.
- Select Properties from the Object or pop-up menu.
- Drag the index icon and drop it in the Object Details view.

- 2 In the Object Details view, select or deselect columns as needed.
- 3 Right-click on the Object Details view and select Save Changes from the pop-up menu.
Any changes you made in the view are immediately saved to the table definition.

Dropping an index

Dropping an index removes it from the database.

❖ **To drop an index from a table:**

- 1 In the Database painter workspace, display the pop-up menu for the index you want to drop.
- 2 Select Drop Index and click Yes.

Working with database views

A database view gives a different (and usually limited) perspective of the data in one or more tables. Although you see existing database views listed in the Objects view, a database view does not physically exist in the database as a table does. Each time you select a database view and use the view's data, DataWindow Designer executes a SQL SELECT statement to retrieve the data and creates the database view.

For more information about using database views, see your DBMS documentation.

Using database views in DataWindow Designer

You can define and manipulate database views in DataWindow Designer. Typically you use database views for the following reasons:

- To give names to frequently executed SELECT statements.
- To limit access to data in a table. For example, you can create a database view of all the columns in the Employee table except Salary. Users of the database view can see and update all information except the employee's salary.
- To combine information from multiple tables for easy access.

In DataWindow Designer, you can create single- or multiple-table database views. You can also use a database view when you define data to create a new database view.

You define, open, and manipulate database views in the View painter, which is similar to the Select painter. For more information about the Select painter, see “Selecting a data source” on page 110.

Updating database views

Some database views are logically updatable and others are not. Some DBMSs do not allow any updating of views. For the rules your DBMS follows, see your DBMS documentation.

❖ To open a database view:

- 1 In the Objects view, expand the list of Views for your database.
- 2 Highlight the view you want to open and select Add To Layout from the pop-up menu, or drag the view’s icon to the Object Layout view.

❖ To create a database view:

- 1 Click the Create View button, or select View or New View from the Object>Insert or pop-up menu.

The Select Tables dialog box displays, listing all tables and views that you can access in the database.

- 2 Select the tables and views from which you will create the view by doing one of the following:
 - Click the name of each table or view you want to open in the list displayed in the Select Tables dialog box, then click the Open button to open them. The Select Tables dialog box closes.
 - Double-click the name of each table or view you want to open. Each object is opened immediately. Then click the Cancel button to close the Select Tables dialog box.

Representations of the selected tables and views display in the View painter workspace:

- 3 Select the columns to include in the view and include computed columns as needed.
- 4 Join the tables if there is more than one table in the view.
For information, see “Joining tables” on page 84.
- 5 Specify criteria to limit rows retrieved (Where tab), group retrieved rows (Group tab), and limit the retrieved groups (Having tab), if appropriate.

For information, see the section on using the SQL Select painter in “Selecting a data source” on page 110. The View painter and the SQL Select painter are similar.

6 When you have completed the view, click the Return button.

7 Name the view.

Include *view* or some other identifier in the view’s name so that you will be able to distinguish it from a table in the Select Tables dialog box.

8 Click the Create button.

DataWindow Designer generates a CREATE VIEW statement and submits it to the DBMS. The view definition is created in the database. You return to the Database painter workspace with the new view displayed in the workspace.

Displaying a database view’s SQL statement

You can display the SQL statement that defines a database view. How you do it depends on whether you are creating a new view in the View painter or want to look at the definition of an existing view.

❖ **To display the SQL statement from the View painter:**

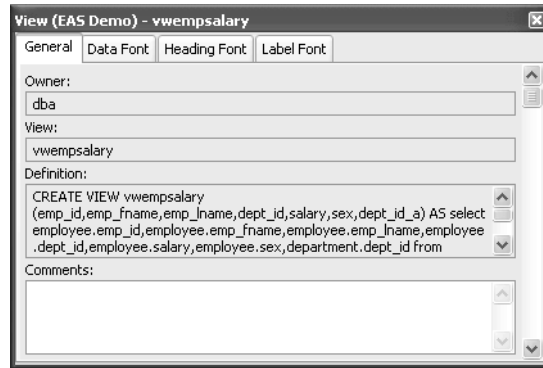
- Select the Syntax tab in the View painter.

DataWindow Designer displays the SQL it is generating. The display is updated each time you change the view.

❖ **To display the SQL statement from the Database painter:**

- Highlight the name of the database view in the Objects view and select Properties from the pop-up menu, or drag the view’s icon to the Object Details view.

The completed SELECT statement used to create the database view displays in the Definition field on the General page:



View dialog box is read-only

You cannot alter the view definition in the Object Details view. To alter a view, drop it and create another view.

Joining tables

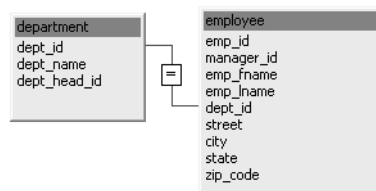
If the database view contains more than one table, you should join the tables on their common columns. When the View painter is first opened for a database view containing more than one table, DataWindow Designer makes its best guess as to the join columns, as follows:

- If there is a primary/foreign key relationship between the tables, DataWindow Designer automatically joins them.
- If there are no keys, DataWindow Designer tries to join tables based on common column names and types.

❖ **To join tables:**

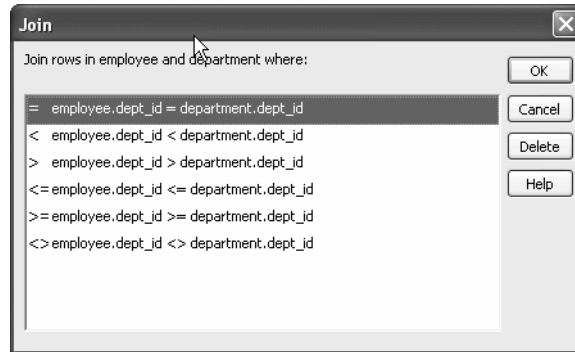
- 1 Click the Join button.
- 2 Click the columns on which you want to join the tables.

In the following screen, the Employee and Department tables are joined on the dept_id column:



- 3 To create a join other than the equality join, click the join representation in the workspace.

The Join dialog box displays:



- 4 Select the join operator you want from the Join dialog box.

If your DBMS supports outer joins, outer join options also display in the Join dialog box. For example, in the preceding dialog box (which uses the Employee and Department tables), you can choose to include rows from the Employee table where there are no matching departments, or rows from the Department table where there are no matching employees.

For more about outer joins, see “Using ANSI outer joins” on page 127.

Dropping a database view

Dropping a database view removes its definition from the database.

❖ To drop a view:

- 1 In the Objects view, select the database view you want to drop.
- 2 Click the Drop Object button or select Drop View from the pop-up menu.

DataWindow Designer prompts you to confirm the drop, then generates a DROP VIEW statement and submits it to the DBMS.

Manipulating data

As you work on the database, you often want to look at existing data or create some data for testing purposes. You might also want to test display formats, validation rules, and edit styles on real data.

DataWindow Designer provides data manipulation for such purposes. With data manipulation, you can:

- Retrieve and manipulate database information
- Save the contents of the database in a variety of formats (such as Excel, PDF, or XML)

Retrieving data

❖ To retrieve data:

- 1 In the Database painter, select the table or database view whose data you want to manipulate.
- 2 Do one of the following:
 - Click one of the three Data Manipulation buttons (Grid, Tabular, or Freeform) in the PainterBar.
 - Select Data or Edit Data from the Object or pop-up menu and choose one of the edit options from the cascading menu that displays.

All rows are retrieved and display in the Results view. As the rows are being retrieved, the Retrieve button changes to a Cancel button. You can click the Cancel button to stop the retrieval.

Exactly what you see in the Results view depends on the formatting style you picked. What you see is actually a DataWindow object. The formatting style you picked corresponds to a type of DataWindow object (grid, tabular, or freeform). In a grid display, you can drag the mouse on a column's border to resize the column.

This window is in the grid format:

Id	Last Name	First Name	Job Role	Street
1	Hildebrand	Jane	Marketing	1280 Washington St.
2	Simmon	Larry	Sales	34 Granville St.
3	Critch	Susan	Product development	45 Center St.
4	Lambert	Terry	Administration	204 Page St.
5	Sullivan	Dorothy	Customer support	54 Minuteman Dr.
6	Paull	Rose	Finance	78 Bay St.
7	Glassmann	Beth	Product development	44 Oak St.
8	Powell	Gene	Training	552 West Main St.

Only a few rows of data display at a time. You can use the First, Prior, Next, and Last buttons to move from page to page.

Modifying data

You can add, modify, or delete rows. When you have finished manipulating the data, you can apply the changes to the database.

If looking at data from a view

Some views are logically updatable and others are not. Some DBMSs do not allow any updating of views.

For the rules your DBMS follows regarding updating of views, see your DBMS documentation.

❖ To modify data:

- 1 Do one of the following:
 - To modify existing data, tab to a field and enter a new value.
 - To add a row, click the Insert Row button and enter data in the new row.
 - To delete a row, click the delete Row button.

When you add or modify data, the data uses the validation rules, display formats, and edit styles that you or others have defined for the table in the Database painter.

- 2 Click the Save Changes button or select Rows>Update to apply changes to the database.

Sorting rows

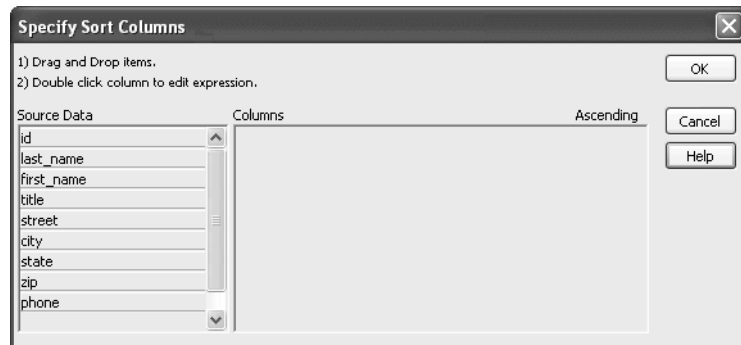
You can sort the data, but any sort criteria you define are for testing only and are not saved with the table or passed to the DataWindow painter.

❖ **To sort the rows:**

- 1 Select Rows>Sort from the menu bar.

The Specify Sort Columns dialog box displays.

- 2 Drag the columns you want to sort on from the Source Data box to the Columns box:



A check box with a check mark in it displays under the Ascending heading to indicate that the values will be sorted in ascending order. To sort in descending order, clear the check box.

Precedence of sorting

The order in which the columns display in the Columns box determines the precedence of the sorting. For example, in the preceding dialog box, rows would be sorted by department ID. Within department ID, rows would be sorted by state.

To change the precedence order, drag the column names in the Column box into the order you want.

- 3 (Optional) Double-click an item in the Columns box to specify an expression to sort on.

The Modify Expression dialog box displays.

- 4 Specify the expression.

For example, if you have two columns, Revenues and Expenses, you can sort on the expression `Revenues - Expenses`.

- Click OK to return to the Specify Sort Columns dialog box with the expression displayed.

If you change your mind

You can remove a column or expression from the sorting specification by simply dragging it and releasing it outside the Columns box.

- When you have specified all the sort columns and expressions, click OK.

Filtering rows

You can limit which rows are displayed by defining a filter.

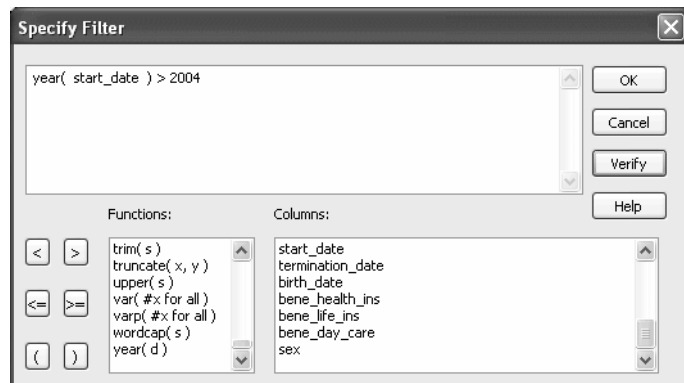
The filters you define are for testing only and are not saved with the table or passed to the DataWindow painter.

❖ To filter the rows:

- Select Rows>Filter from the menu bar.

The Specify Filter dialog box displays.

- Enter a boolean expression that DataWindow Designer will test against each row:



If the expression evaluates to TRUE, the row is displayed. You can paste functions, columns, and operators in the expression.

- Click OK.

DataWindow Designer filters the data. Only rows meeting the filter criteria are displayed.

❖ **To remove the filter:**

- 1 Select Rows>Filter from the menu bar.

The Specify Filter dialog box displays, showing the current filter.

- 2 Delete the filter expression, then click OK.

Filtered rows and updates

Filtered rows are updated when you update the database.

Viewing row information

You can display information about the data you have retrieved.

❖ **To display row information:**

- Select Rows>Described from the menu bar.

The Describe Rows dialog box displays showing the number of:

- Rows that have been deleted in the painter but not yet deleted from the database
- Rows displayed in Preview
- Rows that have been filtered
- Rows that have been modified in the painter but not yet modified in the database

All row counts are zero until you retrieve the data from the database or add a new row. The count changes when you modify the displayed data or test filter criteria.

Importing data

You can import data from an external source and then save the imported data in the database.

❖ **To import data:**

- 1 Select Rows>Import from the menu bar.

The Select Import File dialog box displays.

- 2 Specify the file from which you want to import the data.
The types of files you can import into the painter are shown in the Files of Type drop-down list.
- 3 Click Open.
DataWindow Designer reads the data from the file. You can click the Save Changes button or select Rows>Update to add the new rows to the database.

Printing data

You can print the data displayed by selecting File>Print from the menu bar. Before printing, you can also preview the output on the screen.

❖ **To preview printed output before printing:**

- 1 Select File>Print Preview from the menu bar.
Preview displays the data as it will print. To display rulers around the page borders in Print Preview, select File>Print Preview Rulers.
- 2 To change the magnification used in Print Preview, select File>Print Preview Zoom from the menu bar.
The Zoom dialog box displays.
- 3 Select the magnification you want and click OK.
Preview zooms in or out as appropriate.
- 4 When you have finished looking at the print layout, select File>Print Preview from the menu bar again.

Saving data

You can save the displayed data in an external file.

❖ **To save the data in an external file:**

- 1 Select File>Save Rows As from the menu bar.
The Save Rows As dialog box displays.

- 2 Choose a format for the file.

You can select from several formats, including Powersoft report (PSR), XML, and HTML.

If you want the column headers saved in the file, select a file format that includes headers, such as Excel With Headers. When you select a *with headers* format, the names of the database columns (not the column labels) will also be saved in the file.

For more information, see “Saving data in an external file” on page 168.

- 3 For TEXT, CSV, SQL, HTML, and DIF formats, select an encoding for the file.

You can select ANSI/DBCS, Unicode LE (Little-Endian), Unicode BE (Big-Endian), or UTF8.

- 4 Name the file and save it.

DataWindow Designer saves all displayed rows in the file; all columns in the displayed rows are saved. Filtered rows are not saved.

Creating and executing SQL statements

The Database painter's Interactive SQL view is a SQL editor in which you can enter and execute SQL statements. The view provides all editing capabilities needed for writing and modifying SQL statements. You can cut, copy, and paste text; search for and replace text; and paint SQL statements. You can also set editing properties to make reading your SQL files easier.

Building and executing SQL statements

You can use the Interactive SQL view to build SQL statements and execute them immediately. The view acts as a notepad in which you can enter SQL statements.

Creating stored procedures

You can use the Interactive SQL view to create stored procedures or triggers, but make sure that the painter's SQL statement terminator character is not the same as the terminator character used in the stored procedure language of your DBMS.

About the statement terminator

By default, DataWindow Designer uses the semicolon as the SQL statement terminator. You can override the semicolon by specifying a different terminator character in the Database painter. To change the terminator character, select Design>Options from the Database painter's menu bar.

Make sure that the character you choose is not reserved for another use by your database vendor. For example, using the slash character (/) causes compilation errors with some DBMSs.

Controlling comments

By default, DataWindow Designer strips off comments when it sends SQL to the DBMS. You can have comments included by clearing the check mark next to Strip Comments in the pop-up menu of the Interactive SQL view.

Entering SQL

You can enter a SQL statement in four ways:

- Pasting the statement
- Typing the statement in the view
- Opening a text file containing the SQL
- Dragging a procedure or function from the Objects view

Pasting SQL

You can paste SELECT, INSERT, UPDATE, and DELETE statements to the view. Depending on which kind of statement you want to paste, DataWindow Designer displays dialog boxes that guide you through painting the full statement.

❖ To paste a SQL statement to the workspace:

- 1 Click the Paste SQL button in the PainterBar, or select Paste Special>SQL from the Edit or pop-up menu.

The SQL Statement Type dialog box displays, listing the types of SQL statements you can use.

- 2 Double-click the appropriate icon to select the statement type.

The Select Table dialog box displays.

- 3 Select the table(s) you will reference in the SQL statement.

You go to the Select, Insert, Update, or Delete painter, depending on the type of SQL statement you are pasting. The Insert, Update, and Delete painters are similar to the Select painter, but only the appropriate tabs display in the SQL toolbox at the bottom of the workspace.

For more information about the Select painter, see “Selecting a data source” on page 110.

- 4 Do one of the following:

- For a SELECT statement, define the statement exactly as in the Select painter when building a view.

You choose the columns to select. You can define computed columns, specify sorting and joining criteria, and WHERE, GROUP BY, and HAVING criteria. For more information, see “Working with database views” on page 81.

- For an INSERT statement, type the values to insert into each column. You can insert as many rows as you want.
- For an UPDATE statement, specify the new values for the columns in the Update Column Values dialog box. Then specify the WHERE criteria to indicate which rows to update.
- For a DELETE statement, specify the WHERE criteria to indicate which rows to delete.

- 5 When you have finished painting the SQL statement, click the Return button in the PainterBar in the Select, Insert, Update, or Delete painter.

You return to the Database painter with the SQL statement pasted into the ISQL view.

Typing SQL

Rather than paste, you can simply type one or more SQL statements directly in the ISQL view.

You can enter most statements supported by your DBMS, including statements you can paint as well as statements you cannot paint, such as a database stored procedure or CREATE TRIGGER statement. You cannot enter certain statements that could destabilize the DataWindow Designer development environment. These include the SET statement and the USE *database* statement.

Sybase Adaptive Server Enterprise stored procedures

When you use the Database painter to execute a Sybase Adaptive Server Enterprise system stored procedure, you *must* start the syntax with the keyword EXEC or EXECUTE. For example, enter EXEC SP_LOCK. You cannot execute the stored procedure simply by entering its name.

Importing SQL from a text file

You can import SQL that has been saved in a text file into the Database painter.

❖ **To read SQL from a file:**

- 1 Put the insertion point where you want to insert the SQL.
- 2 Select Paste Special>From File from the Edit or pop-up menu.
- 3 Select the file containing the SQL, and click OK.

Dragging a procedure or function from the Objects view

From the tree view in the Objects view, you can select an existing procedure or function that contains a SQL statement you want to enter, and drag it to the Interactive SQL view.

Executing SQL

When you have the SQL statements you want in the workspace, you can submit them to the DBMS.

❖ **To execute the SQL:**

- Click the Execute button, or select Design>Execute SQL from the menu bar.

If the SQL retrieves data, the data appears in grid format in the Results view. If there is a database error, you see a message box describing the problem.

For a description of what you can do with the data, see “Manipulating data” on page 86.

Customizing the editor

The Interactive SQL view provides the same editing capabilities as the file editor. It also has Script, Font, and Coloring properties that you can change to make SQL files easier to read. With no change in properties, SQL files have black text on a white background and a tab stop setting of 3 for indentation.

Setting Script and
Font properties

Select Design>Options from the menu bar to open the Database Preferences dialog box. The Script and Font properties are the same as those you can set for the file editor.

For more information, see “Using the file editor” on page 17.

Editor properties apply elsewhere

When you set Script and Font properties for the Database painter, the settings also apply to the file editor.

Setting Coloring
properties

You can set the text color and background color for SQL styles (such as datatypes and keywords) so that the style will stand out and the SQL code will be more readable. You set Coloring properties on the Coloring tab page.

Enabling syntax coloring

Be sure the Enable Syntax Coloring check box is selected before you set colors for SQL styles. You can turn off all Coloring properties by clearing the check box.

Controlling access to the current database

The Database painter's Design menu provides access to a series of dialog boxes you can use to control access to the current database. In some DBMSs, for example, you can assign table access privileges to users and groups.

Which menu items display on the Design menu and which dialog boxes display depend on your DBMS.

For information about support for security options in your DBMS, see *Connecting to Your Database* and your DBMS documentation.

DataWindows

This part introduces you to the many styles of DataWindows available in DataWindow Designer and describes how to create and work with them.

Defining DataWindow Objects

About this chapter

This chapter describes how to define DataWindow objects to display and manipulate data.

Contents

Topic	Page
About DataWindow objects	99
Choosing a presentation style	101
Building a DataWindow object	109
Selecting a data source	110
Using Quick Select	112
Using SQL Select	121
Using Query	136
Using External	136
Using Stored Procedure	137
Using ADO DataSet	139
Choosing DataWindow object-wide options	141
Generating and saving a DataWindow object	142
Defining queries	145
What's next	147

About DataWindow objects

A DataWindow object is an object you use to retrieve, present, and manipulate data from a relational database or other data source (such as an Excel worksheet or dBASE file).

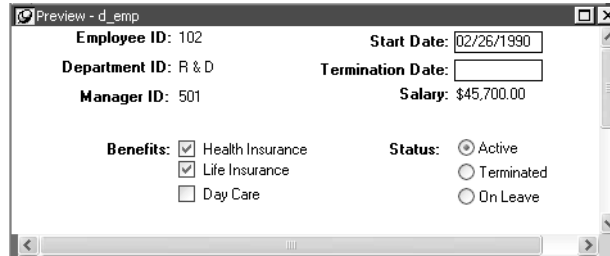
DataWindow objects have knowledge about the data they are retrieving. You can specify display formats, presentation styles, and other data properties so that users can make the most meaningful use of the data.

DataWindow object examples

You can display the data in the format that best presents the data to your users.

Edit styles

If a column can take only a small number of values, you can have the data appear as radio buttons in a DataWindow object so that users know what their choices are.



Display formats

If a column displays phone numbers, salaries, or dates, you can specify the format appropriate to the data.

The screenshot shows a DataWindow object titled 'Preview - d_formats' displaying a table of employee data. The columns are formatted as follows:

Employee ID	Last Name	Phone	Salary	Start Date
102	Whitney	(617) 555-3985	\$45,700.00	02/26/2000
105	Cobb	(617) 555-3840	\$62,000.00	07/02/1999
129	Chin	(404) 555-2341	\$78,500.00	08/04/2001
148	Jordan	(617) 555-7835	\$51,432.00	10/04/2004
160	Breault	(617) 555-3099	\$87,490.00	12/16/1998

Validation rules

If a column can take numbers only in a specific range, you can specify a simple validation rule for the data, without writing any code, to make sure users enter valid data.

Enhancing DataWindow objects

If you want to enhance the presentation and manipulation of data in a DataWindow object, you can include computed fields, pictures, and graphs that are tied directly to the data retrieved by the object.

Reports versus DataWindow objects

Reports and DataWindow objects are the same objects. You can open and modify both in the DataWindow painter. However, a report is not updatable and can only be used to present data. For information about how you can specify whether users can update the data in a DataWindow object, see Chapter 6, “Controlling Updates in DataWindow Objects.”

Choosing a presentation style

The presentation style you select for a DataWindow object determines the format DataWindow Designer uses to display the DataWindow object in the Design view. You can use the format as displayed or modify it to meet your needs.

When you create a DataWindow object, you can choose from the presentation styles listed in the following table.

Table 4-1: DataWindow Presentation styles

Using this DataWindow wizard	You create a new DataWindow object
Composite	That includes other DataWindow objects
Crosstab	With summary data in a spreadsheet-like grid
Freeform	With the data columns going down the page and labels next to each column
Graph	With data displayed in a graph
Grid	With data in row and column format with grid lines separating rows and columns
Group	With data in rows that are divided into groups
Label	That presents data as labels
N-Up	With two or more rows of data next to each other
Tabular	With data columns going across the page and headers above each column
TreeView	With hierarchical data in rows that are divided into levels in a TreeView

Using the Tabular style

The Tabular presentation style presents data with the data columns going across the page and headers above each column. As many rows from the database will display at one time as can fit in the DataWindow object. You can reorganize the default layout any way you want by moving columns and text:

The screenshot shows a window titled 'Preview - emp_total_compensation'. The main title is 'Total Compensation Report Salary Plus Benefits'. In the top right corner, it says 'Page 1 of 4' and '6/8/2004'. A summary box contains the following text: 'Value of health ins. = \$4,800', 'Value of life insurance = \$(5.43 x salary)/1,000', and 'Value of day care = \$5,200'. Below this is a table with the following columns: Department ID, Employee ID, Employee First Name, Employee Last Name, Salary, Health Ins., Life Ins., Day Care, and Salary Plus Benefits. The data rows are as follows:

Department ID	Employee ID	Employee First Name	Employee Last Name	Salary	Health Ins.	Life Ins.	Day Care	Salary Plus Benefits
100	102	Fran	Whitney	\$45,700	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$50,748
	105	Matthew	Cobb	\$62,000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$67,137
	160	Robert	Breault	\$57,490	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$67,802
	243	Natasha	Shishov	\$72,995	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$78,191
	247	Kiwi	Dwivedi	\$48,874	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$58,384

Using the Freeform style

The Freeform presentation style presents data with the data columns going down the page and labels next to each column. You can reorganize the default layout any way you want by moving columns and text. The Freeform style is often used for data entry forms.

The screenshot shows a window titled 'Preview - emp_info_freeform_watermark'. The main title is 'Employee Information' and the date is '6/8/2004', with 'Page 1 of 25' in the bottom right. The form contains the following fields:

- Employee ID: 102
- Manager ID: 501
- Emp. First Name: Fran
- Emp. Last Name: Whitney
- Department ID: 100
- Street: 49 East Washington Street
- City: Needham
- State: MA
- Zip Code: 02192-
- Phone: (617) 555-3985
- Status: Active, Terminated, On Leave
- Soc. Sec. No.: 017-24-8033
- Salary: \$45,700
- Start Date: 02/26/1990
- Termination Date: 00/00/0000
- Birth Date: 06/05/1962
- Health Insurance:
- Life Insurance:

Using the Grid style

The Grid presentation style shows data in row-and-column format with grid lines separating rows and columns. With other styles, you can move text, values, and other objects around freely in designing the report. With the grid style, the grid lines create a rigid structure of cells.

An advantage of the Grid style is that users can reorder and resize columns at runtime.

Original Grid report

This grid report shows employee information. Several of the columns have a large amount of extra white space:

Employee ID	First Name	Last Name	Street	City	State
102	Fran	Whitney	49 East Washington Street	Needham	MA
105	Matthew	Cobb	77 Pleasant Street	Waltham	MA
129	Philip	Chin	59 Pond Street	Atlanta	GA
148	Julie	Jordan	144 Great Plain Avenue	Winchester	MA
160	Robert	Breault	58 Cherry Street	Milton	MA
184	Melissa	Espinoza	112 Apple Tree Way	Stow	MA
191	Jeanette	Bertrand	209 Concord Street	Acton	MA
195	Marc	Dill	89 Hancock Street	Milton	MA
207	Jane	Francis	112 Hawthorne Drive	Concord	MA

Grid report with modified column widths













This grid report was created from the original one by decreasing the width of some columns:

Employee ID	First Name	Last Name	Street	City	State	Phone
102	Fran	Whitney	49 East Washington Street	Needham	MA	(617) 555-3985
105	Matthew	Cobb	77 Pleasant Street	Waltham	MA	(617) 555-3840
129	Philip	Chin	59 Pond Street	Atlanta	GA	(404) 555-2341
148	Julie	Jordan	144 Great Plain Avenue	Winchester	MA	(617) 555-7835
160	Robert	Breault	58 Cherry Street	Milton	MA	(617) 555-3099
184	Melissa	Espinoza	112 Apple Tree Way	Stow	MA	(508) 555-2319
191	Jeanette	Bertrand	209 Concord Street	Acton	MA	(508) 555-8138
195	Marc	Dill	89 Hancock Street	Milton	MA	(617) 555-2144
207	Jane	Francis	112 Hawthorne Drive	Concord	MA	(603) 555-9022

Using the Label style

The Label presentation style shows data as labels. With this style you can create mailing labels, business cards, name tags, index cards, diskette labels, file folder labels, and many other types of labels.

Mailing labels

 Rodrigo Guevara East Main Street Framingham, MA 01701	 Jeannette Bertrand 209 Concord Street Acton, MA 01720	 James Coleman 57 Heather Hill Drive Acton, MA 01720
 Joseph Barker 58 West Drive Bedford, MA 01730	 Irene Barletta 37 Gleason Street Bedford, MA 01730	 Robert Nielsen 55 Sargent Avenue Bedford, MA 01730
 Catherine Pickett 45 Appleton Road Bedford, MA 01730	 Sheila Romero 1 Oakview Terrace Bedford, MA 01730	 Doug Charlton 57 Webster Street Concord, MA 01742
 Scott Evans 10-A Sunrise Circle Concord, MA 01742	 Jane Francis 12 Hawthorne Drive Concord, MA 01742	 Jennifer Litton 17 Downing Street Concord, MA 01742

Business cards

<i>My company</i> Terry Lambert Administration 204 Peace St. Canton, MA 94608 Phone: (617) 555-2246 Fax: (617) 555-3692	<i>My company</i> Terry Lambert Administration 204 Peace St. Canton, MA 94608 Phone: (617) 555-2246 Fax: (617) 555-3692
<i>My company</i> Terry Lambert Administration 204 Peace St. Canton, MA 94608 Phone: (617) 555-2246 Fax: (617) 555-3692	<i>My company</i> Terry Lambert Administration 204 Peace St. Canton, MA 94608 Phone: (617) 555-2246 Fax: (617) 555-3692

Name tags

Lynn Page Sales	Charles Crowley Human resources
Dana Burnill Product development	William Caruso Finance

Specifying label properties

If you choose the Label style, you are asked to specify the properties for the label after specifying the data source. You can choose from a list of predefined label types or enter your own specifications manually.

Where label definitions come from

DataWindow Designer gets the information about the predefined label formats from a preferences file called *PBLAB105.INI*.

Using the N-Up style

The N-Up style presents two or more rows of data next to each other. It is similar to the Label style in that you can have information from several rows in the database across the page. However, the information is not meant to be printed on labels. The N-Up presentation style is useful if you have periodic data; you can set it up so that each period repeats in a row.

After you select a data source, you are asked how many rows to display across the page.

For each column in the data source, DataWindow Designer defines n columns in the DataWindow object (column_1 to column_n), where n is the number of rows you specified.

Table example

For a table of daily stock prices, you can define the DataWindow object as five across, so each row in the DataWindow object displays five days' prices (Monday through Friday). Suppose you have a table with two columns, day and price, that record the closing stock price each day for three weeks.

In the following n-up DataWindow object, 5 was selected as the number of rows to display across the page, so each line in the DataWindow object shows five days' stock prices. A computed field was added to get the average closing price in the week:

Monday	Tuesday	Wednesday	Thursday	Friday					
Header ↑									
day_1	price_1	day_2	price_2	day_3	price_3	day_4	price_4	day_5	price_5
"Average price for week: \$" + (price[0] + price[1] + price[2] + price[3] + price[4] + price[5] + price[6] + price[7] + price[8] + price[9]) / 5									
Detail ↑									
Summary ↑									
Footer ↑									

About computed fields in n-up DataWindow objects

You use subscripts, such as price[0], to refer to particular rows in the detail band in n-up DataWindow objects.

For more information, see Chapter 5, "Enhancing DataWindow Objects."

Here is the DataWindow object in the Preview view:

Monday	Tuesday	Wednesday	Thursday	Friday
05/04/01 62.00	05/05/01 63.00	05/06/01 66.00	05/07/01 65.00	05/08/01 62.00
Average price for week: \$63.6				
05/11/01 65.00	05/12/01 69.00	05/13/01 66.00	05/14/01 68.00	05/15/01 67.00
Average price for week: \$67				
05/18/01 67.00	05/19/01 71.00	05/20/01 70.00	05/21/01 72.00	05/22/01 75.00
Average price for week: \$71				

Another way to get multiple-column DataWindow objects

In an n-up DataWindow object, the data is displayed across and then down. If you want your data to go down the page and then across in multiple columns, as in a phone list, you should create a standard tabular DataWindow object, then specify newspaper columns.

For more information on newspaper columns, see Chapter 5, “Enhancing DataWindow Objects.”

Using the Group style

The Group presentation style provides an easy way to create grouped DataWindow objects, where the rows are divided into groups, each of which can have statistics calculated for it. Using this style generates a tabular DataWindow object that has grouping properties defined.

This Group style report groups by department and lists employees and salaries. It also includes a subtotal and a grand total for the salary column:

Employee Report
08/14/02

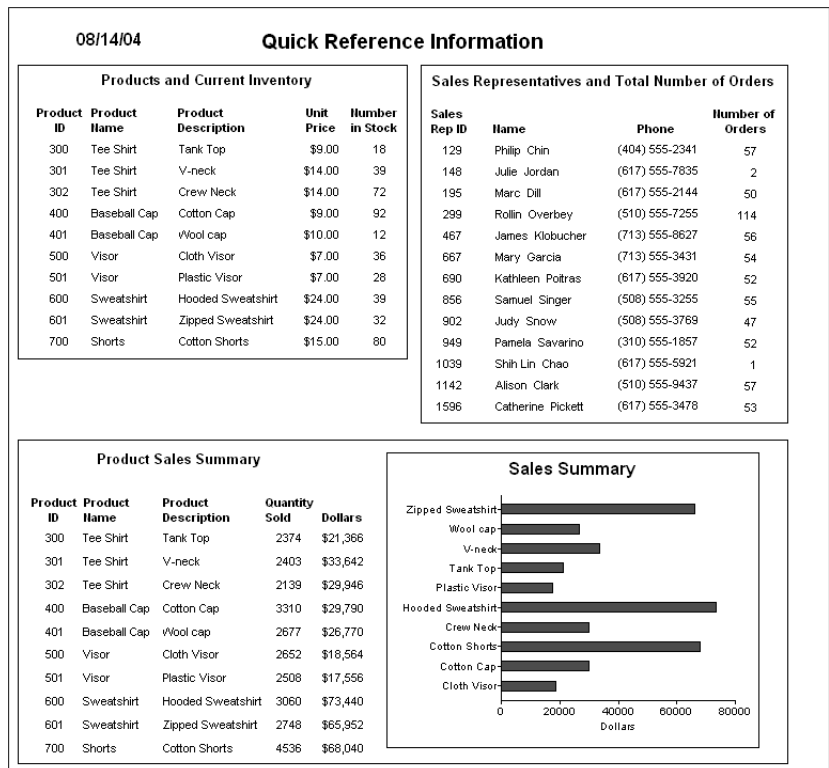
Department ID	Employee ID	First Name	Last Name	Salary
500				
	191	Jeannette	Bertrand	\$92,780
	703	Jose	Martinez	\$91,051
	750	Jane	Braun	\$77,730
	868	Felicia	Kuo	\$88,200
	921	Charles	Crowley	\$61,700
	1013	Joseph	Barker	\$47,290
	1570	Anthony	Rebeiro	\$54,576
	1615	Sheila	Romero	\$77,500
	1658	Michael	Lynch	\$64,903
			Total for department:	\$655,730
			Grand Total:	\$4,101,107

For more about the Group presentation style, see Chapter 9, “Filtering, Sorting, and Grouping Rows.”

Using the Composite style

The Composite presentation style allows you to combine multiple DataWindow objects in the same object. It is particularly handy if you want to print more than one DataWindow object on a page.

This composite report consists of three nested tabular reports. One of the tabular reports includes a graph:



For more about the Composite presentation style, see Chapter 11, “Using Nested Reports.”

Using the Graph and Crosstab styles

In addition to the (preceding) text-based presentation styles, DataWindow Designer provides two styles that allow you to display information graphically: Graph and Crosstab.

There is a graph report in the composite report in “Using the Composite style” on page 107. This crosstab report counts the number of employees that fit into each cell. For example, there are three employees in department 100 who make between \$30,000 and \$39,999:

Number of employees by department and salary 30,000 includes up to 39,999						Total number of employees making the salary	
Salary	Dept Id	100	200	300	400		500
20000					2	4	6
30000	3	8	2	5	3		21
40000	6	5	2	5	1		19
50000	4	3	3	2			12
60000	4	1		2	1		8
70000	2	1	1				4
80000	2	1					3
90000	1						1
130000			1				1
Total number of employees in the department		22	19	9	16	9	

For more information about these two presentation styles, see Chapter 15, “Working with Graphs,” and Chapter 12, “Working with Crosstabs.”

Using the TreeView style

The TreeView presentation style provides an easy way to create DataWindow objects that display hierarchical data in a TreeView, where the rows are divided into groups that can be expanded and collapsed. Icons (+ or –) show whether the state of a group in the TreeView is expanded or collapsed, and lines connect parents and their children.

This TreeView style report groups by manager ID and state and lists employee information and salaries:

Dept. Head ID	State	City	Employee ID	Name	Salary
501	MA				
	TX				
703					
902	CA	Emeryville	299	Rollin Overbey	\$39,300.00
			1142	Alison Clark	\$45,000.00
		Long Beach			
	GA				
	MA	Bedford	1596	Catherine Pickett	\$47,653.00
		Boston			
		Concord			
		Lexington			
		Marblehead			
		Milton			
Needham					
Newton					
Stow					
Wellesley					

For more about the TreeView presentation style, see Chapter 13, “Working with TreeViews.”

Building a DataWindow object

You use a wizard to build a new DataWindow object. To create a DataWindow object or use the DataWindow painter, you must be connected to the database whose data you will be accessing. When you open the DataWindow painter or select a data source in the wizard, DataWindow Designer connects you to the DBMS and database you used last. If you need to connect to a different database, do so before working with a DataWindow object.

Column limit

There is a limit of 1000 on the number of columns in a DataWindow object.

For information about changing your database connection, see *Connecting to Your Database*.

❖ **To create a new DataWindow object:**

- 1 Select File>New from the menu bar and select the DataWindow tab.
- 2 Choose a presentation style for the DataWindow object.
The presentation style determines how the data is displayed. See “Choosing a presentation style” on page 101. When you choose the presentation style, the appropriate DataWindow object wizard starts up.
- 3 If you want data to be retrieved in the Preview view when the DataWindow object opens, select the Retrieve on Preview check box.
- 4 Define the data source.
See “Selecting a data source” on page 110.
- 5 Choose options for the DataWindow object and click Next.
See “Choosing DataWindow object-wide options” on page 141.
- 6 Review your specifications and click Finish.
The DataWindow object displays in the Design view.
- 7 Save the DataWindow object in a library.

Selecting a data source

The data source you choose determines how you select the data that will be used in the DataWindow object.

About the term *data source*

The term *data source* used here refers to how you use the DataWindow painter to specify the data to retrieve into the DataWindow object. Data source can also refer to where the data comes from, such as an Adaptive Server Anywhere data source (meaning a database file) or an XML data source (meaning an XML file).

If the data is in the database

If the data for the DataWindow object will be retrieved from a database, choose one of the following data sources:

Data source	Use when
Quick Select	The data is from a single table (or from tables that are related through foreign keys) and you need only to choose columns, selection criteria, and sorting.
SQL Select	You want more control over the SQL SELECT statement generated for the data source <i>or</i> your data is from tables that are not connected through a key. For example, you need to specify grouping, computed columns, or retrieval arguments within the SQL SELECT statement.
Query	The data has been defined as a query.
Stored Procedure	The data is defined in a stored procedure.
ADO DataSet	You are using an ADO DataSet for data access and you want to use the DataWindow as a presentation tool.

If the data is not in a database

Select the External data source if:

- The DataWindow object will be populated programmatically.
- Data will be imported from an external file, such as an XML, comma-separated values (*CSV*), tab-separated text (*TXT*), or dBASE (*DBF*) file.

You can also use an ODBC driver to access data from a file. DataWindow Designer includes ODBC drivers for most file types.

For more information, see *Connecting to Your Database*.

After you choose a data source in the various DataWindow wizards, you specify the data. The data source you choose determines what displays in the wizards and how you define the data.

Why use a DataWindow if the data is not from a DBMS

Even when the data is not coming from the database, there are many times when you want to take advantage of the intelligence of a DataWindow object:

- **Data Validation** You have full access to validation rules for data
- **Display Formats** You can use any existing display formats to present the data, or create your own
- **Edit Styles** You can use any existing edit styles, such as radio buttons and edit masks, to present the data, or create your own

Using Quick Select

The easiest way to define a data source is using Quick Select.

❖ **To define the data using Quick Select:**

- 1 Click Quick Select in the Choose Data Source dialog box in the wizard and click Next.
- 2 Select the table that you will use in the DataWindow object.
For more information, see "Selecting a table" next.
- 3 Select the columns to be retrieved from the database.
For more information, see "Selecting columns" on page 114.
- 4 (Optional) Sort the rows before you retrieve data.
For more information, see "Specifying sorting criteria" on page 115.
- 5 (Optional) Select what data to retrieve.
For more information, see "Specifying selection criteria" on page 115.
- 6 Click the OK button Quick Select dialog box.
You return to the wizard to complete the definition of the DataWindow object.

Quick Select limitations

When you choose Quick Select as your data source, you cannot:

- Specify grouping before rows are retrieved
- Include computed columns
- Specify retrieval arguments for the SELECT statement that are supplied at runtime.

To use these options when you create a DataWindow object, choose SQL Select as your data source. If you decide later that you want to use retrieval arguments, you can define them by modifying the data source. For more information, see Chapter 5, "Enhancing DataWindow Objects."

Selecting a table

When you choose Quick Select, the Quick Select dialog box displays. The Tables box lists tables and views in the current database.

Displaying table comments

To display a comment about a table, position the pointer on the table and click the right mouse button or select the table.

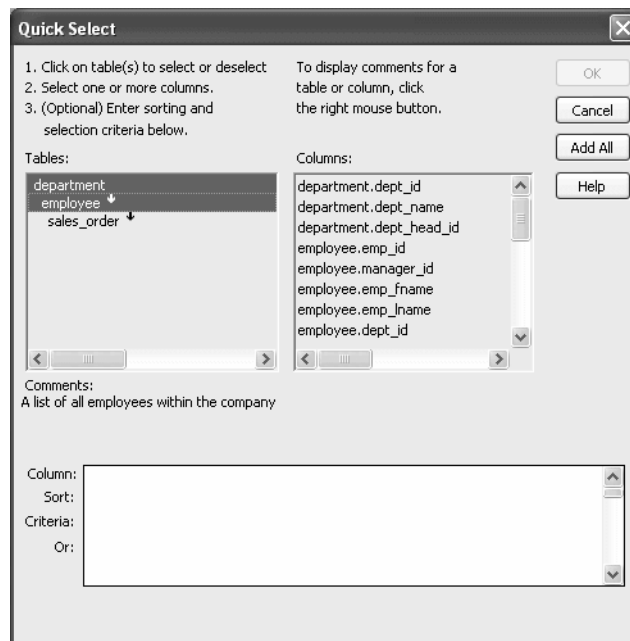
Which tables and views display?

The DBMS determines what tables and views display. For some DBMSs, all tables and views display, whether or not you have authorization. If you select a table or view you are not authorized to access, the DBMS issues a message.

For ODBC databases, the tables and views that display depend on the driver for the data source. Adaptive Server Anywhere does not restrict the display, so all tables and views display, whether or not you have authorization.

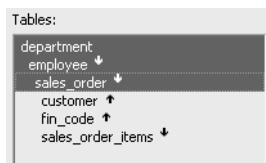
Tables with key relationships

When you select a table, the table's column names display in the Columns box, and any tables having a key relationship with the selected table display in the Tables box. These tables are indented and marked with an arrow to show their relationship to the selected table. You can select any of these related tables if you want to include columns from them in the DataWindow object.



Meaning of the up and down arrows

An arrow displays next to a table to indicate its relationship to the selected table. The arrow always points in the *many* direction of the relationship—toward the selected table (up) if the selected table contains a foreign key in the relationship and away from the selected table (down) if the selected table contains a primary key in the relationship:



In this preceding illustration, the selected table is sales_order. The Up arrows indicate that a foreign key in the sales_order table is mapped to the primary key in the customer and fin_code tables. The Down arrow indicates that the sales_order_items table contains a foreign key mapped to the primary key in the sales_order table.

How columns from additional tables display

The column names of selected tables display in the Columns box. If you select more than one table, the column names are identified as:

tablename.columnname

For example, department.dept_name and employee.emp_id display when the Employee table and the Department table are selected.

To return to the original table list

Click the table you first selected at the top of the table list.

Selecting columns

You can select columns from the primary table and from its related tables. Select the table whose columns you want to use in the Tables box, and add columns from the Columns box:

- To add a column, select it in the Columns box.
- To add all the columns that display in the Columns box, click Add All.
- To remove a column, deselect it in the Columns box.
- To view comments that describe a table or column, position the pointer on a table or column name, and press and hold the right mouse button.

As you select columns, they display in the grid at the bottom of the dialog box in the order in which you select them. If you want the columns to display in a different order in the DataWindow object, select a column name you want to move in the grid and drag it to the new location.

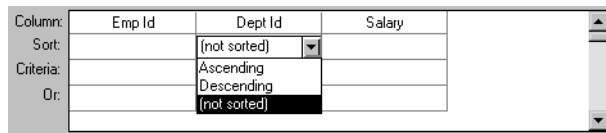
Specifying sorting criteria

In the grid at the bottom of the Quick Select dialog box, you can specify if you want the retrieved rows to be sorted. As you specify sorting criteria, DataWindow Designer builds an ORDER BY clause for the SELECT statement.

❖ To sort retrieved rows on a column:

- 1 Click in the Sort row for the column you want to sort on.

DataWindow Designer displays a drop-down list:



- 2 Select the sorting order for the rows: Ascending or Descending.

Multilevel sorts

You can specify as many columns for sorting as you want. DataWindow Designer processes the sorting criteria left to right in the grid: the first column with Ascending or Descending specified becomes the highest level sorting column, the next column with Ascending or Descending specified becomes the next level sorting column, and so on.

If you want to do a multilevel sort that does not match the column order in the grid, drag the columns to the correct order and then specify the columns for sorting.

Specifying selection criteria

You can enter selection criteria in the grid to specify which rows to retrieve. For example, instead of retrieving data about all employees, you might want to limit the data to employees in Sales and Marketing, or to employees in Sales who make more than \$80,000.

As you specify selection criteria, DataWindow Designer builds a WHERE clause for the SELECT statement.

❖ **To specify selection criteria:**

1 Click the Criteria row below the first column for which you want to select the data to retrieve.

2 Enter an expression, or if the column has an edit style, select or enter a value.

If the column is too narrow for the criterion, drag the grid line to enlarge the column. This enlargement does not affect the column size in a DataWindow object.

3 Enter additional expressions until you have specified the data you want to retrieve.

About edit styles

If a column has an edit style associated with it in the extended attribute system tables (that is, the association was made in the Database painter), if possible, the edit style is used in the grid. Drop-down list boxes are used for columns with code tables and columns using the CheckBox and RadioButton edit styles.

SQL operators supported in Quick Select

You can use these SQL relational operators in the retrieval criteria:

Table 4-2: SQL relational operators used in retrieval criteria

Operator	Meaning
=	Is equal to (default operator)
>	Is greater than
<	Is less than
<>	Is not equal to
>=	Is greater than or equal to
<=	Is less than or equal to
LIKE	Matches this pattern
NOT LIKE	Does not match this pattern
IN	Is in this set of values
NOT IN	Is not in this set of values

Because = is the default operator, you can enter the value 100 instead of = 100, or the value New Hampshire instead of = New Hampshire.

Comparison operators You can use the LIKE, NOT LIKE, IN, and NOT IN operators to compare expressions.

Use LIKE to search for strings that match a predetermined pattern. Use NOT LIKE to find strings that do not match a predetermined pattern. When you use LIKE or NOT LIKE, you can use wildcards:

- The percent sign (%), like the DOS wildcard asterisk (*), matches multiple characters. For example, `Good%` matches all names that begin with `Good`.
- The underscore character (_) matches a single character. For example, `Good _ _ _` matches all seven-letter names that begin with `Good`.

Use IN to compare and include a value that is in a set of values. Use NOT IN to compare and include values that are not in a set of values. For example, the following clause selects all employees in department 100, 200, or 500:

```
SELECT * from employee
WHERE dept_id IN (100, 200, 500)
```

Using NOT IN in this clause would exclude employees in those departments.

Connection operators You can use the OR and AND logical operators to connect expressions.

DataWindow Designer makes some assumptions based on how you specify selection criteria. When you specify:

- Criteria for more than one column on one line
DataWindow Designer assumes a logical AND between the criteria. A row from the database is retrieved if *all* criteria in the line are met.
- Two or more lines of selection criteria
DataWindow Designer assumes a logical OR. A row from the database is retrieved if the criterion in *any* of the lines is met.

To override these defaults, begin an expression with the AND or OR operator:

Operator	Meaning
OR	The row is selected if one expression OR another expression is true
AND	The row is selected if one expression AND another expression are true

This technique is particularly handy when you want to retrieve a range of values in a column. See example 6 below.

SQL expression examples

The first six examples in this section all refer to a grid that contains three columns: EmpId, DeptId, and Salary.

Example 1

The expression `<50000` in the Criteria row in the Salary column in the grid retrieves information for employees whose salaries are less than \$50,000.

Column:	Emp Id	Dept Id	Salary
Sort:			
Criteria:			<50000
Or:			

The **SELECT** statement that DataWindow Designer creates is:

```
SELECT emp_id, dept_id, salary
FROM employee
WHERE salary < 50000
```

Example 2

The expression `100` in the Criteria row in the DeptId column in the grid retrieves information for employees who belong to department 100.

Column:	Emp Id	Dept Id	Salary
Sort:			
Criteria:		100	
Or:			

The **SELECT** statement that DataWindow Designer creates is:

```
SELECT emp_id, dept_id, salary
FROM employee
WHERE dept_id = 100
```

Example 3

The expression `>300` in the Criteria row in the EmpId column and the expression `<50000` in the Criteria row in the Salary column in the grid retrieve information for any employee whose employee ID is greater than 300 *and* whose salary is less than \$50,000.

Column:	Emp Id	Dept Id	Salary
Sort:			
Criteria:	>300		<50000
Or:			

The **SELECT** statement that DataWindow Designer creates is:

```
SELECT emp_id, dept_id, salary
FROM employee
WHERE emp_id >300 AND salary <50000
```

Example 4

The expressions 100 in the Criteria row and >300 in the Or row for the DeptId column, together with the expression <50000 in the Criteria row in the Salary column, retrieve information for employees who belong to:

- Department 100 *and* have a salary less than \$50,000
- or*
- A department whose ID is greater than 300, no matter what their salaries

Column:	Emp Id	Dept Id	Salary
Sort:			
Criteria:		100	<50000
Or:		>300	

The SELECT statement that DataWindow Designer creates is:

```
SELECT emp_id, dept_id, salary
FROM employee
WHERE (dept_id = 100 AND salary < 50000)
OR dept_id > 300
```

Example 5

The expression IN(100, 200, 500) in the Criteria row in the DeptId column in the grid retrieves information for employees who are in department 100 or 200 or 500.

Column:	Emp Id	Dept Id	Salary
Sort:			
Criteria:		IN (100,200,500)	
Or:			

The SELECT statement that DataWindow Designer creates is:

```
SELECT emp_id, dept_id, salary
FROM employee
WHERE dept_id IN (100, 200, 500)
```

Example 6

This example shows the use of the word AND in the Or criteria row. In the Criteria row, >=500 is in the EmpId column and >=30000 is in the Salary column. In the Or row, AND <=1000 is in the EmpId column and AND <=50000 is in the Salary column. These criteria retrieve information for employees who have an employee ID from 500 to 1000 and a salary from \$30,000 to \$50,000.

Column:	Emp Id	Dept Id	Salary
Sort:			
Criteria:	>= 500		>= 30000
Or:	AND <= 1000		AND <= 50000

The SELECT statement that DataWindow Designer creates is:

```
SELECT emp_id, dept_id, salary
FROM employee
WHERE (emp_id >= 500 AND emp_id <= 1000)
AND (salary >= 30000 AND salary <= 50000)
```

Example 7

In a grid with three columns: Emp Last Name, Emp First Name, and Salary, the expressions LIKE C% in the Criteria row and LIKE G% in the Or row in the Emp Last Name column retrieve information for employees who have last names that begin with C or G.

Column:	Emp Last Name	Emp First Name	Salary	
Sort:				
Criteria:	LIKE C%			
Or:	LIKE G%			

The SELECT statement that DataWindow Designer creates is:

```
SELECT emp_last_name, emp_first_name, salary
FROM employee
WHERE emp_last_name LIKE 'C%'
OR emp_last_name LIKE 'G%'
```

Providing SQL functionality to users

You can allow your users to specify selection criteria in a DataWindow object using these techniques at runtime:

- You can automatically pop up a window prompting users to specify criteria each time, just before data is retrieved.

For more information, see Chapter 5, “Enhancing DataWindow Objects.”

- You can place the DataWindow object in query mode using the Modify method.

For more information, see the *DataWindow Programmer’s Guide*.

Using SQL Select

In specifying data for a DataWindow object, you have more options for specifying complex SQL statements when you use SQL Select as the data source. When you choose SQL Select, you go to the Select painter, where you can paint a SELECT statement that includes the following:

- More than one table
- Selection criteria (WHERE clause)
- Sorting criteria (ORDER BY clause)
- Grouping criteria (GROUP BY and HAVING clauses)
- Computed columns
- One or more arguments to be supplied at runtime

Saving your work as a query

While in the Select painter, you can save the current SELECT statement as a query by selecting File>Save Query from the menu bar. Doing so allows you to easily use this data specification again in other reports.

For more information about queries, see “Defining queries” on page 145.

❖ **To define the data using SQL Select:**

- 1 Click SQL Select in the Choose Data Source dialog box in the wizard and click Next.

The Select Tables dialog box displays.

- 2 Select the tables and/or views that you will use in the DataWindow object.

For more information, see “Selecting tables and views” next.

- 3 Select the columns to be retrieved from the database.

For more information, see “Selecting columns” on page 124.

- 4 Join the tables if you have selected more than one.

For more information, see “Joining tables” on page 126.

- 5 Select retrieval arguments if appropriate.

For more information, see “Using retrieval arguments” on page 129.

- 6 Limit the retrieved rows with `WHERE`, `ORDER BY`, `GROUP BY`, and `HAVING` criteria, if appropriate.
For more information, see “Specifying selection, sorting, and grouping criteria” on page 130.
- 7 If you want to eliminate duplicate rows, select `Distinct` from the Design menu. This adds the `DISTINCT` keyword to the `SELECT` statement.
- 8 Click the Return button on the PainterBar.
You return to the wizard to complete the definition of the `DataWindow` object.

Selecting tables and views

After you have chosen `SQL Select`, the `Select Tables` dialog box displays in front of the `Table Layout` view of the `Select` painter. What tables and views display in the dialog box depends on the `DBMS`. For some `DBMS`s, all tables and views display, whether or not you have authorization. Then, if you select a table or view you are not authorized to access, the `DBMS` issues a message.

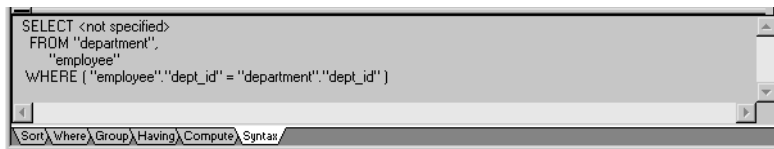
For `ODBC` databases, the tables and views that display depend on the driver for the data source. `Adaptive Server Anywhere` does not restrict the display, so all tables and views display, whether or not you have authorization.

❖ To select the tables and views:

- Do one of the following:
 - Click the name of each table or view you want to open.
Each table you select is highlighted. (To deselect a table, click it again.) Click the `Open` button to close the `Select Tables` dialog box.
 - Double-click the name of each table or view you want to open.
Each object opens immediately behind the `Select Tables` dialog box. Click the `Cancel` button to close the `Select Tables` dialog box.

Representations of the selected tables and views display. You can move or size each table to fit the space as needed.

Below the Table Layout view, several tabbed views also display by default. You use the views (for example, Compute, Having, Group) to specify the SQL Select statement in more detail. You can turn the views on and off from the View menu on the menu bar.



Specifying what is displayed

You can display the label and datatype of each column in the tables (the label information comes from the extended attribute system tables). If you need more space, you can choose to hide this information.

❖ **To hide or display comments, datatypes, and labels:**

- 1 Position the pointer on any unused area of the Table Layout view and select Show from the pop-up menu.

A cascading menu displays.

- 2 Select or clear Datatypes, Labels, or Comments as needed.

Colors in the Select painter

The colors used by the Select painter to display the Table Layout view background and table information are specified in the Database painter. You can also set colors for the text and background components in the table header and detail areas.

For more information about specifying colors in the Database painter, see “Modifying database preferences” on page 58.

Adding and removing tables and views

You can add tables and views to your Table Layout view at any time.

Table 4-3: Adding tables and views in the Select painter

To do this	Do this
Add tables or views	Click the Tables button in the PainterBar and select tables or views to add
Remove a table or view	Display its pop-up menu and select Close
Remove all tables and views	Select Design>Undo All from the menu bar

You can also remove individual tables and views from the Table Layout view, or clear them all at once and begin selecting a new set of tables.

How DataWindow Designer joins tables

If you select more than one table in the Select painter, DataWindow Designer joins columns based on their key relationship.

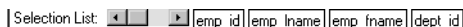
For information about joins, see “Joining tables” on page 126.

Selecting columns

You can click each column you want to include from the table representations in the Table Layout view. DataWindow Designer highlights selected columns and places them in the Selection List at the top of the Select painter.

❖ **To reorder the selected columns:**

- Drag a column in the Selection List with the mouse. Release the mouse button when the column is in the proper position in the list.

Selection List: 

❖ **To select all columns from a table:**

- Move the pointer to the table name and select Select All from the pop-up menu.

❖ **To include computed columns:**

- 1 Click the Compute tab to make the Compute view available (or select View>Compute if the Compute view is not currently displayed).

Each row in the Compute view is a place for entering an expression that defines a computed column.

- 2 Enter one of the following:

- An expression for the computed column. For example: `salary/12`
- A function supported by your DBMS. For example, the following is an Adaptive Server Anywhere function:

```
substr("employee"."emp_fname",1,2)
```

You can display the pop-up menu for any row in the Compute view. Using the pop-up menu, you can select and paste the following into the expression:

- Names of columns in the tables used in the report, form, or pipeline
- Any retrieval arguments you have specified
- Functions supported by the DBMS

About these functions

The functions listed here are provided *by your DBMS*. They are not DataWindow Designer functions. This is so because you are now defining a SELECT statement that will be sent to your DBMS for processing.

- 3 Press the Tab key to get to the next row to define another computed column, or click another tab to make additional specifications.

DataWindow Designer adds the computed columns to the list of columns you have selected.

About computed columns and computed fields

Computed columns you define in the Select painter are added to the SQL statement and used by the DBMS to retrieve the data. The expression you define here follows your DBMS's rules.

You can also choose to define computed fields, which are created and processed dynamically by DataWindow Designer after the data has been retrieved from the DBMS. There are advantages to doing this. For example, work is offloaded from the database server, and the computed fields update dynamically as data changes in the DataWindow object. (If you have many rows, however, this updating can result in slower performance.) For more information, see Chapter 5, "Enhancing DataWindow Objects."

Displaying the underlying SQL statement

As you specify the data for the DataWindow object in the Select painter, DataWindow Designer generates a SQL `SELECT` statement. It is this SQL statement that will be sent to the DBMS when you retrieve data into the DataWindow object. You can look at the SQL as it is being generated while you continue defining the data for the DataWindow object.

❖ To display the SQL statement:

- Click the Syntax tab to make the Syntax view available, or select View>Syntax if the Syntax view is not currently displayed.

You may need to use the scroll bar to see all parts of the SQL `SELECT` statement. This statement is updated each time you make a change.

Editing the `SELECT` statement syntactically

Instead of modifying the data source graphically, you can directly edit the `SELECT` statement in the Select painter.

Converting from syntax to graphics

If the SQL statement contains unions or the `BETWEEN` operator, it may not be possible to convert the syntax back to graphics mode. In general, once you convert the SQL statement to syntax, you should maintain it in syntax mode.

❖ **To edit the SELECT statement:**

- 1 Select Design>Convert to Syntax from the menu bar.
DataWindow Designer displays the SELECT statement in a text window.
- 2 Edit the SELECT statement.
- 3 Do one of the following:
 - Select Design>Convert to Graphics from the menu bar to return to the Select painter.
 - Click the Return button to return to the wizard if you are building a new DataWindow object, or to the DataWindow painter if you are modifying an existing DataWindow object.

Joining tables

If the DataWindow object will contain data from more than one table, you should join the tables on their common columns. If you have selected more than one table, DataWindow Designer joins columns according to whether they have a key relationship:

- Columns with a primary/foreign key relationship are joined automatically.
- Columns with no key relationship are joined, if possible, based on common column names and types.

DataWindow Designer links joined tables in the Select painter Table Layout view. DataWindow Designer joins can differ depending on the order in which you select the tables, and sometimes the DataWindow Designer best-guess join is incorrect, so you may need to delete a join and manually define a join.

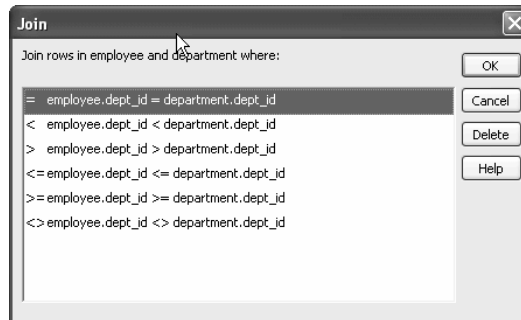
❖ **To delete a join:**

- 1 Click the join operator connecting the tables.
The Join dialog box displays.
- 2 Click Delete.

❖ **To join tables:**

- 1 Click the Join button in the PainterBar.
- 2 Click the columns on which you want to join the tables.
- 3 To create a join other than an equality join, click the join operator in the Table Layout view.

The Join dialog box displays:



- 4 Select the join operator you want and click OK.

If your DBMS supports outer joins, outer join options also display in the Join dialog box.

Using ANSI outer joins

All DataWindow Designer database interfaces provide support for ANSI SQL-92 outer join SQL syntax generation. DataWindow Designer supports both left and right outer joins in graphics mode in the Select painter, and full outer and inner joins in syntax mode. Depending on your database interface, you might need to set the OJSyntax DBParm to enable ANSI outer joins. For more information, see OJSyntax in the online Help.

The syntax for ANSI outer joins is generated according to the following BNF (Backus Naur form):

```
OUTER-join ::=
table-reference {LEFT | RIGHT} OUTER JOIN table-reference ON
search-condition
```

```
table-reference ::=
table_view_name [correlation_name] | OUTER-join
```

Order of evaluation
and nesting

In ANSI SQL-92, when nesting joins, the result of the first outer join (determined by order of ON conditions) is the operand of the outer join that follows it. In DataWindow Designer, an outer join is considered to be nested if the *table-reference* on the left of the JOIN has been used before within the same outer join nested sequence.

The order of evaluation for ANSI syntax nested outer joins is determined by the order of the ON search conditions. This means that you must create the outer joins in the intended evaluation order and add nested outer joins to the end of the existing sequence, so that the second *table-reference* in the outer join BNF above will always be a *table_view_name*.

Nesting example

For example, if you create a left outer join between a column in Table1 and a column in Table2, then join the column in Table2 to a column in Table3, the product of the outer join between Table1 and Table2 is the operand for the outer join with Table3.

For standard database connections, the default generated syntax encloses the outer joins in escape notation {oj . . . } that is parsed by the driver and replaced with DBMS-specific grammar:

```
SELECT Table1.col1, Table2.col1, Table3.col1
FROM {oj {oj Table1 LEFT OUTER JOIN Table2 ON Table1.col1 =
Table2.col1}
LEFT OUTER JOIN Table3 ON Table2.col1 = Table3.col1}
```

Table references

Table references are considered equal when the table names are equal and there is either no alias (correlation name) or the same alias for both. Reusing the operand on the right is not allowed, because ANSI does not allow referencing the *table_view_name* twice in the same statement without an alias.

Determining left and right outer joins

When you create a join condition, the table you select first in the painter is the left operand of the outer join. The table that you select second is the right operand. The condition you select from the Joins dialog box determines whether the join is a left or right outer join.

For example, suppose you select the dept_id column in the employee table, then select the dept_id column in the department table, then choose the following condition:

```
employee.dept_id = department.dept_id and rows from
department that have no employee
```

The syntax generated is:

```
SELECT employee.dept_id, department.dept_id
FROM {oj "employee" LEFT OUTER JOIN "department" ON
"employee"."dept_id" = "department"."dept_id"}
```

If you select the condition with rows from department that have no employee, you create a right outer join instead.

Equivalent statements

The syntax generated when you select table A then table B and create a left outer join is equivalent to the syntax generated when you select table B then table A and create a right outer join.

For more about outer joins, see your DBMS documentation.

Using retrieval arguments

Adding retrieval arguments

If you know which rows will be retrieved into the DataWindow object at runtime—that is, if you can fully specify the SELECT statement without having to provide a variable—you do not need to specify retrieval arguments.

If you decide later that you need arguments, you can return to the Select painter to define the arguments.

Defining retrieval arguments in the DataWindow painter

You can select View>Column Specifications from the menu bar. In the Column Specification view, a column of check boxes next to the columns in the data source lets you identify the columns users should be prompted for. This, like the Retrieval Arguments prompt, calls the Retrieve method.

See Chapter 5, “Enhancing DataWindow Objects.”

If you want the user to be prompted to identify which rows to retrieve, you can define retrieval arguments when defining the SQL SELECT statement. For example, consider these situations:

- Retrieving the row in the Employee table for an employee ID entered into a text box. You must pass that information to the SELECT statement as an argument at runtime.
- Retrieving all rows from a table for a department selected from a drop-down list. The department is passed as an argument at runtime.

Using retrieval arguments at runtime

If a DataWindow object has retrieval arguments, call the Retrieve method of the DataWindow control to retrieve data at runtime and pass the arguments in the method.

❖ **To define retrieval arguments:**

- 1 In the Select painter, select Design>Retrieval Arguments from the menu bar.
- 2 Enter a name and select a datatype for each argument.

You can enter any valid SQL identifier for the argument name. The position number identifies the argument position in the Retrieve method you code in a script that retrieves data into the DataWindow object.
- 3 Click Add to define additional arguments as needed and click OK when done.

Specifying an array as a retrieval argument

You can specify an array of values as your retrieval argument. Choose the type of array from the Type drop-down list in the Specify Retrieval Arguments dialog box. You specify an array if you want to use the IN operator in your WHERE clause to retrieve rows that match one of a set of values. For example:

```
SELECT * from employee
WHERE dept_id IN (100, 200, 500)
```

retrieves all employees in department 100, 200, or 500. If you want your user to specify the list of departments to retrieve, you define the retrieval argument as a number array (such as *100, 200, 500*).

In the code that does the retrieval, you declare an array and reference it in the Retrieve method.

DataWindow Designer passes the appropriate comma-delimited list to the method (such as *100, 200, 500* if $x[1] = 100$, $x[2] = 200$, and $x[3] = 500$).

When building the SELECT statement, you reference the retrieval arguments in the WHERE or HAVING clause, as described in the next section.

Specifying selection, sorting, and grouping criteria

In the SELECT statement associated with a DataWindow object, you can add selection, sorting, and grouping criteria that are added to the SQL statement and processed by the DBMS as part of the retrieval.

Table 4-4: Adding selection, sorting, and grouping criteria to the SELECT statement

To do this	Use this clause
Limit the data that is retrieved from the database	WHERE
Sort the retrieved data before it is brought into the DataWindow object	ORDER BY

To do this	Use this clause
Group the retrieved data before it is brought into the DataWindow object	GROUP BY
Limit the groups specified in the GROUP BY clause	HAVING

Dynamically selecting, sorting, and grouping data

Selection, sorting, and grouping criteria that you define in the Select painter are added to the SQL statement and processed by the DBMS as part of the retrieval. You can also define selection, sorting, and grouping criteria that are created and processed dynamically by DataWindow Designer *after* data has been retrieved from the DBMS.

For more information, see Chapter 9, “Filtering, Sorting, and Grouping Rows.”

Referencing retrieval arguments

If you have defined retrieval arguments, you reference them in the WHERE or HAVING clause. In SQL statements, variables (called host variables) are always prefaced with a colon to distinguish them from column names.

For example, if the DataWindow object is retrieving all rows from the Department table where the dept_id matches a value provided by the user at runtime, your WHERE clause will look something like this:

```
WHERE dept_id = :Entered_id
```

where Entered_id was defined previously as an argument in the Specify Retrieval Arguments dialog box.

Referencing arrays

Use the IN operator and reference the retrieval argument in the WHERE or HAVING clause.

For example, if you reference an array defined as deptarray, the expression in the WHERE view might look like this:

```
"employee.dept_id" IN (:deptarray)
```

You need to supply the parentheses yourself.

Defining WHERE criteria

You can limit the rows that are retrieved into the DataWindow object by specifying selection criteria that correspond to the WHERE clause in the SELECT statement.

For example, if you are retrieving information about employees, you can limit the employees to those in Sales and Marketing, or to those in Sales and Marketing who make more than \$50,000.

❖ **To define WHERE criteria:**

- 1 Click the Where tab to make the Where view available (or select View>Where if the Where view is not currently displayed).

Each row in the Where view is a place for entering an expression that limits the retrieval of rows.

- 2 Click in the first row under Column to display columns in a drop-down list, or select Columns from the pop-up menu.
- 3 Select the column you want to use in the left-hand side of the expression. The equality (=) operator displays in the Operator column.

Using a function or retrieval argument in the expression

To use a function, select Functions from the pop-up menu and click a listed function. These are the functions provided by the DBMS.

To use a retrieval argument, select Arguments from the pop-up menu. You must have defined a retrieval argument already.

- 4 (Optional) Change the default equality operator.
Enter the operator you want, or click to display a list of operators and select an operator.
- 5 Under Value, specify the right-hand side of the expression. You can:
 - Type a value.
 - Paste a column, function, or retrieval argument (if there is one) by selecting Columns, Functions, or Arguments from the pop-up menu.
 - Paste a value from the database by selecting Value from the pop-up menu, then selecting a value from the list of values retrieved from the database. (It may take some time to display values if the column has many values in the database.)
 - Define a nested SELECT statement by selecting Select from the pop-up menu. In the Nested Select dialog box, you can define a nested SELECT statement. Click Return when you have finished.
- 6 Continue to define additional WHERE expressions as needed.

For each additional expression, select a logical operator (AND or OR) to connect the multiple boolean expressions into one expression that DataWindow Designer evaluates as true or false to limit the rows that are retrieved.

- 7 Define sorting (Sort view), grouping (Group view), and limiting (Having view) criteria as appropriate.
- 8 Click the Return button to return to the DataWindow painter.

Defining ORDER BY criteria

You can sort the rows that are retrieved into the DataWindow object by specifying columns that correspond to the ORDER BY clause in the SELECT statement.

For example, if you are retrieving information about employees, you can sort on department, and then within each department, you can sort on employee ID.

❖ To define ORDER BY criteria:

- 1 Click the Sort tab to make the Sort view available (or select View>Sort if the Sort view is not currently displayed).

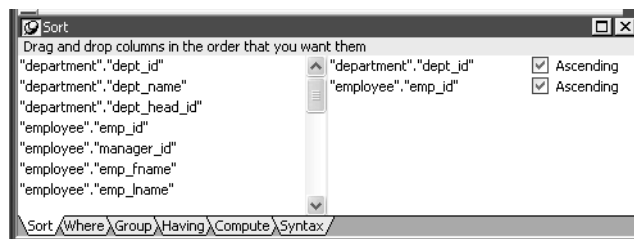
The columns you selected display in the order of selection. You might need to scroll to see your selections.

- 2 Drag the first column you want to sort on to the right side of the Sort view.

This specifies the column for the first level of sorting. By default, the column is sorted in ascending order. To specify descending order, clear the Ascending check box.

- 3 Continue to specify additional columns for sorting in ascending or descending order as needed.

You can change the sorting order by dragging the selected column names up or down. With the following sorting specification, rows will be sorted first by department name, then by employee ID:



Defining GROUP BY criteria

- 4 Define limiting (Where view), grouping (Group view), and limiting groups (Having view) criteria as appropriate.
- 5 Click the SQL Select button to return to the DataWindow painter.

You can group the retrieved rows by specifying groups that correspond to the GROUP BY clause in the SELECT statement. This grouping happens *before* the data is retrieved into the DataWindow object. Each group is retrieved as one row into the DataWindow object.

For example, if in the SELECT statement you group data from the Employee table by department ID, you will get one row back from the database for every department represented in the Employee table. You can also specify computed columns, such as total and average salary, for the grouped data. This is the corresponding SELECT statement:

```
SELECT dept_id, sum(salary), avg(salary)
FROM employee
GROUP BY dept_id
```

If you specify this with the Employee table in the EAS Demo DB, you get five rows back, one for each department.

Dept ID	Sum(salary)	Avg(salary)
100	\$1,292,198	\$58,736
200	\$919,428	\$48,391
300	\$535,500	\$59,500
400	\$698,251	\$43,641
500	\$315,730	\$35,081

For more about GROUP BY, see your DBMS documentation.

❖ **To define GROUP BY criteria:**

- 1 Click the Group tab to make the Group view available (or select View>Group if the Group view is not currently displayed).

The columns in the tables you selected display in the left side of the Group view. You might need to scroll to see your selections.

- 2 Drag the first column you want to group onto the right side of the Group view.

This specifies the column for grouping. Columns are grouped in the order in which they are displayed in the right side of the Group view.

- 3 Continue to specify additional columns for grouping within the first grouping column as needed.

To change the grouping order, drag the column names in the right side to the positions you want.

- 4 Define sorting (Sort view), limiting (Where view), and limiting groups (Having view) criteria as appropriate.
- 5 Click the Return button to return to the DataWindow painter.

Defining HAVING criteria

If you have defined groups, you can define HAVING criteria to restrict the retrieved groups. For example, if you group employees by department, you can restrict the retrieved groups to departments whose employees have an average salary of less than \$50,000. This corresponds to:

```
SELECT dept_id, sum(salary), avg(salary)
FROM employee
GROUP BY dept_id
HAVING avg(salary) < 50000
```

If you specify this with the Employee table in the EAS Demo DB, you will get three rows back, because there are three departments that have average salaries less than \$50,000.

Dept ID	Sum[salary]	Avg[salary]
200	\$919,428	\$48,391
400	\$698,251	\$43,641
500	\$315,730	\$35,081

❖ To define HAVING criteria:

- Click the Having tab to make the Having view available (or select View>Having if the Having view is not currently displayed).

Each row in the Having view is a place for entering an expression that limits which groups are retrieved. For information on how to define criteria in the Having view, see the procedure in “Defining WHERE criteria” on page 131.

Using Query

When you choose Query as the data source, you select a predefined SQL SELECT statement (a query) as specifying the data for your DataWindow object.

❖ **To define the data using Query:**

- 1 While using any of the DataWindow wizards, click Query in the Choose Data Source dialog box. Then click Next.

The Select Query dialog box displays.

- 2 Type the name of a query or use the Browse button to find the query. Then click Next.
- 3 Finish interacting with the DataWindow object wizard as needed for the presentation style you are using.

To learn how to create queries, see “Defining queries” on page 145.

Using External

If the data for the DataWindow object does not come from a database, specify External as the data source. You then specify the data columns and their types so DataWindow Designer can build the appropriate DataWindow object to hold the data. These columns make up the result set. DataWindow Designer places the columns you specified in the result set in the DataWindow object.

❖ **To define the data using External:**

- 1 Click External in the Choose Data Source dialog box in the wizard and click Next.

The Define Result Set dialog box displays for you to specify the first column in the result set.

- 2 Enter the name and type of the column.

Available datatypes are listed in the drop-down list.

- 3 Click Add to enter the name and type of any additional columns you want in the result set.
- 4 Click Next when you have added all the columns you want.

What you do next In a script, you need to tell DataWindow Designer how to get data into the DataWindow object in your application. Typically, you import data at runtime using a method (such as `ImportFile` and `ImportString`) or do some data manipulation and use a `SetItem` method to populate the DataWindow.

For more about these methods, see the online help.

You can also import data values from an external file into the DataWindow object or report.

Using Stored Procedure

A stored procedure is a set of precompiled and preoptimized SQL statements that performs some database operation. Stored procedures reside where the database resides, and you can access them as needed.

Defining data using a stored procedure You can specify a stored procedure as the data source for a DataWindow object if your DBMS supports stored procedures.

For information on support for stored procedures, see your database documentation.

If the Stored Procedure icon is not displayed

The icon for the Stored Procedure data source displays in the Choose Data Source dialog box in the DataWindow object wizards only if the database to which you are connected supports stored procedures.

❖ **To define the data using Stored Procedure:**

- 1 Select Stored Procedure in the Choose Data Source dialog box in the wizard and click Next.

The Select Stored Procedure dialog box displays a list of the stored procedures in the current database.
- 2 Select a stored procedure from the list.

To list system procedures, select the System Procedure check box.

The syntax of the selected stored procedure displays below the list of stored procedures.
- 3 Specify how you want the result set description built:

- To build the result set description automatically, clear the Manual Result Set check box and click Next.

DataWindow Designer executes the stored procedure and builds the result set description for you.

- To define the result set description manually, select the Manual Result Set check box and click Next.

In the Define Stored Procedure Result Set dialog box:

- Enter the name and type of the first column in the result set.
- To add additional columns, click Add.

Your preference is saved

DataWindow Designer records your preference for building result set descriptions for stored procedure DataWindow objects in the variable *Stored_Procedure_Build* in the DataWindow Designer initialization file. If this variable is set to 1, DataWindow Designer will automatically build the result set; if the variable is set to 0, you are prompted to define the result set description.

- 4 Continue in the DataWindow object wizard as needed for the presentation style you are using.

When you have finished interacting with the wizard, you go to the DataWindow painter with the columns specified in the result set placed in the DataWindow object.

For information about defining retrieval arguments for DataWindow objects, see Chapter 5, “Enhancing DataWindow Objects.”

Editing a result set description

After you create a result set that uses a stored procedure, you can edit the result set description from the DataWindow painter.

❖ **To edit the result set description:**

- 1 Select Design>Data Source from the menu bar.

This displays the Column Specification view if it is not already displayed.

- 2 Select Stored Procedure from the Column Specification view’s pop-up menu.

The Modify Stored Procedure dialog box displays.

- 3 Edit the Execute statement, select another stored procedure, or add arguments.

The syntax is:

```
execute sp_procname;num arg1 = :arg1, arg2 = :arg2..., argn =:argn
```

where `sp_procname` is the name of the stored procedure, `num` is the stored procedure group suffix, and `arg1`, `arg2`, and `argn` are the stored procedure's arguments.

The group suffix is an optional integer used in some DBMSs to group procedures of the same name so that they can be dropped together with a single `DROP PROCEDURE` statement. For other DBMSs the number is ignored.

- 4 When you have defined the entire result set, click OK.

You return to the DataWindow painter with the columns specified in the result set placed in the DataWindow object.

For information about defining retrieval arguments for DataWindow objects, see Chapter 5, "Enhancing DataWindow Objects."

Using ADO DataSet

If you want to use a DataTable in an ADO.NET DataSet to build the DataWindow, specify ADO DataSet as the data source. A File Open dialog box displays so that you can select a typed DataSet in an XML schema definition (XSD) file. You can then select one (and only one) of the DataTables in the DataSet.

All the columns in the selected DataTable are included in the DataWindow. If the DataTable is a result set from a join of multiple tables, the DataWindow will include columns from multiple tables.

Crosstab not supported

You cannot use the Crosstab presentation style with the ADO DataSet data source.

❖ **To define the data using ADO DataSet:**

- 1 Click ADO DataSet in the Choose Data Source dialog box in the wizard and click Next.

A file selection dialog box displays.

- 2 Select the XSD file you want to use to access the data.
- 3 Select one DataTable that you will use as the data source for the DataWindow object.
- 4 Complete the DataWindow wizard as needed for the presentation style you are using.

When you have finished interacting with the wizard, you go to the DataWindow painter. All the columns in the DataTable display in the DataWindow object. You can manipulate the presentation characteristics in the Design view.

Previewing the DataWindow

You cannot retrieve rows into the DataWindow object in DataWindow Designer, but you can import rows from an external file as described in “Using External” on page 136.

Datatype mappings

Table 4-5 lists .NET datatypes and the DataWindow datatypes to which they map when you use a DataTable as a data source.

Table 4-5: Datatype mappings from .NET to the DataWindow

.NET datatype	DataWindow datatype
System.Boolean	long (handled as a boolean at runtime)
System.Byte	long
System.Char	string(1)
System.DateTime	datetime
System.Decimal	decimal(0,0)
System.Double	number
System.Guid	not supported
System.Int16	long
System.Int32	long
System.Int64	decimal
System.SByte	ulong
System.Single	real
System.String	string(40)
System.Timespan	real
System.UInt16	ulong
System.UInt32	ulong
System.UInt64	decimal
System.Byte[]	not supported

Defaults for string and decimal datatypes

The System.String type does not always specify a maximum length. The length of string columns defaults to 40 characters in the DataWindows. The System.Decimal types do not specify a precision, and the precision defaults to 0 in the DataWindow. You can change these defaults in the Column Specifications view in the DataWindow painter.

For more information about using DataSets with DataWindows, see the DataWindow .NET *Programmer's Guide*.

Choosing DataWindow object-wide options

You can set the default options, such as colors and borders, that DataWindow Designer uses in creating the initial draft of a DataWindow object.

DataWindow generation options are for styles that use a layout made up of bands, which include Freeform, Grid, Label, N-Up, Tabular, Group, TreeView, and Crosstab. DataWindow Designer maintains a separate set of options for each of these styles.

When you first create any of these style DataWindow objects, you can choose options in the wizard and save your choices as the future defaults for the style.

❖ **To specify default colors and borders for a style:**

- 1 Select Design>Options from the menu bar.

The DataWindow Options dialog box displays.

- 2 Select the Generation tab page if it is not on top.
- 3 Select the presentation style you want from the Presentation Style drop-down list.

The values for properties shown on the page are for the currently selected presentation style.

- 4 Change one or more of the following properties:

Property	Meaning for the DataWindow object
Background color	The default color for the background.
Text border and color	The default border and color used for labels and headings.
Column border and color	The default border and color used for data values.
Wrap Height (Freeform only)	The height of the detail band. When the value is None, the number of columns selected determines the height of the detail band. The columns display in a single vertical line. When the value is set to a number, the detail band height is set to the number specified and columns wrap within the detail band.

- 5 Click OK.

About color selections

If you select Window Background, Application Workspace, Button Face, or Window Text from the Color drop-down list, the DataWindow object uses the colors specified in the Windows Control Panel on the computer on which the DataWindow object is running.

Your choices are saved

DataWindow Designer saves your generation option choices as the defaults to use when creating a DataWindow object with the same presentation style.

Generating and saving a DataWindow object

When you have finished interacting with the wizard, DataWindow Designer generates the DataWindow object and opens the DataWindow painter.

When generating the DataWindow object, DataWindow Designer might use information from a set of tables called the extended attribute system tables. If this information is available, DataWindow Designer uses it.

About the extended attribute system tables and DataWindow objects

The extended attribute system tables are a set of tables maintained by the Database painter. They contain information about database tables and columns. Extended attribute information extends database definitions by recording information that is relevant to using database data in screens and reports.

For example, labels and headings you defined for columns in the Database painter are used in the generated DataWindow object. Similarly, if you associated an edit style with a column in the Database painter, that edit style is automatically used for the column in the DataWindow object.

When generating a DataWindow object, DataWindow Designer uses the following information from the extended attribute system tables:

For	DataWindow Designer uses
Tables	Fonts specified for labels, headings, and data
Columns	Text specified for labels and headings Display formats Validation rules Edit styles

If there is no extended attribute information for the database tables and columns you are using, you can set the text for headings and labels, the fonts, and the display formats in the DataWindow painter. The difference is that you have to do this individually for every DataWindow object that you create using the data.

If you want to change something that came from the extended attribute system tables, you can change it in the DataWindow painter. The changes you make in the DataWindow painter apply only to the DataWindow object you are working on.

The advantage of using the extended attribute system tables is that it saves time and ensures consistency. You only have to specify the information once, in the database. Since DataWindow Designer uses the information whenever anyone creates a new DataWindow object with the data, it is more likely that the appearance and labels of data items will be consistent.

For more information about the extended attribute system tables, see Chapter 3, “Managing the Database,” and Appendix B, “The Extended Attribute System Tables.”

Saving the DataWindow object

When you have created a DataWindow object, you should save it. The first time you save it you give it a name. As you work, you should save your DataWindow object frequently so that you do not lose changes.

❖ **To save the DataWindow object:**

- 1 Select File>Save from the menu bar.

If you have previously saved the DataWindow object, DataWindow Designer saves the new version in the same library and returns you to the DataWindow painter.

If you have not previously saved the DataWindow object, DataWindow Designer displays the Save DataWindow dialog box.

- 2 (Optional) Enter comments in the Comments box to describe the DataWindow object.
- 3 Enter a name for the DataWindow object in the DataWindows box.
- 4 Specify the library in which the DataWindow object is to be saved and click OK.

Naming the DataWindow object

The DataWindow object name can be any valid DataWindow Designer identifier up to 40 contiguous characters. A common convention is to prefix the name of the DataWindow object with d_.

For information about DataWindow Designer identifiers, see Appendix A, “Identifiers.”

Modifying an existing DataWindow object

❖ **To modify an existing DataWindow object:**

- 1 Select File>Open from the menu bar.

The Open dialog displays.

- 2 Select the object type and the library.

DataWindow Designer lists the DataWindow objects in the current library.

- 3 Select the object you want.

DataWindow Designer opens the DataWindow painter and displays the DataWindow object. You can also open a DataWindow object by double-clicking it in the System Tree.

To learn how you can modify an existing DataWindow object, see Chapter 5, “Enhancing DataWindow Objects.”

Defining queries

A query is a SQL `SELECT` statement created in the Query painter and saved with a name so that it can be used repeatedly as the data source for a DataWindow object.

Queries save time, because you specify all the data requirements just once. For example, you can specify the columns, which rows to retrieve, and the sorting order in a query. Whenever you want to create a DataWindow object using that data, simply specify the query as the data source.

❖ **To define a query:**

- 1 Select File>New from the menu bar.
- 2 In the New dialog box, select the Database tab.
- 3 Select the Query icon and click OK.
- 4 Select tables in the Select Tables dialog box and click Open.

You can select columns, define sorting and grouping criteria, define computed columns, and so on, exactly as you do when creating a DataWindow object using the SQL Select data source.

For more about defining the `SELECT` statement, see “Using SQL Select” on page 121.

Previewing the query

While creating a query, you can preview it to make sure it is retrieving the correct rows and columns.

❖ **To preview a query:**

- 1 Select Design>Preview from the menu bar.

DataWindow Designer retrieves the rows satisfying the currently defined query in a grid-style DataWindow object.

- 2 Manipulate the retrieved data as you do in the Database painter in the Output view.

You can sort and filter the data, but you cannot insert or delete a row or apply changes to the database. For more about manipulating data, see Chapter 3, “Managing the Database.”

- 3 When you have finished previewing the query, click the Close button in the PainterBar to return to the Query painter workspace.

Saving the query

❖ **To save a query:**

- 1 Select File>Save Query from the menu bar.

If you have previously saved the query, DataWindow Designer saves the new version and returns you to the Query painter. If you have not previously saved the query, DataWindow Designer displays the Save Query dialog box.

- 2 Enter a name for the query in the Queries box (see “Naming the query” next).

- 3 (Optional) Enter comments to describe the query,.

These comments display in the Library painter. It is a good idea to use comments to remind yourself and others of the purpose of the query.

- 4 Specify the library in which to save the query, and click OK.

Naming the query The query name can be any valid DataWindow Designer identifier up to 40 characters. When you name queries, use a unique name to identify each one. A common convention is to use a two-part name: a standard prefix that identifies the object as a query (such as q_) and a unique suffix. For example, you might name a query that displays employee data q_emp_data. For information about DataWindow Designer identifiers, see Appendix A, “Identifiers.”

Modifying a query

❖ **To modify a query:**

- 1 Select File>Open from the menu bar.
- 2 Select the Queries object type and then the query you want to modify, and click OK.
- 3 Modify the query as needed.

What's next

After you have generated your DataWindow object, you will probably want to preview it to see how it looks. After that, you might want to enhance the DataWindow object in the DataWindow painter before using it. DataWindow Designer provides many ways for you to make a DataWindow object easier to use and more informative for users. See Chapter 5, “Enhancing DataWindow Objects,” next.

Enhancing DataWindow Objects

About this chapter

Before you put a DataWindow object into production, you can enhance it to make it easier to use and interpret data. You do that in the DataWindow painter. This chapter describes basic enhancements you can make to a DataWindow object.

Contents

Topic	Page
Working in the DataWindow painter	150
Using the Preview view of a DataWindow object	158
Saving data in an external file	168
Modifying general DataWindow object properties	171
Storing data in a DataWindow object using the Data view	185
Retrieving data	187

Related topics

Other ways to enhance DataWindow objects are covered in later chapters:

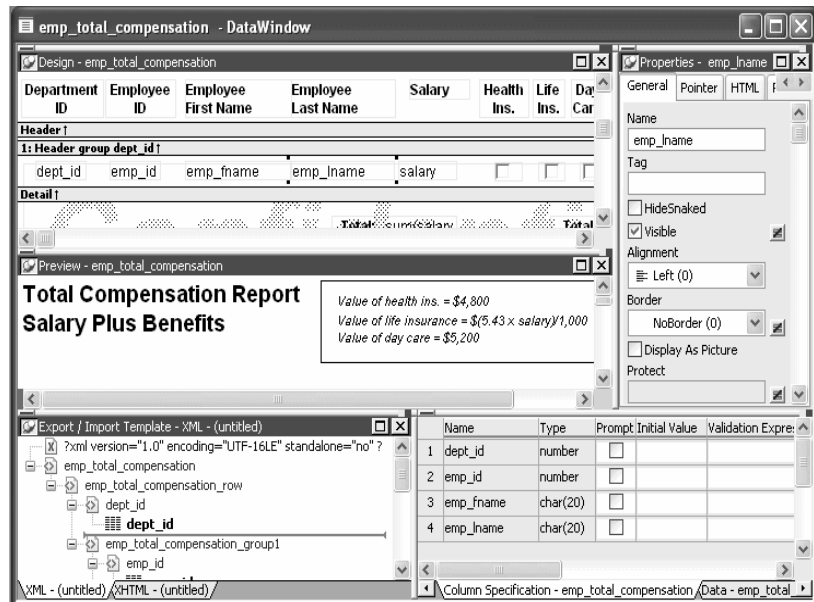
Chapter	Explains how to
Chapter 7, “Working with Controls in DataWindow Objects”	Add controls to a DataWindow object and reorganize, position, and rotate them
Chapter 6, “Controlling Updates in DataWindow Objects”	Control update capabilities
Chapter 8, “Displaying and Validating Data”	Specify display formats, edit styles, and validation rules for column data
Chapter 9, “Filtering, Sorting, and Grouping Rows”	Limit which rows are displayed, the order in which they are displayed, and whether they are divided into groups
Chapter 10, “Highlighting Information in DataWindow Objects”	Highlight data by using conditional expressions to modify the properties of controls in DataWindow objects
Chapter 11, “Using Nested Reports”	Place reports inside DataWindow objects
Chapter 15, “Working with Graphs”	Use graphs to visually present information retrieved in a DataWindow object

Chapter	Explains how to
Chapter 12, "Working with Crosstabs"	Use crosstabs to present analyses of data retrieved in a DataWindow object
Chapter 13, "Working with TreeViews"	Use TreeViews to group data and display it hierarchically in a way that allows you to expand and collapse it

Working in the DataWindow painter

The DataWindow painter provides views related to the DataWindow object you are working on. Interacting with these views is how you work in the DataWindow painter.

The following picture shows a DataWindow object in the DataWindow painter with the default layout.



Design view

The Design view at the top left shows a representation of the DataWindow object and its controls. You use this view to design the layout and appearance of the DataWindow object. Changes you make are immediately shown in the Preview view and the Properties view.

Preview view	The Preview view in the middle on the left shows the DataWindow object with data as it will appear at runtime. If the Print Preview toggle is selected, you see the DataWindow object as it would appear when printed.
Export/Import Template view for XML	The Export/Import Template view for XML at the bottom left shows a default template for exporting and importing data in XML format. You can define custom templates for import and export. The templates are saved with the DataWindow object. For more information, see Chapter 14, “Exporting and Importing XML Data.”
Export Template view for XHTML	The Export Template view for XHTML (not shown; see XHTML tab at the bottom left) shows a default template for exporting data in XHTML format. You can define custom XHTML export templates for customizing XML Web DataWindow generation. The templates are saved with the DataWindow object. For more information, see the <i>DataWindow .NET Programmer’s Guide</i> .
Properties view	The Properties view at the top right displays the properties for the currently selected control(s) in the DataWindow object, for the currently selected band in the DataWindow object, or for the DataWindow object itself. You can view and change the values of properties in this view.
Control List view	The Control List view in the stacked pane at the bottom right lists all controls in the DataWindow object. Selecting controls in this view selects them in the Design view and the Properties view. You can also sort controls by Control Name, Type, or Tag.
Data view	The Data view in the stacked pane at the bottom right displays the data that can be used to populate a DataWindow object and allows manipulation of that data.
Column Specifications view	The Column Specifications view in the stacked pane at the bottom right shows a list of the columns in the data source. For the columns, you can add, modify, and delete initial values, validation expressions, and validation messages. You can also specify that you want a column to be included in a prompt for retrieval criteria during data retrieval. To add a column to the DataWindow object, you can drag and drop the column from the Column Specifications view to the Design view. For external or stored procedure data sources, you can add, delete, and edit columns (column name, type, and length).

Understanding the DataWindow painter Design view

For most presentation styles, the DataWindow painter Design view is divided into areas called bands. Each band corresponds to a section of the displayed DataWindow object.

DataWindow objects with these presentation styles are divided into four bands: header, detail, summary, and footer. Each band is identified by a bar containing the name of the band above the bar and an Arrow pointing to the band.

These bands can contain any information you want, including text, drawing controls, graphs, and computed fields containing aggregate totals.

The following picture shows the Design view for a tabular DataWindow object.

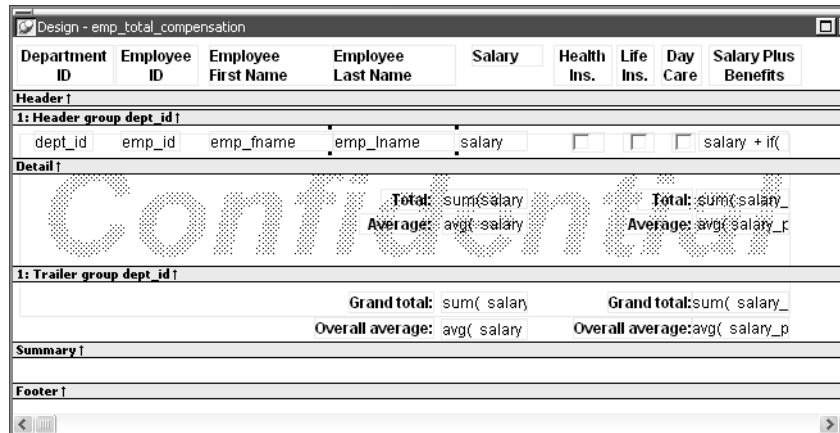


Table 5-1: Bands in the DataWindow painter Design view

Band	Used to display
Header	Information at the top of every screen or page, such as the name of the report or current date
Detail	Data from the database or other data source
Summary	Summary information that displays after all the data, such as totals and counts
Footer	Information displayed at the bottom of every page or screen, such as page number and page count

The header band

The header band contains heading information that is displayed at the top of every screen or page. The presentation style determines the contents of the header band:

- If the presentation style is Tabular, Grid, or N-Up, the headings defined for the columns in the Database painter display in the header band and the columns display on a single line across the detail band
- If the presentation style is Freeform, the header band is empty and labels display in the detail band next to each column

You can specify additional heading information (such as a date) in the header band and you can include pictures, graphic controls, and color to enhance the appearance of the band.

Displaying the current date

To include the current date in the header, you place a computed field that uses the Today DataWindow expression function in the header band. For information, see “Adding computed fields to a DataWindow object” on page 205.

The detail band

The detail band displays the retrieved data. It is also where the user enters new data and updates existing data. The number of rows of data that display in the DataWindow object at one time is determined by the following expression:

$$\frac{(\text{Height of the DataWindow object} - \text{Height of headers and footers})}{\text{Height of the detail band}}$$

The presentation style determines the contents of the detail band:

- If the presentation style is Tabular, Grid, N-Up, or Label, the detail band displays column names, representing the columns
- If the presentation style is Freeform, the labels defined for the columns in the Database painter display in the detail band with boxes for the data to the right

How DataWindow Designer names the columns in the Design view

If the DataWindow object uses one table, the names of the columns in the Design view are the same as the names in the table.

If the DataWindow object uses more than one table, the names of the columns in the Design view are *tablename_columnname*. DataWindow Designer prefaces the name of the column with the table name to prevent ambiguity, since different tables can have columns with the same name.

When you design the detail band of a DataWindow object, you can specify display and validation information for each column of the DataWindow object and add other controls, such as text, pictures, graphs, and drawing controls.

The summary and footer bands

You use the summary and footer bands of the DataWindow object the same way you use summary pages and page footers in a printed report:

- The contents of the summary band display at the end, after all the detail rows; this band often summarizes information in the DataWindow object
- The contents of the footer band display at the bottom of each screen or page of the DataWindow object; this band often displays the page number and name of the report

Using the DataWindow painter toolbars

The DataWindow painter contains three customizable PainterBars and a StyleBar.

For more information about using toolbars, see “Using toolbars” on page 21.

PainterBars

The PainterBars include buttons for standard operations (such as Save, Print, and Undo on PainterBar1), for common formatting operations (such as Currency, Percent, and Tab Order on PainterBar2), and for database operations (such as Retrieve and Insert Row on PainterBar3).

They also include six drop-down toolbars, which are indicated by a small black triangle on the right part of a button. Table 5-2 lists the drop-down toolbars that are available. The Controls toolbar is on PainterBar1. The other drop-down toolbars are on PainterBar2.

Table 5-2: Drop-down toolbars in the DataWindow painter

Toolbar	Used to
Background Color	Specify the background color of one or more selected controls.
Borders	Specify borders for one or more selected controls.
Controls	Specify controls to add to a DataWindow object.
Foreground Color	Specify the foreground color of one or more selected controls. In a text control, the foreground color specifies the color of the text.
Layout	Specify the alignment, sizing, and spacing of selected controls.
Slide	Specify sliding for controls.

StyleBar

The StyleBar includes buttons for applying properties (such as bold) to selected text elements.

Using the Properties view in the DataWindow painter

Each part of the DataWindow object (such as text, columns, computed fields, bands, even the DataWindow object itself) has a set of properties appropriate to the part. The properties display in the Properties view.

You can use the Properties view to modify the parts of the DataWindow object.

❖ **To use the Properties view to modify the parts of the DataWindow object:**

- 1 Position the mouse over the part you want to modify.
- 2 Display the part's pop-up menu and select Properties.

If it is not already displayed, the Properties view displays. The view displays the properties of the currently selected control(s), the band, or the DataWindow object itself. The contents of the Properties view change as different controls are selected (made current).

For example, the Properties view for a column has seven tabbed property pages of information that you access by clicking the appropriate tab. If you want to choose an edit style for the column, you click the Edit tab. This brings the Edit page to the front of the Properties view.

Selecting controls in the DataWindow painter

The DataWindow painter provides several ways to select controls to act on. You can select multiple controls and act on all the selected controls as a unit. For example, you can move all of them or change the fonts used to display text for all of them.

Lasso selection

Use lasso selection when possible because it is fast and easy. Lasso selection is another name for the method described below for selecting neighboring multiple controls.

❖ **To select one control in a DataWindow object in the Design view:**

- Click it.

The control displays with handles on it. Previously selected controls are no longer selected.

❖ **To select neighboring multiple controls in a DataWindow object in the Design view (lasso selection):**

- 1 Press and hold the left mouse button at one corner of the neighboring controls.
- 2 Drag the mouse over the controls you want to select.
A bounding box (the lasso) displays.
- 3 Release the mouse button.

All the controls in the bounding box are selected.

❖ **To select non-neighboring multiple controls in a DataWindow object in the Design view:**

- 1 Click the first control.
- 2 Press and hold the Ctrl key and click additional controls.

All the controls you click are selected.

❖ **To select controls by type in the DataWindow object:**

- Do one of the following:
 - Select Edit>Select>Select All to select all controls
 - Select Edit>Select>Select Text to select all text
 - Select Edit>Select>Select Columns to select all columns

- ❖ **To select controls by position in the DataWindow object:**
 - Do one of the following:
 - Select Edit>Select>Select Above to select all controls above the currently selected control
 - Select Edit>Select>Select Below to select all controls below it
 - Select Edit>Select>Select Left to select all controls to the left of it
 - Select Edit>Select>Select Right to select all controls to the right of it
- ❖ **To select controls in a DataWindow object in the Control List view:**
 - 1 Select View>Control List from the menu bar.
 - 2 Click a control in the list.
 - 3 Press and hold the Ctrl key and click additional controls if desired.

Displaying information about the selected control

The name, x and y coordinates, width, and height of the selected control are displayed in the MicroHelp bar. If multiple controls are selected, *Group Selected* displays in the Name area and the coordinates and size do not display.

Resizing bands in the DataWindow painter Design view

You can change the size of any band in the DataWindow object.

- ❖ **To resize a band in the DataWindow painter Design view:**
 - Position the pointer on the bar representing the band and drag the bar up or down to shrink or enlarge the band.

Using zoom in the DataWindow painter

You can zoom the display in and out in four views in the DataWindow painter: the Design view, Preview view, Data View, and Column Specifications view. For example, if you are working with a large DataWindow object, you can zoom out the Design view so you can see all of it on your screen, or you can zoom in on a group of controls to better see their details.

❖ **To zoom the display in the DataWindow painter:**

- 1 Select the view you want to zoom (click in the view).

You can zoom the Design view, Preview view, Data View, and Column Specifications view.

- 2 Select Design>Zoom from the menu bar.
- 3 Select a built-in zoom percentage, or set a custom zoom percentage by typing an integer in the Custom box.

Undoing changes in the DataWindow painter

You can undo your change by pressing Ctrl+Z or selecting Edit>Undo from the menu bar. Undo requests affect all views.

Using the Preview view of a DataWindow object

You use the Preview view of a DataWindow object to view it as it will appear with data and test the processing that takes place in it.

❖ **To display the Preview view of a DataWindow object open in the DataWindow painter:**

- 1 If the Preview view is not already displayed, select View>Preview from the menu bar.

In the Preview view, the bars that indicate the bands do not display, and, if you selected Retrieve on Preview in the DataWindow wizard, DataWindow Designer retrieves all the rows from the database. You are prompted to supply arguments if you defined retrieval arguments.

In external DataWindow objects

If the DataWindow object uses the External data source, no data is retrieved. You can import data, as described in “Importing data into a DataWindow object” on page 163.

In DataWindow objects that have stored data

If the DataWindow object has stored data in it, no data is retrieved from the database.

As the rows are being retrieved, the Retrieve button in the PainterBar changes to a Cancel button. You can click the Cancel button to stop the retrieval.

2 Test your DataWindow object.

For example, modify some data, update the database, re-retrieve rows, and so on, as described below.

Retrieving data

Where DataWindow Designer gets data

DataWindow Designer follows this order of precedence to supply the data in your DataWindow object:

- 1 If you have saved data in the DataWindow object, DataWindow Designer uses the saved rows from the DataWindow object and does not retrieve data from the database.
- 2 DataWindow Designer uses the data in the cache, if there is any.
- 3 If there is no data in the cache yet, DataWindow Designer retrieves data from the database automatically, with one exception. If the Retrieve on Preview option is off, you have to request retrieval explicitly, as described next.

Previewing without retrieving data

If you do not want DataWindow Designer to retrieve data from the database automatically when the Preview view opens, you can set the Retrieve on Preview option. The Preview view shows the DataWindow object without retrieving data.

❖ **To be able to preview without retrieving data automatically:**

- 1 Select Design>Options from the menu bar.

The DataWindow Options dialog box displays.

- 2 Clear the Retrieve on Preview check box on the General page.

When this check box is cleared, your request to preview the DataWindow object does not result in automatic data retrieval from the database.

Retrieve on Preview check box is available in the DataWindow wizards

During the initial creation of a DataWindow object, you can set the Retrieve on Preview option.

DataWindow Designer uses data caching When DataWindow Designer first retrieves data, it stores the data internally. When it refreshes the Preview view, DataWindow Designer displays the stored data instead of retrieving rows from the database again. This can save you a lot of time, since data retrieval can be time consuming.

How using data from the cache affects you Because DataWindow Designer accesses the cache and does not automatically retrieve data every time you preview, you might not have what you want when you preview. The data you see in preview and the data in the database can be out of sync.

For example, if you are working with live data that changes frequently or with statistics based on changing data and you spend time designing the DataWindow object, the data you are looking at may no longer match the database. In this case, retrieve again just before printing.

Explicitly retrieving data You can explicitly request retrieval at any time.

❖ **To retrieve rows from the database:**

- Do one of the following:
 - Click the Retrieve button in the PainterBar.
 - Select Rows>Retrieve from the menu bar.
 - Select Retrieve from the Preview view's pop-up menu.

Supplying argument values or criteria

If the DataWindow object has retrieval arguments or is set up to prompt for criteria, you are prompted to supply values for the arguments or to specify criteria.

DataWindow Designer retrieves the rows. As DataWindow Designer retrieves, the Retrieve button changes to a Cancel button. You can click the Cancel button to stop the retrieval at any time.

Sharing data with the Data view The Data view displays data that can be used to populate a DataWindow object. When the ShareData pop-up menu item in the Data view is checked, changes you make in the Data view are reflected in the Preview view and vice versa.

Other options that affect retrieval These other options can affect retrieval:

- **Retrieve Rows As Needed** Lets you specify that only the rows needed to display the current portion of the DataWindow object should be retrieved. When you scroll downward, additional rows are retrieved. This can speed up reporting in certain situations.

See "Retrieving rows as needed" on page 188.

- **Retrieve Rows to Disk** Lets you specify that DataWindow Designer should save retrieved data on your hard disk in a temporary file rather than keep the data in memory. When you preview the DataWindow object, DataWindow Designer swaps rows of data from the temporary file into memory as needed.

For information, see “Saving retrieved rows to disk” on page 189.

Modifying data

You can add, modify, or delete rows in the Preview view. When you have finished manipulating the data, you can apply the changes to the database.

If looking at data from a view or from more than one table

By default, you cannot update data in a DataWindow object that contains a view or more than one table. For more about updating DataWindow objects, see Chapter 6, “Controlling Updates in DataWindow Objects.”

❖ **To modify existing data:**

- Tab to the field and enter a new value.

The Preview view uses validation rules, display formats, and edit styles that you have defined for the columns, either in the Database painter or in this particular DataWindow object.

To save the changes to the database, you must apply them, as described below.

❖ **To add a row:**

- 1 Click the Insert Row button.

DataWindow Designer creates a blank row.

- 2 Enter data for a row.

To save the changes to the database, you must apply them, as described below.

❖ **To delete a row:**

- Click the Delete Row button.

DataWindow Designer removes the row from the display.

To save the changes to the database, you must apply them, as described below.

❖ **To apply changes to the database:**

- Click the Update Database button.

DataWindow Designer updates the table with all the changes you have made.

Modifying data at runtime

Using the PainterBar buttons in the Preview view is equivalent to calling control functions such as InsertRow, DeleteRow, and UpdateData or using Button controls to perform actions at runtime.

Viewing row information

You can display information about the data you have retrieved.

❖ **To display the row information:**

- Select Rows>Described from the menu bar.

The Describe Rows dialog box displays, showing the number of:

- Rows that have been deleted in the painter *but not yet deleted from the database*
- Rows displayed in the Preview view
- Rows that have been filtered
- Rows that have been modified in the painter *but not yet modified in the database*

All row counts are zero until you retrieve the data from the database or add a new row. The count changes when you modify the displayed data or test filter criteria.

Importing data into a DataWindow object

You can import and display data from an external source and save the imported data in the database.

❖ **To import data into a DataWindow object:**

- 1 Select Rows>Import from the menu bar.
- 2 Specify the file from which you want to import the data.

The types of files that you can import into the painter display in the List Files of Type drop-down list.

- 3 Click Open.

DataWindow Designer reads the data from the file into the painter. You can then click the Update Database button in the PainterBar to add the new rows to the database.

Data from file must match DataWindow definition

When importing data from a file, the datatypes of the data must match, column for column, all the columns in the DataWindow definition (the columns specified in the SELECT statement), not just the columns that are displayed in the DataWindow object.

For information about importing XML data, see Chapter 14, “Exporting and Importing XML Data.”

Using print preview

You can print the data displayed in the Preview view. Before printing, you can preview the output on the screen. Your computer must have a default printer specified, otherwise properties handled by the printer driver, such as page orientation, are ignored.

❖ **To preview printed output before printing:**

- Be sure the Preview view is selected (current) and then select File>Print Preview from the menu bar.

Print Preview displays the DataWindow object as it will print.

Controlling the display of rulers

You can choose whether to display rulers around page borders.

❖ **To control the display of rulers in Print Preview:**

- Select/deselect File>Print Preview Rulers from the menu bar.

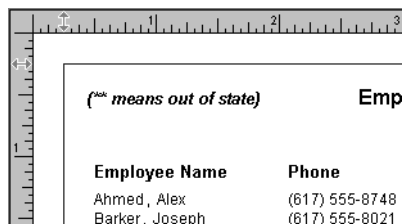
Changing margins

You can dynamically change margins while previewing a DataWindow object.

❖ **To change the margins in Print Preview:**

- Drag the margin boundaries on the rulers.

The following picture shows the left and top margin boundaries. There are also boundaries for the right and bottom margins. The picture shows the outline of the margin. If you do not want to see the outline, clear the Print Preview Shows Outline check box on the Print Specifications page in the Properties view.



Zooming the page

You can reduce or enlarge the amount of the page that displays in the Print Preview view. This does not affect the printed output.

❖ **To zoom the page on the display screen:**

- 1 Select File>Print Preview Zoom from the menu bar.
- 2 Select the magnification you want and click OK.

The display of the page zooms in or out as appropriate. The size of the contents of the page changes proportionately as you zoom. This type of zooming affects your display but does not affect printing.

Zooming the contents

In addition to zooming the display on the screen, you can also zoom the contents, affecting the amount of material that prints on a page.

❖ **To zoom the contents of a DataWindow object with respect to the printed page:**

- 1 Select Design>Zoom from the menu bar.
- 2 Select the magnification you want and click OK.

The contents of the page zooms in or out as appropriate. If you enlarge the contents so they no longer fit, DataWindow Designer creates additional pages as needed.

Printing data

You can print a DataWindow object while the Preview view is displayed. You can print all pages, a range of pages, only the current page, or only odd or even pages. You can also specify whether you want multiple copies, collated copies, and printing to a file.

Avoiding large rows

To avoid multiple blank pages and other anomalies in printed reports, no row in the DataWindow object should be larger than the size of the target page. The page boundary is often reached in long text columns with `AutoSizeHeight` on. It can also be reached when detail rows are combined with page and group headers and trailers, or when they contain multiple nested DataWindow objects or a column that has been resized to be larger than the page.

When a row contains large multiline edit columns, it can be broken into a series of rows, each containing one text line. These text lines become the source for a nested DataWindow object. The nested DataWindow object determines how many of its rows fit in the remaining page space.

Page break before last row

The summary band in a report is always printed on the same page as the last row of data. This means that you sometimes find a page break before the last row of data even if there is sufficient space to print the row. If you want the last row to print on the same page as the preceding rows, the summary band must be made small enough to fit on the page as well.

To change printers or settings before printing

You can choose `File>Printer Setup` from the menu bar.

❖ **To print a DataWindow object:**

- 1 Select `File>Print Report` from the menu bar to display the Print dialog box.
- 2 Specify the number of copies to print.
- 3 Specify the pages: select `All` or `Current Page`, or type page numbers and/or page ranges in the Pages box.
- 4 Specify all pages, even pages, or odd pages in the Print drop-down list.
- 5 If you want to print to a file rather than to the printer, select the `Print to File` check box.

- 6 If you want to change the collating option, clear or select the Collate Copies check box and click OK.

If you specified print to file, the Print to File dialog box displays.

- 7 Enter a file name and click OK.

The extension *PRN* indicates that the file is prepared for the printer. Change the drive, the directory, or both, if you want.

Working in a grid DataWindow object

If you are viewing a grid-style DataWindow object in the Preview view, you can make the following changes. Whatever you do in the Preview view is reflected in the Design view:

- Resize columns
- Reorder columns
- Split the display into two horizontal scrolling regions

You can use this feature to keep one or more columns stationary on the screen while scrolling through other columns.

- Copy data to the clipboard

These features are also available to your users

Users of your application can also manipulate columns in these ways in a grid DataWindow object at runtime.

❖ **To resize a column in a grid DataWindow object:**

- 1 Position the mouse pointer at a column boundary in the header.

The pointer changes to a two-headed arrow.

- 2 Press and hold the left mouse button and drag the mouse to move the boundary.
- 3 Release the mouse button when the column is the correct width.

❖ **To reorder columns in a grid DataWindow object:**

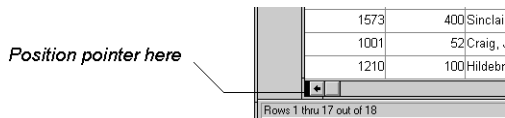
- 1 Press and hold the left mouse button on a column heading.

DataWindow Designer selects the column and displays a line representing the column border.

- 2 Drag the mouse left or right to move the column.
- 3 Release the mouse button.

❖ **To use split horizontal scrolling in a grid DataWindow object:**

- 1 To divide the grid into two regions that can scroll independently of each other, position the mouse pointer at the left end of the horizontal scrollbar.



The pointer changes to a two-headed arrow.

- 2 Press and hold the left mouse button and drag the mouse to the right to create a new horizontal scrolling border.
- 3 Release the mouse button.

You now have two independent scrolling regions in the grid DataWindow object.

❖ **To copy data to the clipboard from a grid DataWindow object:**

- 1 Select the cells whose data you want to copy to the clipboard:
 - To select an entire column, click its header.
 - To select neighboring columns, press and hold Shift, then click the headers.
 - To select non-neighboring columns, press and hold Ctrl, then click the headers.
 - To select cells, press the left mouse button on the bottom border of a cell and drag the mouse.

Selected cells are highlighted.

- 2 Select Edit>Copy from the menu bar.

The contents of the selected cells are copied to the clipboard. If you copied the contents of more than one column, the data is separated by tabs.

Saving data in an external file

While previewing, you can save the data retrieved in an external file. Note that the data and headers (if specified) are saved. Information in the footer or summary bands is not saved unless you are saving as PDF or as a PSR file.

❖ **To save the data in a DataWindow object in an external file:**

- 1 Select File>Save Rows As from the menu bar.

The Save As dialog box displays.

- 2 Choose a format for the file from the Save As Type drop-down list.

If you want the column headers saved in the file, select a file format that includes headers (such as Excel With Headers). When you select a *with headers* format, the names of the database columns (not the column labels) are also saved in the file.

When you choose a format, DataWindow Designer supplies the appropriate file extension.

- 3 For TEXT, CSV, SQL, HTML, and DIF formats, select an encoding for the file.

You can select ANSI/DBCS, Unicode LE (Little-Endian), Unicode BE (Big-Endian), or UTF8.

- 4 Name the file and click Save.

DataWindow Designer saves all displayed rows in the file; all columns in the displayed rows are saved. Filtered rows are not saved.

The rest of this section provides more information about saving data in PDF, HTML, and Powersoft Report (PSR) formats.

For more information about saving data as XML, see Chapter 14, “Exporting and Importing XML Data.”

Saving the data as PDF

When you save a report in Portable Document Format (PDF), the data is printed to a PostScript file and automatically distilled to PDF using GNU Ghostscript. You can save all types of reports to PDF except OLE and RichText reports.

DataWindow Designer uses a PostScript printer driver specifically designed for distilling purposes to configure the PDF output. You can choose to use a different PostScript printer driver if you want to customize your PostScript settings for generating PDF.

Users must have administrative privileges to create a PDF file.

To use a custom PostScript printer driver, you must set some properties.

- ❖ **To save customized distilled PDF output in the DataWindow painter:**
 - 1 Select the Data Export tab in the Properties view for the DataWindow object.
 - 2 Select PDF from the Format to Configure drop-down list, select Distill! from the Method drop-down list, and select the Distill Custom PostScript check box.
 - 3 Select the Print Specifications tab and specify the name of the printer whose settings you want to use in the Printer Name box.
 - 4 Save the DataWindow object, then select File>Save Rows As, select PDF as the Save As Type, specify a file name, and click Save.

System requirements

To support saving as PDF using GNU Ghostscript, you must install Ghostscript files from the GNU Web site. The default PostScript printer driver and related files are installed in the *drivers* directory.

When you deploy applications that use the ability to save as PDF using the distill method, the Ghostscript files and the *drivers* directory must be in the same directory as the *pbdwe105.dll* runtime file. For example, if you deploy your application and *pbdwe105.dll* and the other PowerBuilder runtime files in a directory called *MyApplication*, the Ghostscript files must be in *MyApplication\gsn.nn*, and the default PostScript printer driver and related files must be in *MyApplication\drivers*.

See the chapter on deploying your application in the *Programmer's Guide* for information about redistributing these files.

Saving the data in HTML Table format

HTML Table format is one of the formats in which you can choose to save data. When you save in HTML Table format, DataWindow Designer saves a style sheet along with the data. If you use this format, you can open the saved file in a browser such as Internet Explorer or Netscape. Once you have the file in HTML Table format, you can continue to enhance the file in HTML.

About the results

Some presentation styles translate better into HTML than others. The Tabular, Group, Freeform, Crosstab, and Grid presentation styles produce good results. The Composite and Graph presentation styles and nested reports produce HTML tables based on the result set (data) only and not on the presentation style. DataWindow Objects with overlapping controls in them might not produce the results you want.

❖ **To save a report as an HTML table:**

- 1 Open a DataWindow object.
- 2 Open the Preview view if it is not already open.
- 3 Select File>Save Rows As from the menu bar.
- 4 Choose the HTML Table format for the file from the Save As Type drop-down list.
- 5 Name the file.

DataWindow Designer creates a file using the name you supplied and the extension *htm*.

- 6 Open a browser such as Netscape or Internet Explorer.
- 7 Use the browser's file open command to open the HTML file.

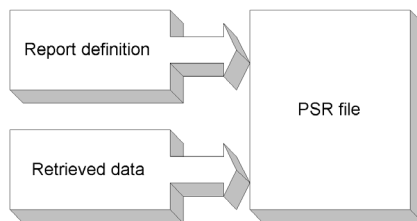
Working with PSR files

A PSR file is a special file with the extension PSR created by PowerBuilder, InfoMaker, or DataWindow Designer.

Windows and PSR files

When DataWindow Designer is installed, the PSR file type is registered with Windows.

A PSR file contains a report definition (source and object) as well as the data contained in the report when the PSR file was created.

Figure 5-1: PSR file

About reports

A report is the same as a nonupdatable DataWindow object. For more information, see “Reports versus DataWindow objects” on page 100.

You can use a PSR file to save a complete report (report design and data). This can be especially important if you need to keep a snapshot of data taken against a database that changes frequently.

DataWindow Designer creates a PSR file when you save data in the Powersoft report file format. See “Saving data in an external file” on page 168. PSR files are used primarily by InfoMaker, a reporting tool. When an InfoMaker user opens a PSR file, InfoMaker displays the report in the Report painter. If InfoMaker is not already running, opening a PSR file automatically starts InfoMaker.

Modifying general DataWindow object properties

This section describes the general DataWindow object properties that you can modify.

Changing the DataWindow object style

The general style properties for a DataWindow object include:

- The unit of measure used in the DataWindow object
- A timer interval for events in the DataWindow object
- A background color for the DataWindow object

DataWindow Designer assigns defaults when it generates the basic DataWindow object. You can change the defaults.

❖ **To change the default style properties:**

- 1 Position the pointer in the background of the DataWindow object, display the pop-up menu, and select Properties.

The Properties view displays with the General page on top.

- 2 Click the unit of measure you want to use to specify distances when working with the DataWindow object:

- PowerBuilder units (PBUs)
- Pixels (smallest element on the display monitor)
- Thousandths of an inch
- Thousandths of a centimeter

Choosing the unit of measure

If you plan to print the contents of the DataWindow object at runtime, change the unit of measure to inches or centimeters to make it easier to specify the margin measurements.

- 3 Specify the number of milliseconds you want between internal timer events in the DataWindow object.

This value determines how often DataWindow Designer updates the time fields in the DataWindow object. (Enter 60,000 milliseconds to specify one minute.)

- 4 Select a background color from the Color drop-down list and click OK. The default color is the window background color.

Setting colors in a DataWindow object

You can set different colors for each element of a DataWindow object to enhance the display of information.

❖ **To set the background color in a DataWindow object:**

- 1 Position the mouse on an empty spot in the DataWindow object, display the pop-up menu, and select Properties.
- 2 On the General page in the Properties view for the DataWindow object, select a color from the Color drop-down list.

❖ **To set the color of a band in a DataWindow object:**

- 1 Position the mouse pointer on the bar that represents the band, display the pop-up menu, then select Properties.
- 2 On the General page in the band's Properties view, select a color from the Color drop-down list.

The choice you make here overrides the background color for the DataWindow object.

❖ **To set colors in controls in a DataWindow object:**

- Position the mouse pointer on the control, display the pop-up menu, then select Properties.

For controls that use text, you can set colors for background and text on the Font page in the Properties view. For drawing controls, you can set colors on the General page in the Properties view.

Specifying properties of a grid DataWindow object

In grid DataWindow objects, you can specify:

- When grid lines are displayed
- How users can interact with the DataWindow object at runtime

❖ **To specify basic grid DataWindow object properties:**

- 1 Position the mouse pointer on the background in a grid DataWindow object, display the pop-up menu, and select Properties.
- 2 Select the options you want in the Grid section on the General page in the Properties view as described in Table 5-3.

Table 5-3: Options for grid DataWindow objects

Option	Result
On	Grid lines always display
Off	Grid lines never display (users cannot resize columns at runtime)
Display Only	Grid lines display only when the DataWindow object displays online
Print Only	Grid lines display only when the contents of the DataWindow object are printed
Column Moving	Columns can be moved at runtime

Option	Result
Mouse Selection	Data can be selected at runtime (and, for example, copied to the clipboard)
Row Resize	Rows can be resized at runtime

Specifying pointers for a DataWindow object

Just as with colors, you can specify different pointers to use when the mouse is over a particular area of the DataWindow object. For example, you might want to change the pointer when the mouse is over a column whose data cannot be changed.

❖ **To change the mouse pointer used at runtime:**

- 1 Position the mouse over the element of the DataWindow object whose pointer you want to define, display the pop-up menu, and select Properties to display the appropriate Properties view.

You can set a pointer for the entire DataWindow object, specific bands, and specific controls.

- 2 Select the Pointer tab.
- 3 Either choose the pointer from the Stock Pointers list or, if you have a file containing pointer definitions (CUR files), enter a pointer file name.

You can use the Browse button to search for the file.

- 4 Click OK.

Defining print specifications for a DataWindow object

When you are satisfied with the look of the DataWindow object, you can define its print specifications.

❖ **To define print specifications for a DataWindow object:**

- 1 In the DataWindow painter, select Properties from the DataWindow object's pop-up menu to display the DataWindow object's Properties view.
- 2 In the Units box on the General page, select a unit of measure.

It is easier to specify the margins when the unit of measure is inches or centimeters.

- 3 Select the Print Specifications tab.
The Print Specifications page uses the units of measure you specified on the General page.
- 4 Specify print specifications for the current DataWindow object.
See Table 5-4 for more information.

Table 5-4: Setting print specifications for DataWindow objects

Setting	Description
Document Name	Specify a name to be used in the print queue to identify the report.
Printer Name	Specify the name of a printer to which this report should be sent. If this box is empty, the report is sent to the default system printer. If the specified printer cannot be found, the report is sent to the default system printer if the Can Use Default Printer check box is selected. If the specified printer cannot be found and the Can Use Default Printer check box is not selected, an error is returned.
Margins	Specify top, bottom, left, and right margins. You can also change margins in the Preview view while you are actually looking at data. If you change margins in the Preview view, the changes are reflected here on the Print Specifications page.
Paper Orientation	Choose one of the following: <ul style="list-style-type: none"> • Default: Uses the default printer setup. • Portrait: Prints the contents of the DataWindow object across the width of the paper. • Landscape: Prints the contents of the DataWindow object across the length of the paper.
Paper Size	Choose a paper size or leave blank to use the default.
Paper Source	Choose a paper source or leave blank to use the default.
Prompt Before Printing	Select to display the standard Print Setup dialog box each time users make a print request.
Can Use Default Printer	Clear this check box if a printer has been specified in the Printer Name box and you do not want the report to be sent to the default system printer if the specified printer cannot be found. This box is checked by default if a printer name is specified.
Display Buttons - Print Preview	Select to display Button controls in Print Preview. The default is to hide them.

Setting	Description
Display Buttons - Print	Select to display Button controls when you print the report. The default is to hide them.
Clip Text	Select to clip static text to the dimensions of a text field when the text field has no visible border setting. The text is always clipped if the text field has visible borders.
Override Print Job	When you print a series of reports using the PrintOpen, PrintDataWindow, and PrintClose methods, all the reports in the print job use the layout, fonts, margins, and other print specifications defined for the computer. Select this check box to override the default print job settings and use the print settings defined for this report.
Collate Copies	Select to collate copies when printing. Collating increases print time because the print operation is repeated to produce collated sets.
Print Preview Shows Outline	Select to display a blue line that represents the margins in Print Preview.
Newspaper Columns Across and Width	If you want a multiple-column report where the data fills one column on a page, then the second, and so on, as in a newspaper, select the number and width of the columns in the Newspaper Columns boxes. See "Printing with newspaper-style columns" next.

Printing with newspaper-style columns

When you define a DataWindow object, you can specify that it print in multiple columns across the page, like a newspaper. A typical use of newspaper-style columns is a phone list, where you want to have more than one column of names on a printed page.

Use Print Preview to see the printed output

Newspaper-style columns are used only when the DataWindow object is printed. They do not appear when a DataWindow object executes (or in Preview). Therefore, to see them in DataWindow Designer, use Print Preview in the DataWindow painter.

❖ To define newspaper-style columns for a DataWindow object:

- 1 Build a tabular DataWindow object with the data you want.
- 2 Select Properties from the DataWindow object's pop-up menu.
- 3 Select the Print Specifications tab.

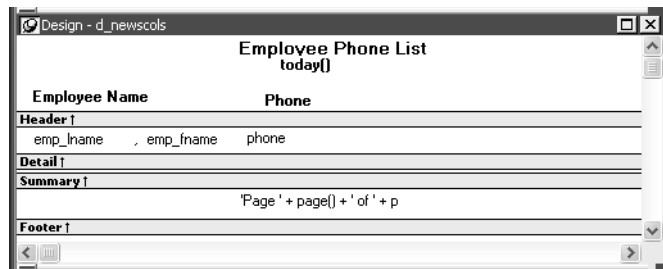
- 4 Specify the number of columns across the page and the width of columns in the Newspaper Columns Across and Newspaper Columns Width properties.
- 5 For each control in the DataWindow object that you do *not* want to have appear multiple times on the page (such as headers), select Properties from the control's pop-up menu and select the HideSnaked check box on the General page in the Properties view.

Example

This example describes how to create a newspaper-style DataWindow object using the Employee table in the EAS Demo DB.

- 1 Create a tabular DataWindow object, selecting the last name, first name, and phone number columns, and add a title, page number, and date.

The Emp_Fname column and the text control holding a comma are defined as Slide Left, so they display just to the right of the Emp_Lname column.



- 2 On the Print Specifications page of the DataWindow object's Properties view, specify two columns across and a column width of 3.5 inches in the Newspaper Columns boxes. (Make sure that Units is set to inches on the General property page.)
- 3 To view the DataWindow object as it will be printed, place the pointer in the Preview view and select File>PrintPreview.

The DataWindow object displays the result set in two columns. Everything above the column headers (which includes page number, title, and date) also shows twice because of the 2-column specification. This information should appear only once per page.

- 4 To specify that page number, title, and date appear only once on the page, you need to suppress printing after the first column. For each of these controls, select Properties from the control's pop-up menu. Then select the HideSnaked check box.

The finished DataWindow object has one set of page heading information and two columns of column header and detail information.

<i>(** means out of state)</i>		Employee Phone List	
		08/20/04	
Employee Name	Phone	Employee Name	Phone
Ahmed, Alex	(617) 555-8748	Pickett, Catherine	(617) 555-3478
Barker, Joseph	(617) 555-8021	Poitras, Kathleen	(617) 555-3920
Barletta, Irene	(617) 555-8345	Powell, Thomas	(617) 555-1956
Bertrand, Jeannette	(508) 555-8138	Preston, Mark	(617) 555-5862
Bigelow, Janet	(617) 555-1493	Rabkin, Andrew	(617) 555-4444
Blaikie, Barbara	(617) 555-9345	Rebeiro, Anthony	(617) 555-5737
Braun, Jane	(617) 555-7857	Romero, Sheila	(617) 555-8138
Breault, Robert	(617) 555-3099	Samuels, Peter	(617) 555-8342
Bucceri, Matthew	(617) 555-5336	** Savarino, Pamela	(310) 555-1857
Butterfield, Joyce	(617) 555-2232	Scott, David	(617) 555-3246
Chao, Shih Lin	(617) 555-5921	Shea, Mary Anne	(617) 555-4616
Charlton, Doug	(508) 555-9246	** Sheffield, John	(713) 555-3877
** Chin, Philip	(404) 555-2341	Shishov, Natasha	(617) 555-2755

Modifying text in a DataWindow object

When DataWindow Designer initially generates the basic DataWindow object, it uses the following attributes and fonts:

- For the text and alignment of column headings and labels, DataWindow Designer uses the extended column attributes made in the Database painter.
- For fonts, DataWindow Designer uses the definitions made in the Database painter for the table.

You can override any of these defaults in a particular DataWindow object.

❖ To change text in a DataWindow object:

- 1 Select the text.

The first box in the StyleBar is now active.

- 2 Type the new text.

Use `-n~r` to embed a newline character in the text.

- ❖ **To change the text properties for a text control in a DataWindow object:**
 - 1 Select the text control.
 - 2 Do one of the following:
 - Change the text properties in the StyleBar.
 - Select the Font page in the control's Properties view and change the properties there.

Defining the tab order in a DataWindow object

When DataWindow Designer generates the basic DataWindow object, it assigns columns a default tab order, the default sequence in which focus moves from column to column when a user presses the Tab key at runtime. DataWindow Designer assigns tab values in increments of 10 in left-to-right and top-to-bottom order.

Tab order is not used in the Design view

Tab order is used when a DataWindow object is executed, but it is not used in the DataWindow painter Design view. In the Design view, the Tab key moves to the controls in the DataWindow object in the order in which the controls were placed in the Design view.

If the DataWindow object contains columns from more than one table

If you are defining a DataWindow object with more than one table, DataWindow Designer assigns each column a tab value of 0, meaning the user cannot tab to the column. This is because, by default, multitable DataWindow objects are not updatable—users cannot modify data in them. You can change the tab values to nonzero values to allow tabbing in these DataWindow objects.

For more about controlling updates in a DataWindow object, see Chapter 6, “Controlling Updates in DataWindow Objects.”

Tab order changes have no effect in grid DataWindow objects

In a grid DataWindow object, the tab sequence is always left to right (except on right-to-left operating systems). Changing the tab value to any number other than 0 has no effect.

❖ **To change the tab order:**

- 1 Select Format>Tab Order from the menu bar or click the Tab Order button on PainterBar2.

The current tab order displays.

- 2 Use the mouse or the Tab key to move the pointer to the tab value you want to change.

- 3 Enter a new tab value in the range 0 to 9999.

0 removes the column from the tab order (the user cannot tab to the column). It does not matter exactly what value you use (other than 0); all that matters is relative value. For example, if you want the user to tab to column B after column A but before column C, set the tab value for column B so it is between the value for column A and the value for column C.

- 4 Repeat the procedure until you have the tab order you want.

- 5 Select Format>Tab Order from the menu bar or click the Tab Order button again.

DataWindow Designer saves the tab order.

Each time you select Tab Order, DataWindow Designer reassigns tab values to include any columns that have been added to the DataWindow object and to allow space to insert new columns in the tab order.

Changing tab order at runtime

To change tab order programmatically at runtime, use the TabOrder property.

Naming controls in a DataWindow object

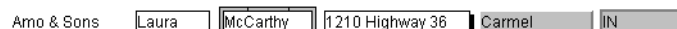
You use names to identify columns and other controls in validation rules, filters, and DataWindow expression functions.

The DataWindow painter automatically generates names for all controls in a DataWindow object. To name columns, labels, and headings, the DataWindow painter uses database and extended attribute information. To name all other controls, it uses a system of prefixes. You can control the prefixes used for automatic name generation and you can specify the name of any control explicitly.

- ❖ **To specify prefixes for naming controls systematically in a DataWindow object:**
 - 1 Select Design>Options from the menu bar and then select the Prefixes tab.
 - 2 Change prefixes as desired and click OK.
- ❖ **To specify a name of a control in a DataWindow object:**
 - 1 Select Properties from the control's pop-up menu and then select the General tab in the Properties view.
 - 2 Type the name in the Name box.

Using borders in a DataWindow object

You can place borders around text, columns, graphs, and crosstabs to enhance their appearance. DataWindow Designer provides six types of borders: Underline, Box, ResizeBorder, ShadowBox, Raised, and Lowered:



Arno & Sons Laura McCarthy 1210 Highway 36 Carmel IN

Border appearance varies

Changing the border style may not have the same effect on all Windows platforms.

- ❖ **To add a border to a control in a DataWindow object:**
 - 1 Select one or more controls.
 - 2 Select the border you want from the Border drop-down toolbar in the PainterBar.

DataWindow Designer places the border around the selected controls.

You can also specify a border for one or more controls in the Properties view on the General page.

Specifying variable-height bands in a DataWindow object

Sometimes DataWindow objects contain columns whose data is of variable length. For example, a Memo column in a table might be a character column that can take up to several thousand characters. Reserving space for that much information for the column in the detail band would make the detail band's height very large, meaning users could see few rows at a time.

The detail band can resize based on the data in the Memo column. If the Memo column has only one line of text, the detail band should be one line. If the Memo column has 20 lines of text, the detail band should be 20 lines high.

To provide a band that resizes as needed, specify that the variable-length columns and the band have Autosize Height. All bands in the DataWindow can be resized, but nested report overflow is supported only in the Detail band. If autosizing would preclude the display of at least one Detail band row per page, other bands cannot be autosized. Autosizing is not supported with the Graph or Label presentation styles.

❖ To create a resizable band in a DataWindow object:

- 1 Select Properties from the pop-up menu of a column that should resize based on the amount of data.
- 2 Select the Autosize Height check box on the Position page.
- 3 Clear the Auto Horz Scroll check box on the Edit page.

DataWindow Designer wraps text in the Preview view instead of displaying text on one scrollable line.

- 4 Repeat steps 1 to 3 for any other columns that should resize.
- 5 Select Properties from the band's pop-up menu.
- 6 Select the Autosize Height check box on the General page.

In the Preview view, the band resizes based on the contents of the columns you defined as having Autosize Height.

Using the RowHeight function with Autosize Height

When a detail band has Autosize Height set to true, you should avoid using the RowHeight DataWindow expression function to set the height of any element in the row. Doing so can result in a logical inconsistency between the height of the row and the height of the element. If you need to use RowHeight, you must set the Y coordinate of the element to 0 on the Position page in the Properties view, otherwise the bottom of the element might be clipped. You must do this for every element that uses such an expression. If you move any elements in the band, make sure that their Y coordinates are still set to 0.

You should not use an expression whose runtime value is greater than the value returned by `RowHeight`. For example, you should not set the height of a column to `rowheight() + 30`. Such an expression produces unpredictable results at runtime.

Clipping columns

You can have Autosize Height columns without an Autosize Height detail band. If such a column expands beyond the size of the detail band in the Preview view, it is clipped.

Modifying the data source of a DataWindow object

When modifying a DataWindow object, you might realize that you have not included all the columns you need, or you might need to define retrieval arguments. You can modify the data source from the DataWindow painter. How you do it depends on the data source.

Modifying SQL SELECT statements

If the data source is SQL (such as Quick Select, SQL Select, or Query), you can graphically modify the SQL SELECT statement.

❖ **To modify a SQL data source:**

- 1 Select Design>Data Source from the menu bar.

DataWindow Designer returns you to the Select painter. (If you used Quick Select to define the data source, this might be the first time you have seen the Select painter.)

- 2 Modify the SELECT statement graphically using the same techniques as when creating it.

For more information, see “Using SQL Select” on page 121.

Modifying the statement syntactically

Select Design>Convert to Syntax from the menu bar to modify the SELECT statement syntactically.

- 3 Click the Return button to return to the painter.

Some changes you make (such as adding or removing columns) require DataWindow Designer to modify the update capabilities of the DataWindow object.

For more information about controlling updates in a DataWindow object, see Chapter 6, “Controlling Updates in DataWindow Objects.”

Changing the table

If you change the table referenced in the SELECT statement, DataWindow Designer maintains the columns in the Design view (now from a different table) only if they match the data types and order of the columns in the original table.

Modifying the retrieval arguments

You can add, modify, or delete retrieval arguments when modifying your data source.

❖ To modify the retrieval arguments:

- 1 In the Select painter, select Design>Retrieval Arguments from the menu bar.

The Specify Retrieval Arguments dialog box displays, listing the existing arguments.

- 2 Add, modify, or delete the arguments.
- 3 Click OK.

You return to the Select painter, or to the text window displaying the SELECT statement if you are modifying the SQL syntactically.

- 4 Reference any new arguments in the WHERE or HAVING clause of the SELECT statement.

For more information about retrieval arguments, see Chapter 4, “Defining DataWindow Objects.”

Modifying the result set

If the data source is External or Stored Procedure, you can modify the result set description.

❖ To modify a result set:

- 1 If the Column Specification view is not open, select View>Column Specifications from the menu bar.
- 2 Review the specifications and make any necessary changes.

If the data source is a stored procedure

If you are modifying the result set for a DataWindow object whose data source is a stored procedure, the pop-up menu for the Column Specification view contains the menu item Stored Procedure.

Select Stored Procedure from the Column Specification view's pop-up menu to edit the Execute statement, select another stored procedure, or add retrieval arguments. For more information about editing the Execute statement, see "Using Stored Procedure" on page 137.

Storing data in a DataWindow object using the Data view

Usually you retrieve data into a DataWindow object from the database, because the data is changeable and you want the latest information. However, sometimes the data you display in a DataWindow object never changes (as in a list of states or provinces), and sometimes you need a snapshot of the data at a certain point in time. In these situations, you can store the data in the DataWindow object itself. You do not need to go out to the database or other data source to display the data.

The most common reason to store data in a DataWindow object is for use as a drop-down DataWindow where the data is not coming from a database. For example, you might want to display a list of postal codes for entering values in a State or Province column in a DataWindow object. You can store those codes in a DataWindow object and use the DropDownDataWindow edit style for the column.

For more information about using the DropDownDataWindow edit style, see Chapter 8, "Displaying and Validating Data."

❖ **To store data in a DataWindow object:**

- 1 If the Data view is not already displayed, select View>Data from the menu bar.

In the default layout for the DataWindow painter, the Data view displays in a stacked pane under the Properties view. All columns defined for the DataWindow object are listed at the top.

- 2 Do any of the following:
 - Click the Insert Row button in the PainterBar to create an empty row and type a row of data. You can enter as many rows as you want.
 - Click the Retrieve button in the PainterBar to retrieve all the rows of data from the database. You can delete rows you do not want to save or manually add new rows.
 - Click the Delete button in the PainterBar to delete unwanted rows.

Data changes are local to the DataWindow object

Adding or deleting data here does not change the data in the database. It only determines what data will be stored with the DataWindow object when you save it. The Update DB button is disabled.

- 3 When you have finished, save the DataWindow object.

When you save the DataWindow object, the data is stored in the DataWindow object.

Sharing data with the Preview view

To see changes you make in the Data view reflected in the Preview view, select ShareData from the pop-up menu in the Data view. The Preview view shows data from the storage buffer associated with the Data view.

Saving the DataWindow object without data

If you saved the DataWindow object with data to obtain a snapshot, you usually need to save it again without data. To do so, select Delete All Rows from the pop-up menu in the Data view before saving.

Sharing DataWindow objects with other developers

Storing data in a DataWindow object is a good way to share data *and its definition* with other developers. They can simply open the DataWindow object on their computers to get the data and all its properties.

What happens at runtime

Data stored in a DataWindow object is stored within the actual object itself, so when a window opens showing such a DataWindow, the data is already there. There is no need to issue Retrieve to get the data.

DataWindow .NET *never* retrieves data into a drop-down DataWindow that already contains data. For all other DataWindow objects, if you retrieve data into a DataWindow object stored with data, DataWindow .NET handles it the same as a DataWindow object that is not stored with data: DataWindow .NET gets the latest data by retrieving rows from the database.

Retrieving data

In a DataWindow object, you can prompt for retrieval criteria, retrieve rows as needed, and save retrieved rows to disk.

Prompting for retrieval criteria in a DataWindow object

You can define your DataWindow object so that it always prompts for retrieval criteria just before it retrieves data.

❖ To prompt for retrieval criteria in a DataWindow object:

- 1 If the Column Specifications view is not already displayed, select View>Column Specifications from the menu bar.

In the default layout for the DataWindow painter, the Column Specifications view displays in a stacked pane under the Properties view. All columns defined for the DataWindow object are listed in the view.

- 2 Select the Prompt check box next to each column for which you want to specify retrieval criteria at runtime.

When you specify prompting for criteria, DataWindow Designer displays the Specify Retrieval dialog box just before a retrieval is to be done.

Each column you selected in the Column Specification view displays in the grid. Users can specify criteria here exactly as in the grid in the Quick Select dialog box. Criteria specified here are added to the WHERE clause for the SQL SELECT statement defined for the DataWindow object.

Testing in DataWindow Designer

You can test the prompting for criteria by retrieving data in the Preview view of the DataWindow object.

Using edit styles

If a column uses a code table or the RadioButton, CheckBox, or DropDownList edit style, an arrow displays in the column header and users can select a value from a drop-down list when specifying criteria:



If you do not want the drop-down list used for a column for specifying retrieval criteria, select the Override Edit check box on the General page of the column's Properties view.

Forcing the entry of criteria

If you have specified prompting for criteria for a column, you can force the entry of criteria for the column by selecting the Equality Required check box on the General page of the column's Properties view. DataWindow Designer underlines the column header in the grid during prompting. Selection criteria for the specified column must be entered, and the = operator must be used.

For more information

The section “Using Quick Select” on page 112 describes in detail how you and your users can specify selection criteria in the grid.

The chapter on dynamic DataWindow objects in the *Programmer's Guide* describes how to write code to allow users to specify retrieval criteria at runtime.

Retrieving rows as needed

If a DataWindow object retrieves hundreds of rows, there can be a noticeable delay at runtime while all the rows are retrieved and before control returns to the user. For these DataWindow objects, DataWindow Designer can retrieve only as many rows as it has to before displaying data and returning control to the user.

For example, if a DataWindow object displays only 10 rows at a time, DataWindow Designer only needs to retrieve 10 or more rows before presenting the data. Then, as the user pages through the data, DataWindow Designer continues to retrieve what is necessary to display the new information. There may be slight pauses while DataWindow Designer retrieves the additional rows, but these pauses are usually preferable to waiting a long time to start working with data.

❖ **To specify that a DataWindow object retrieve only as many rows as it needs to:**

- Select Rows>Retrieve Options>Rows As Needed from the menu bar.

With this setting, DataWindow Designer presents data and returns control to the user when it has retrieved enough rows to display in the DataWindow object.

Retrieve Rows As Needed is overridden if you have specified sorting or have used aggregate functions, such as Avg and Sum, in the DataWindow object. This is because DataWindow Designer must retrieve every row before it can sort or perform aggregates.

In a multiuser situation, Retrieve Rows As Needed might lock other people out of the tables.

Saving retrieved rows to disk

If you want to maximize the amount of memory available to DataWindow Designer and other running applications, DataWindow Designer can save retrieved data on your hard disk in a temporary file rather than keep the data in memory. DataWindow Designer swaps rows of data from the temporary file into memory as needed to display data.

❖ **To maximize available memory by saving retrieved rows to disk:**

- Select Rows>Retrieve Options>Rows to Disk from the menu bar.

With this setting, when displaying data, DataWindow Designer swaps rows of data from the temporary file into memory instead of keeping all the retrieved rows of data in memory.

Controlling Updates in DataWindow Objects

About this chapter

When DataWindow Designer generates the basic DataWindow object, it defines whether the data is updatable. This chapter describes the default settings and how you can modify them.

Contents

Topic	Page
About controlling updates	191
Specifying the table to update	193
Specifying the unique key columns	193
Specifying an identity column	194
Specifying updatable columns	194
Specifying the WHERE clause for update/delete	195
Specifying update when key is modified	197
Using stored procedures to update the database	198

About controlling updates

When DataWindow Designer generates the basic DataWindow object, it defines whether the data is updatable by default as follows:

- If the DataWindow object contains columns from a single table and includes that table's key columns, DataWindow Designer defines all columns as updatable and specifies a nonzero tab order for each column, allowing users to tab to the columns.
- If the DataWindow object contains columns from two or more tables or from a view, DataWindow Designer defines all columns as not being updatable and sets all tab orders to zero, preventing users from tabbing to them.

You can accept the default settings or modify the update characteristics for a DataWindow object.

If using a Stored Procedure or External data source

If the data source is Stored Procedure or External, you can use the FindNextModiedRow method to write your own update code.

What you can do

You can:

- Allow updates in a DataWindow object associated with multiple tables or a view; you can define one of the tables as being updatable
- Prevent updates in a DataWindow object associated with one table
- Prevent updates to specific columns in a DataWindow object that is associated with an updatable table
- Specify which columns uniquely identify a row to be updated
- Specify which columns will be included in the WHERE clause of the UPDATE or DELETE statement DataWindow Designer generates to update the database
- Specify whether DataWindow Designer generates an UPDATE statement, or a DELETE then an INSERT statement, to update the database when users modify the values in a key column

Updatability of views

Some views are logically updatable; some are not. For the rules your DBMS follows for updating views, see your DBMS documentation.

❖ **To specify update characteristics for a DataWindow object:**

- 1 Select Rows>Update Properties from the menu bar.

The Specify Update Properties dialog box displays.

- 2 To prevent updates to the data, make sure the Allow Updates box is not selected.

To allow updates, select the Allow Updates box and specify the other settings as described below.

- 3 Click OK.

Changing tab values

DataWindow Designer does not change the tab values associated with columns after you change the update characteristics of the DataWindow object. If you have allowed updates to a table in a multitable DataWindow object, you should change the tab values for the updatable columns so that users can tab to them.

For more information, see “Defining the tab order in a DataWindow object” on page 179.

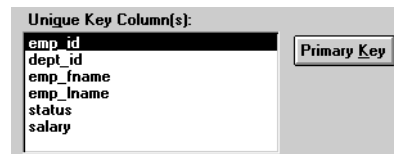
Specifying the table to update

Each DataWindow object can update one table, which you select from the Table to Update box in the Specify Update Properties dialog box.



Specifying the unique key columns

The Unique Key Columns box in the Specify Update Properties dialog box specifies which columns DataWindow Designer uses to identify a row being updated. DataWindow Designer uses the column or columns you specify here as the key columns when generating the WHERE clause to update the database (as described below):



The key columns you select here must uniquely identify a row in the table. They can be the table's primary key, though they don't have to be.

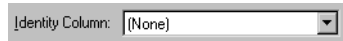
Using the primary key

Clicking the Primary Key button cancels any changes in the Unique Key Columns box and highlights the primary key for the updatable table.

Specifying an identity column

Many DBMSs allow you to specify that the value for a column in a new row is to be automatically assigned by the DBMS. This kind of column is called an identity column. Different DBMSs provide different types of identity columns.

For example, some DBMSs allow you to define autoincrement columns so that the column for a new row is automatically assigned a value one greater than that of the previous highest value. You could use this feature to specify that an order number be automatically incremented when someone adds a new order:



By specifying an identity column in the Specify Update Properties dialog box, you tell DataWindow Designer to bring back the value of a new row's identity column after an insert in the DataWindow object so that users can see it.

For information about identity columns in your DBMS, see your DBMS documentation.

Specifying updatable columns

You can make all or some of the columns in a table updatable.

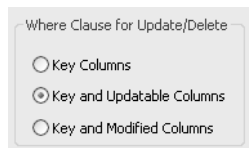
Updatable columns are displayed highlighted. Click a nonupdatable column to make it updatable. Click an updatable column to make it nonupdatable.

Changing tab values

If you have changed the updatability of a column, you should change its tab value. If you have allowed a column to be updated, you should change its tab value to a nonzero number so users can tab to it.

Specifying the WHERE clause for update/delete

Sometimes multiple users are accessing the same tables at the same time. In these situations, you need to decide when to allow your application to update the database. If you allow your application to always update the database, it could overwrite changes made by other users:



You can control when updates succeed by specifying which columns DataWindow Designer includes in the WHERE clause in the UPDATE or DELETE statement used to update the database:

```
UPDATE table...
SET column = newvalue
WHERE col1 = value1
AND col2 = value2 ...
```

```
DELETE
FROM table
WHERE col1 = value1
AND col2 = value2 ...
```

Using timestamps

Some DBMSs maintain timestamps so you can ensure that users are working with the most current data. If the SELECT statement for the DataWindow object contains a timestamp column, DataWindow Designer includes the key column and the timestamp column in the WHERE clause for an UPDATE or DELETE statement regardless of which columns you specify in the Where Clause for Update/Delete box.

If the value in the timestamp column changes (possibly due to another user modifying the row), the update fails.

To see whether you can use timestamps with your DBMS, see *Connecting to Your Database*.

Choose one of the options in Table 6-1 in the Where Clause for Update/Delete box. The results are illustrated by an example following the table.

Table 6-1: Specifying the WHERE clause for UPDATE and DELETE

Option	Result
Key Columns	<p>The WHERE clause includes the key columns only. These are the columns you specified in the Unique Key Columns box.</p> <p>The values in the originally retrieved key columns for the row are compared against the key columns in the database. No other comparisons are done. If the key values match, the update succeeds.</p> <hr/> <p>Caution Be very careful when using this option. If you tell DataWindow Designer only to include the key columns in the WHERE clause and someone else modified the same row after you retrieved it, their changes will be overwritten when you update the database (see the example following this table).</p> <hr/> <p>Use this option only with a single-user database or if you are using database locking. In other situations, choose one of the other two options described in this table.</p>
Key and Updatable Columns	<p>The WHERE clause includes all key and updatable columns.</p> <p>The values in the originally retrieved key columns and the originally retrieved updatable columns are compared against the values in the database. If any of the columns have changed in the database since the row was retrieved, the update fails.</p>
Key and Modified Columns	<p>The WHERE clause includes all key and modified columns.</p> <p>The values in the originally retrieved key columns and the modified columns are compared against the values in the database. If any of the columns have changed in the database since the row was retrieved, the update fails.</p>

Example

Consider this situation: a DataWindow object is updating the Employee table, whose key is Emp_ID; all columns in the table are updatable. Suppose the user has changed the salary of employee 1001 from \$50,000 to \$65,000. This is what happens with the different settings for the WHERE clause columns:

- If you choose Key Columns for the WHERE clause, the UPDATE statement looks like this:

```
UPDATE Employee
SET Salary = 65000
WHERE Emp_ID = 1001
```

This statement will succeed *regardless of whether other users have modified the row since your application retrieved the row*. For example, if another user had modified the salary to \$70,000, that change will be overwritten when your application updates the database.

- If you choose Key and Modified Columns for the WHERE clause, the UPDATE statement looks like this:

```
UPDATE Employee
SET Salary = 65000
WHERE Emp_ID = 1001
  AND Salary = 50000
```

Here the UPDATE statement is also checking the original value of the modified column in the WHERE clause. The statement will fail if another user changed the salary of employee 1001 since your application retrieved the row.

- If you choose Key and Updatable Columns for the WHERE clause, the UPDATE statement looks like this:

```
UPDATE Employee
SET Salary = 65000
WHERE Emp_ID = 1001
  AND Salary = 50000
  AND Emp_Fname = original_value
  AND Emp_Lname = original_value
  AND Status = original_value
  ...
```

Here the UPDATE statement is checking all updatable columns in the WHERE clause. This statement will fail if any of the updatable columns for employee 1001 have been changed since your application retrieved the row.

Specifying update when key is modified

The Key Modification property determines the SQL statements DataWindow Designer generates whenever a key column—a column you specified in the Unique Key Columns box—is changed. The options are:

- Use DELETE then INSERT (default)
- Use UPDATE

How to choose a setting

Consider the following when choosing the Key Modification setting:

- If multiple rows are changed, DELETE and INSERT always work. In some DBMSs, UPDATE fails if the user modifies two keys and sets the value in one row to the original value of the other row.
- You might choose the setting here based on your DBMS triggers. For example, if there is an Insert trigger, select Use Delete then Insert.
- If only one row can be modified by the user before the database is updated, use UPDATE because it is faster.

Using stored procedures to update the database

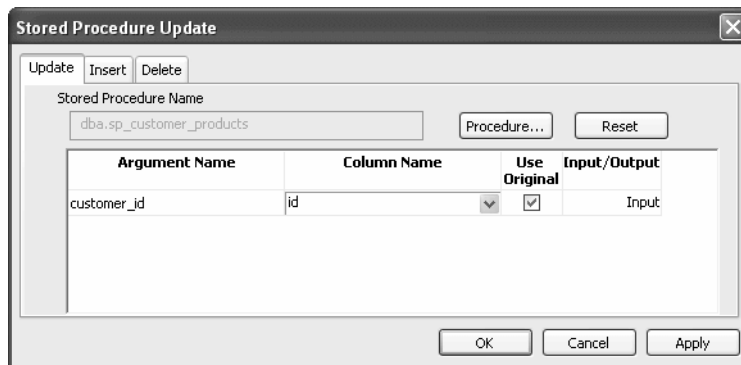
Updates to the database can be performed using stored procedures.

Why use stored procedures?

The DataWindow control submits updates to the database by dynamically generating INSERT, DELETE, and UPDATE SQL statements after determining the status of each row in the DataWindow object. You can also define procedural SQL statements in a stored procedure for use by all applications accessing a database. Using stored procedures to perform database updates allows you to enhance database security, integrity, and performance. Since stored procedures provide for conditional execution, you can also use them to enforce additional business rules.

Updating using stored procedures

The Stored Procedure Update dialog box only allows you to associate an existing stored procedure with your DataWindow object. The stored procedure must have been previously defined in the database.



❖ **To use stored procedures to update the database**

- 1 In the DataWindow painter, select Rows>Stored Procedure Update to display the Stored Procedure Update dialog box.
- 2 Select the tab for the SQL update method (Delete, Insert, or Update) with which you want to associate a stored procedure.
- 3 Click the Procedure button, select the stored procedure you want to have execute when the SQL update method is generated, and click OK.

The parameters used in the stored procedure are displayed in the Argument Name list in the order in which they are defined in the procedure. Column Name lists the columns used in your DataWindow object.

- 4 Associate a column in the DataWindow object or an expression with a procedure parameter.

If a stored procedure uses parameters that are not matched to column names, you can substitute the value from a DataWindow object computed field or expression.

Matching a column to a procedure parameter

You must be careful to correctly match a column in the DataWindow object to a procedure parameter, since DataWindow Designer is able to verify only that datatypes match.

- 5 If the parameter is to receive a column value, indicate whether the parameter will receive the updated column value entered through the DataWindow object or retain the original column value from the database.

Typically, you select Use Original when the parameter is used in a WHERE clause in an UPDATE or DELETE SQL statement. If you do not select Use Original, the parameter will use the new value entered for that column.

Typically, you would use the new value when the parameter is used in an INSERT or UPDATE SQL statement.

What happens when the stored procedure is executed

The stored procedure you associate with a SQL update method in the Stored Procedure Update dialog box is executed when the DataWindow control calls the UpdateData method. The DataWindow control examines the table in the DataWindow object, determines the appropriate SQL statement for each row, and submits the appropriate stored procedure (as defined in the Stored Procedure Update dialog box) with the appropriate column values substituted for the procedure arguments.

If a stored procedure for a particular SQL update method is not defined, the DataWindow control submits the appropriate SQL syntax in the same manner it always has.

Data

Using Describe and Modify

You can use the DataWindow Describe and Modify methods to access DataWindow property values including the stored procedures associated with a DataWindow object. For information, see the DataWindow object property Table.property in the *DataWindow Object Reference*.

Restrictions on the use of Modify

Since a database driver can only report stored procedure names and parameter names and position, it cannot verify that changes made to stored procedures are valid. Consequently, if you use Modify to change a stored procedure, be careful that you do not inadvertently introduce changes into the database.

In addition, using Modify to enable a DataWindow object to use stored procedures to update the database when it is not already using stored procedures requires that the type qualifier be specified first. Calling the type qualifier ensures that internal structures are built before subsequent calls to Modify. If a new method or method arguments are specified without a preceding definition of type, Modify fails.

Working with Controls in DataWindow Objects

About this chapter

One of the ways you can enhance a DataWindow object is to add controls, such as columns, drawing objects, buttons, and computed fields. You can also change the layout of the DataWindow object by reorganizing, positioning, and rotating controls. This chapter shows you how.

Contents

Topic	Page
Adding controls to a DataWindow object	201
Reorganizing controls in a DataWindow object	214
Positioning controls in a DataWindow object	221
Rotating controls in a DataWindow object	222

Adding controls to a DataWindow object

This section describes adding controls to enhance your DataWindow object.

Adding columns to a DataWindow object

You can add columns that are included in the data source to a DataWindow object. When you first create a DataWindow object, each of the columns in the data source is automatically placed in the DataWindow object. Typically, you would add a column to restore one that you had deleted from the DataWindow object, or to display the column more than once in the DataWindow object.

Adding columns not previously retrieved to the data source

To specify that you want to retrieve a column not previously retrieved (that is, add a column to the data source), you must modify the data source.

See “Modifying the data source of a DataWindow object” on page 183.

❖ **To add a column from the data source to a DataWindow object:**

- 1 Select Insert>Control>Column from the menu bar.
- 2 Click where you want to place the column.

The Select Column dialog box displays, listing all columns included in the data source of the DataWindow object.

- 3 Select the column and click OK.

Adding text to a DataWindow object

When DataWindow Designer generates a basic DataWindow object from a presentation style and data source, it places columns and their headings in the workspace. You can add text anywhere you want to make the DataWindow object easier to understand.

❖ **To add text to a DataWindow object:**

- 1 Select Insert>Control>Text from the menu bar.
- 2 Click where you want the text.

DataWindow Designer places the text control in the Design view and displays the word `text`.

- 3 Type the text you want.
- 4 (Optional) Change the font, size, style, and alignment for the text using the StyleBar.

Displaying an ampersand character

If you want to display an ampersand character, type a double ampersand in the Text field. A single ampersand causes the next character to display with an underscore because it is used to indicate accelerator keys.

Adding drawing controls to a DataWindow object

You can add the following drawing controls to a DataWindow object to enhance its appearance:

- Rectangle
- RoundRectangle
- Line
- Oval

❖ To place a drawing control in a DataWindow object:

- 1 Select the drawing control from the Insert>Control menu.
- 2 Click where you want the control to display.
- 3 Resize or move the drawing control as needed.
- 4 Use the drawing control's Properties view to change its properties as needed.

For example, you might want to specify a fill color for a rectangle or thickness for a line.

Adding a group box to a DataWindow object

To visually enhance the layout of a DataWindow object, you can add a group box. A group box is a static frame used to group and label a set of controls in a DataWindow object. The following example shows two group boxes in a report (nonupdatable DataWindow object). The Address group box groups address information and the Phone/Fax group box groups telephone numbers.

Contact Information			<input type="button" value="Filter the list of contacts"/>
Clarke , Molly Sales	Address	Phone/Fax	
	55 Pine Grove Rd. Lexington , MA 02173	(617) 555-4325 (617) 555-7638	
Kaplan , Burt Sales	Address	Phone/Fax	
	49 Keaton Lane Arlington , MA 02174	(617) 555-3687 (617) 555-2398	

❖ **To add a group box to a DataWindow object:**

- 1 Select Insert>Control>Group Box from the menu bar and click in the Design view.
- 2 With the group box selected, type the text to display in the frame.
- 3 Move and resize the group box as appropriate.

Adding pictures to a DataWindow object

You can place pictures, such as your company logo, in a DataWindow object to enhance its appearance. If you place a picture in the header, summary, or footer band of the DataWindow object, the picture displays each time the content of that band displays. If you place the picture in the detail band of the DataWindow object, it displays in each row.

❖ **To place a picture in a DataWindow object:**

- 1 Select Insert>Control>Picture from the menu bar.
- 2 Click where you want the picture to display.
The Select Picture dialog box displays.
- 3 Use the Browse button to find the file or enter a file name in the File Name box. Then click Open.
The picture must be a bitmap (BMP), runlength-encoded (RLE), Windows metafile (WMF), Graphics Interchange Format (GIF), or Joint Photographic Experts Group (JPEG) file.
- 4 Display the pop-up menu and select Original Size to display the bitmap in its original size.
You can use the mouse to change the size of the bitmap in the DataWindow painter.
- 5 Click the Invert Image check box on the General page of the Properties view to display the picture with its colors inverted.

Tips for using pictures

To display a different picture for each row of data, retrieve a column containing picture file names from the database.

For more information, see “Specifying additional properties for character columns” on page 67.

To compute a picture name at runtime, use the `Bitmap` function in the expression defining a computed field. If you change the bitmap in the Picture control in a DataWindow object, you need to reset the original size property. The property automatically reverts to the default setting when you change the bitmap.

To use a picture to indicate that a row has focus at runtime, use the `SetRowFocusIndicator` function.

Adding computed fields to a DataWindow object

You can use computed fields in any band of the DataWindow object. Typical uses with examples include:

- Calculations based on column data that change for each retrieved row

If you retrieve yearly salary, you can define a computed field in the detail band that displays monthly salary: `Salary / 12`.

- Summary statistics of the data

In a grouped DataWindow object, you can use a computed field to calculate the totals of a column, such as salary, for each group: `sum (salary for group 1)`.

- Concatenated fields

If you retrieve first name and last name, you can define a computed field that concatenates the values so they appear with only one space between them: `Fname + " " + Lname`.

- System information

You can place the current date and time in a DataWindow object's header using the built-in functions `Today()` and `Now()` in computed fields.

Computed columns versus computed fields

When creating a DataWindow object, you can define computed columns and computed fields as follows:

- In the Select painter, you can define computed columns when you are defining the `SELECT` statement that will be used to retrieve data into the DataWindow object.

- In the DataWindow painter, you can define computed fields after you have defined the SELECT statement (or other data source).

The difference between the two ways

When you define the computed column in the Select painter, the value is calculated by the DBMS when the data is retrieved. The computed column's value does not change until data has been updated and retrieved again.

When you define the computed field in the DataWindow painter, the value of the column is calculated in the DataWindow object after the data has been retrieved. The value changes dynamically as the data in the DataWindow object changes.

Example

Consider a DataWindow object with four columns: Part number, Quantity, Price, and Cost. Cost is computed as Quantity * Price.

Part #	Quantity	Price	Cost
101	100	1.25	125.00

If Cost is defined as a computed column in the Select painter, the SELECT statement is as follows:

```
SELECT part.part_num,  
       part.part_qty,  
       part.part_price,  
       part.part_qty * part.part_price  
FROM part;
```

If the user changes the price of a part in the DataWindow object in this scenario, the cost does not change in the DataWindow object until the database is updated and the data is retrieved again. The user sees a display with the changed price but the unchanged, incorrect cost.

Part #	Quantity	Price	Cost
101	100	2.50	125.00

If Cost is defined as a computed field in the DataWindow object, the SELECT statement is as follows, with no computed column:

```
SELECT part.part_num,  
       part.part_qty,  
       part.part_price  
FROM part;
```

The computed field is defined in the DataWindow object as Quantity * Price.

In this scenario, if the user changes the price of a part in the DataWindow object, the cost changes immediately.

Part #	Quantity	Price	Cost
101	100	2.50	250.00

Recommendation

If you want your DBMS to do the calculations on the server before bringing data down and you do not need the computed values to be updated dynamically, define the computed column as part of the `SELECT` statement.

If you need computed values to change dynamically, define computed fields in the DataWindow painter Design view, as described next.

Defining a computed field in the DataWindow painter Design view

❖ To define a computed field in the DataWindow painter Design view:

- 1 Select `Insert>Control>Computed Field` from the menu bar.
- 2 Click where you want to place the computed field.

If the calculation is to be based on column data that changes for each row, make sure you place the computed field in the detail band.

The Modify Expression dialog box displays, listing:

- DataWindow expression functions you can use in the computed field
- The columns in the DataWindow object
- Operators and parentheses

- 3 Enter the expression that defines the computed field as described in "Entering the expression" next.
- 4 (Optional) Click `Verify` to test the expression.
DataWindow Designer analyzes the expression.
- 5 Click `OK`.

Entering the expression

You can enter any valid DataWindow expression when defining a computed field. You can paste operators, columns, and DataWindow expression functions into the expression from information in the Modify Expression dialog box. Use the `+` operator to concatenate strings.

DataWindow expressions

You are entering a DataWindow expression, not a SQL expression processed by the DBMS, so the expression follows the rules for DataWindow expressions.

Referring to next and previous rows

You can refer to other rows in a computed field. This is particularly useful in n-up DataWindow objects when you want to refer to another row in the detail band. Use this syntax:

ColumnName[x]

where *x* is an integer. 0 refers to the current row (or first row in the detail band), 1 refers to the next row, -1 refers to the previous row, and so on.

Examples

Table 7-1 shows some examples of computed fields.

Table 7-1: Computed field examples

To display	Enter this expression	In this band
Current date at top of each page	Today ()	Header
Current time at top of each page	Now ()	Header
Current page at bottom of each page	Page ()	Footer
Total page count at bottom of each page	PageCount ()	Footer
Concatenation of Fname and Lname columns for each row	Fname + " " + Lname	Detail
Monthly salary if Salary column contains annual salary	Salary / 12	Detail
Four asterisks if the value of the Salary column is greater than \$50,000	IF (Salary > 50000, "****", "")	Detail
Average salary of all retrieved rows	Avg (Salary)	Summary
Count of retrieved rows, assuming each row contains a value for EmpID	Count (EmpID)	Summary

For complete information about the functions you can use in computed fields in the DataWindow painter, see the *DataWindow Object Reference* in the online Help.

Menu options and PainterBar buttons for common functions

DataWindow Designer provides a quick way to create computed fields that summarize values in the detail band, display the current date, or show the current page number.

❖ **To summarize values:**

- 1 Select one or more columns in the DataWindow object's detail band.
- 2 Select Insert>Control from the menu bar.
- 3 Select one of the options at the bottom of the cascading menu: Average, Count, or Sum.

The same options are available at the bottom of the Controls drop-down toolbar on the PainterBar.

DataWindow Designer places a computed field in the summary band or in the group trailer band if the DataWindow object is grouped. The band is resized automatically to hold the computed field. If there is already a computed field that matches the one being generated, it is skipped.

❖ **To insert a computed field for the current date or page number:**

- 1 Select Insert>Control from the menu bar.
- 2 Select Today() or Page n of n from the options at the bottom of the cascading menu.

The same options are available at the bottom of the Controls drop-down toolbar on the PainterBar.

- 3 Click anywhere in the DataWindow object.

If you selected Today, DataWindow Designer inserts a computed field containing this expression: `TODAY()`. For Page n of n, the computed field contains this expression: `'Page ' + page() + ' of ' + pageCount()`.

Adding custom buttons that place computed fields

You can add buttons to the PainterBar in the DataWindow painter that place computed fields using any of the aggregate functions, such as Max, Min, and Median.

❖ **To customize the PainterBar with custom buttons for placing computed fields:**

- 1 Place the mouse pointer over the PainterBar and select Customize from the pop-up menu.

The Customize dialog box displays.

- 2 Click Custom in the Select palette group to display the set of custom buttons.

- 3 Drag a custom button into the Current toolbar group and release it.

The Toolbar Item Command dialog box displays.

- 4 Click the Function button.
The Function For Toolbar dialog box displays.
- 5 Select a function and click OK.
You return to the Toolbar Item Command dialog box.
- 6 Specify text and microhelp that displays for the button, and click OK.
DataWindow Designer places the new button in the PainterBar. You can click it to add a computed field to your DataWindow object the same way you use the built-in Sum button.

Adding buttons to a DataWindow object

The Button control is a command or picture button that can be placed in a DataWindow object. When clicked at runtime, the button activates either a built-in or user-supplied action.

For example, you can place a button in a report and specify that clicking it opens the Filter dialog box, where users can specify a filter to be applied to the currently retrieved data.

❖ **To add a button to a DataWindow object:**

- 1 Select Insert>Control>Button from the menu bar.
- 2 Click where you want the button to display.

You may find it useful to put a Delete button or an Insert button in the detail band. Clicking a Delete button in the detail band will delete the row next to the button clicked. Clicking an Insert button in the detail band will insert a row following the current row.

Be careful when putting buttons in the detail band

Buttons in the detail band repeat for every row of data, which is not always desirable. Buttons in the detail band are not visible during retrieval, so a Cancel button in the detail band would be unavailable when needed.

- 3 With the button still selected, type the text to display on the button.
- 4 Display the General page of the Properties view for the button.

- 5 Select the action you want to assign to the button from the Action drop-down list.

For information about actions, see “Actions assignable to buttons in DataWindow objects” on page 211.
- 6 If you want to add a picture to the button, select the Action Default Picture check box or enter the name of the Picture file to display on the button.
- 7 Click OK.

Controlling the display of buttons in print preview and on printed output

You can choose whether to display buttons in print preview or in printed output. You control this in the Properties view for the DataWindow object (not the Properties view for the button).

❖ **To control the display of buttons in a DataWindow object in print preview and on printed output:**

- 1 Display the DataWindow object’s Properties view with the Print Specification page on top.
- 2 Select the Display Buttons – Print check box.

The buttons are included in the printed output when the DataWindow object is printed.
- 3 Select the Display Buttons – Print Preview check box.

The buttons display on the screen when viewing the DataWindow object in print preview.

Actions assignable to buttons in DataWindow objects

Table 7-2 shows the actions you can assign to a button in a DataWindow object. Each action is associated with a numeric value (the Action DataWindow object property) and a return code (the actionreturncode event argument).

Table 7-2: Button actions for DataWindow objects

Action	Effect	Value	Action return code
User Defined (default)	Allows the developer to program the ButtonClicked event with no intervening action occurring.	0	The return code from the user's coded event script.
Retrieve (Yield)	Retrieves rows from the database. Before retrieval occurs, the option to yield is turned on; this will allow the Cancel action to take effect during a long retrieve.	1	Number of rows retrieved. -1 if retrieve fails.
Retrieve	Retrieves rows from the database. The option to yield is not automatically turned on.	2	Number of rows retrieved. -1 if retrieve fails.
Cancel	Cancels a retrieval that has been started with the option to yield.	3	0
Page Next	Scrolls to the next page.	4	The row displayed at the top of the DataWindow control when the scrolling is complete or attempts to go past the first row. -1 if an error occurs.
Page Prior	Scrolls to the prior page.	5	The row displayed at the top of the DataWindow control when the scrolling is complete or attempts to go past the first row. -1 if an error occurs.
Page First	Scrolls to the first page.	6	1 if successful. -1 if an error occurs.
Page Last	Scrolls to the last page.	7	The row displayed at the top of the DataWindow control when the scrolling is complete or attempts to go past the first row. -1 if an error occurs.
Sort	Displays Sort dialog box and sorts as specified.	8	1 if successful. -1 if an error occurs.
Filter	Displays Filter dialog box and filters as specified.	9	Number of rows filtered. Number < 0 if an error occurs.
Delete Row	If button is in detail band, deletes row associated with button; otherwise, deletes the current row.	10	1 if successful. -1 if an error occurs.
Append Row	Inserts row at the end.	11	Row number of newly inserted row.

Action	Effect	Value	Action return code
Insert Row	If button is in detail band, inserts row using row number associated with the button; otherwise, inserts row using the current row.	12	Row number of newly inserted row.
Update	Saves changes to the database. If the update is successful, a Commit will be issued; if the update fails, a Rollback will be issued.	13	1 if successful. -1 if an error occurs.
Save Rows As	Displays Save As dialog box and saves rows in the format specified.	14	Number of rows filtered. Number < 0 if an error occurs.
Print	Prints one copy of the DataWindow object.	15	0
Preview	Toggles between preview and print preview.	16	0
Preview With Rulers	Toggles between rulers on and off.	17	0
Query Mode	Toggles between query mode on and off.	18	0
Query Sort	Allows user to specify sorting criteria (forces query mode on).	19	0
Query Clear	Removes the WHERE clause from a query (if one was defined).	20	0

Adding graphs to a DataWindow object

Graphs are one of the best ways to present information. For example, if your application displays sales information over the course of a year, you can easily build a graph in a DataWindow object to display the information visually.

DataWindow Designer offers many types of graphs and provides you with the ability to control the appearance of a graph to best meet your application's needs.

For information on using graphs, see Chapter 15, “Working with Graphs.”

Adding InkPicture controls to a DataWindow object

The InkPicture control is designed for use on a Tablet PC and provides the ability to capture ink input from users of Tablet PCs. The control captures signatures, drawings, and other annotations that do not need to be recognized as text.

The InkPicture control is fully functional on Tablet PCs. If the Microsoft Tablet PC Software Development Kit (SDK) 1.7 is installed on other computers, InkPicture controls in DataWindow objects can accept ink input from the mouse.

For more information about ink controls and the Tablet PC, and to download the Tablet PC SDK, go to the Microsoft Tablet PC Web site at <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/tabletpc.asp>

You use an InkPicture control with a table that has a blob column to store the ink data, and optionally a second blob column to provide a background image.

The InkPicture control behaves like a Picture control that accepts annotation. You can associate a picture with the control so that the user can draw annotations on the picture, then save the ink, the picture, or both. If you want to use the control to capture and save signatures, you usually do not associate a picture with it.

To add an InkPicture control to a DataWindow object, select **Insert>Control>InkPicture** from the menu. A dialog box displays to let you specify a blob column to store the ink data and another to use as a background image. After you specify the columns in the dialog box, the InkPicture control displays in the DataWindow and its Properties view includes a Definition tab page where you can view or change the column definitions.

Adding reports to a DataWindow object

You can nest reports (nonupdatable DataWindow objects) in a DataWindow object.

For information on nesting reports, see Chapter 11, “Using Nested Reports.”

Reorganizing controls in a DataWindow object

This section describes the activities that help you change the layout and appearance of the controls in a DataWindow object.

Displaying boundaries for controls in a DataWindow object

When reorganizing controls in the Design view, it is sometimes helpful to see how large all the controls are. That way you can easily check for overlapping controls and make sure that the spacing around controls is what you want.

❖ **To display control boundaries in a DataWindow object:**

- 1 Select Design>Options from the menu bar.

The DataWindow Options dialog box displays.

- 2 Select the Show Edges check box.

DataWindow Designer displays the boundaries of each control in the DataWindow object.

Boundaries display only in the Design view

The boundaries displayed for controls are for use only in the Design view. They do not display in a running DataWindow object or in a printed report.

Using the grid and the ruler in a DataWindow object

The DataWindow painter provides a grid and a ruler to help you align controls.

❖ **To use the grid and the ruler:**

- 1 Select Design>Options from the menu bar.

The DataWindow Options dialog box displays. The Alignment Grid box contains the alignment grid options.

- 2 Use the options as needed:

Option	Meaning
Snap to Grid	Make controls snap to a grid position when you place them or move them.
Show Grid	Show or hide the grid when the workspace displays.
X	Specify the size (width) of the grid cells.
Y	Specify the size (height) of the grid cells.
Show Ruler	Show a ruler. The ruler uses the units of measurement specified in the Style dialog box. See “Changing the DataWindow object style” on page 171.

Your choices for the grid and the ruler are saved and used the next time you start DataWindow Designer.

Deleting controls in a DataWindow object

- ❖ **To delete controls in a DataWindow object:**
 - 1 Select the controls you want to delete.
 - 2 Select Edit>Delete from the menu bar or press the Delete key.

Moving controls in a DataWindow object

In all presentation styles except Grid

In all presentation styles except Grid, you can move all the controls (such as headings, labels, columns, graphs, and drawing controls) anywhere you want.

- ❖ **To move controls in a DataWindow object:**
 - 1 Select the controls you want to move.
 - 2 Do one of the following:
 - Drag the controls with the mouse.
 - Press an arrow key to move the controls in one direction.

In grid DataWindow objects

You can reorder columns in a grid DataWindow object at runtime. See “Working in a grid DataWindow object” on page 166.

Copying controls in a DataWindow object

You can copy controls within a DataWindow object and to other DataWindow objects. All properties of the controls are copied.

- ❖ **To copy a control in a DataWindow object:**
 - 1 Select the control.
 - 2 Select Edit>Copy from the menu bar.

The control is copied to a private DataWindow Designer clipboard.
 - 3 Copy (paste) the control to the same DataWindow object or to another one:
 - To copy the control within the same DataWindow object, select Edit>Paste from the menu bar.

- To copy the control to another DataWindow object, open the desired DataWindow object and paste the control.

DataWindow Designer pastes the control at the same location as in the source DataWindow object. If you are pasting into the same DataWindow object, you should move the pasted control so it does not cover the original control. DataWindow Designer displays a message box if the control you are pasting is not valid in the destination DataWindow object.

Resizing controls in a DataWindow object

You can resize a control using the mouse or the keyboard. You can also resize multiple controls to the same size using the Layout drop-down toolbar on PainterBar2.

Using the mouse

To resize a control using the mouse, select it, then grab an edge and drag it with the mouse.

Using the keyboard

To resize a control using the keyboard, select it and then do the following:

To make the control	Press
Wider	Shift+Right Arrow
Narrower	Shift+Left Arrow
Taller	Shift+Down Arrow
Shorter	Shift+Up Arrow

In grid DataWindow objects

You can resize columns in grid DataWindow objects.

❖ To resize a column in a grid DataWindow object:

- 1 Position the mouse pointer at a column boundary.
The pointer changes to a two-headed arrow.
- 2 Press and hold the left mouse button and drag the mouse to move the boundary.
- 3 Release the mouse button when the column is the correct width.

Aligning controls in a DataWindow object

Often you want to align several controls or make them all the same size. You can use the grid to align the controls or you can have DataWindow Designer align them for you.

❖ **To align controls in a DataWindow object:**

- 1 Select the control whose position you want to use to align the others.
DataWindow Designer displays handles around the selected control.
- 2 Extend the selection by pressing and holding the Ctrl key and clicking the controls you want to align with the first one.
All the controls have handles on them.
- 3 Select Format>Align from the menu bar.
- 4 From the cascading menu, select the dimension along which you want to align the controls.

For example, to align the controls along the left side, select the first choice on the cascading menu. You can also use the Layout drop-down toolbar on PainterBar2.

DataWindow Designer moves all the selected controls to align with the first one.

Equalizing the space between controls in a DataWindow object

If you have a series of controls and the spacing is fine between two of them but wrong for the rest, you can easily equalize the spacing around all the controls.

❖ **To equalize the space between controls in a DataWindow object:**

- 1 Select the two controls whose spacing is correct.
To do so, click one control, then press Ctrl and click the second control.
- 2 Select the other controls whose spacing match that of the first two controls. To do so, press Ctrl and click each control.
- 3 Select Format>Space from the menu bar.
- 4 From the cascading menu, select the dimension whose spacing you want to equalize.

You can also use the Layout drop-down toolbar on PainterBar2.

Equalizing the size of controls in a DataWindow object

Suppose you have several controls in a DataWindow object and want their sizes to be the same. You can accomplish this manually or by using the Format menu.

❖ **To equalize the size of controls in a DataWindow object:**

- 1 Select the control whose size is correct.
- 2 Press Ctrl and click to select the other controls whose size should match that of the first control.
- 3 Select Format>Size from the menu bar.
- 4 From the cascading menu, select the dimension whose size you want to equalize.

You can also use the Layout drop-down toolbar on PainterBar2.

Sliding controls to remove blank space in a DataWindow object

You can specify that you want to eliminate blank lines or spaces in a DataWindow object by sliding columns and other controls to the left or up if there is blank space. You can use this feature to remove blank lines in mailing labels or to remove extra spaces between fields (such as first and last name).

Slide is used by default in nested reports

DataWindow Designer uses slide options automatically when you nest a report to ensure that the reports are positioned properly.

❖ **To use sliding columns or controls in a DataWindow object:**

- 1 Select Properties from the control's pop-up menu and then select the Position tab in the Properties view.

2 Select the Slide options you want:

Option	Description
Slide Left	Slide the column or control to the left if there is nothing to the left. Be sure the control does not overlap the control to the left. Sliding left will not work if the controls overlap.
Slide Up - All Above	Slide the column or control up if there is nothing in the row above. The row above must be completely empty for the column or control to slide up.
Slide Up - Directly Above	Slide the column or control up if there is nothing <i>directly above it</i> in the row above.

These options are also available on PainterBar2.

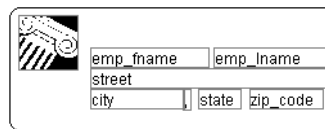
If you are sliding columns up

Even blank columns have height; if you want columns to slide up, you need to specify as Autosize Height all columns above them that might be blank and that you want to slide other columns up through.

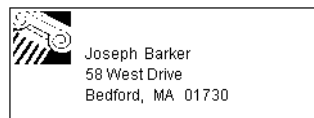
Example

In a mailing label that includes first and last names, as well as address information, you can use sliding to combine the columns appropriately.

In the following label, emp_fname, the comma, state, and zip_code are specified as slide left. Edges are shown to indicate the spacing between the columns. Notice that there is a small amount of space between controls. This space is necessary for Slide Left to work properly:



When you preview (run) the DataWindow object, the last name, comma, state, and zip code slide left to remove the blank space:



Positioning controls in a DataWindow object

Table 7-3 shows the properties for each control in a DataWindow object that determine how it is positioned within the DataWindow object.

Table 7-3: Position properties for controls in a DataWindow object

Property	Meaning
Background	Control is behind other controls. It is not restricted to one band. This is useful for adding a watermark (such as the word CONFIDENTIAL) to the background of a report.
Band	Control is placed within one band. It cannot extend beyond the band's border.
Foreground	Control is in front of other controls. It is not restricted to one band.
Moveable	Control can be moved at runtime and in preview. This is useful for designing layout.
Resizable	Control can be resized at runtime and in preview. This is useful for designing layout.
HideSnaked	Control appears only in the first column on the page; in subsequent columns the control does not appear. This is only for newspaper columns, where the entire DataWindow object snakes from column to column (set on the General page of the Properties view).

Default positioning

DataWindow Designer uses the defaults shown in Table 7-4 when you place a new control in a DataWindow object.

Table 7-4: Default position properties for controls in a DataWindow object

Control	Default positioning
Graph	Foreground, movable, resizable
All other controls	Band, not movable, not resizable

❖ To change the position of a control in a DataWindow object:

- 1 Select Properties from the control's pop-up menu and then select the Position tab.
- 2 From the Layer option drop-down list, select Background, Band, or Foreground.
- 3 Select Resizable or Moveable as appropriate.

Rotating controls in a DataWindow object

Controls that display text such as text controls, columns, and computed fields can be rotated from the original baseline of the text. The Escapement option on the Font property page for the control lets you specify the amount of rotation, also known as escapement.

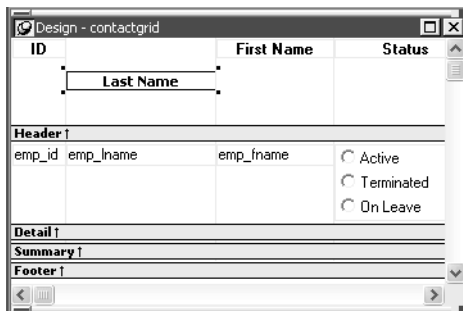
Several other properties of a rotated control affect its final placement when the DataWindow object runs. The location of the control in Design view, the amount of rotation specified for it, and the location of the text within the control (for example, centered text as opposed to left-aligned text) all contribute to what you see in the DataWindow object Preview view.

The following procedure includes design practices that help ensure that you get the final results you want. As you become more experienced, you can drop or alter some of the steps. The procedure recommends setting a visible border on the control so that you can see where the control is located in the Preview view and making the control movable in the Preview view, which is often helpful.

❖ **To rotate a control in a DataWindow object:**

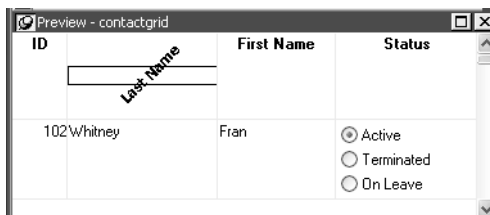
- 1 Select the control in the Design view.
- 2 Change its border to Box (General property page>Border>Box) and make it movable (Position property page>Moveable check box).
- 3 In Design view, enlarge the area in which the control is placed.
For example, in a grid DataWindow object, make the band deeper and move the control down into the center of the band.
- 4 Display the Modify expression dialog box for the Escapement property. (Click the button next to the Escapement property on the Font property page.)
- 5 Specify the amount of rotation you want as an integer in tenths of a degree. (For example, 450 means 45 degrees of rotation; 0 means horizontal or no rotation.)

The origin of rotation is the center of the top border of the box containing the text. It is often helpful to use left-aligned text (General property page>Alignment>Left) because it makes it easier to position the control correctly. This example shows text centered within the control.



If the box that contains the text overlaps the border of the page or the border of a label in a DataWindow object with the Label presentation style, the origin of rotation is the center of the portion of the top border that is within the page or label, and the portion that is outside the page or label is cut off. This can cause the text in the box to run to a second line when it is rotated. If you want the text to display close to the border, you can add one or more line breaks (“~r~n”) before the text and adjust the size of the box.

- 6 To display the current rotation in Preview, close the Preview view and reopen it (View>Preview on the menu bar).



- 7 Drag and drop the control in the Preview view or Design view until it is where you want it.
- 8 In Design view, select the control that is being rotated, remove the temporary border, and deselect the Moveable check box.

If you are using a conditional expression for rotation

If you are specifying different rotations depending on particular conditions, you might need to add conditions to the x and y properties for the control to move the control conditionally to match the various amounts of rotation. An alternative to moving the control around is to have multiple controls positioned exactly as you want them, taking into account the different amounts of rotation. Then you can add a condition to the visible property of each control to ensure that the correctly rotated control shows.

About this chapter

This chapter describes how to customize your DataWindow object by modifying the display values in columns and specifying validation rules.

Contents

Topic	Page
About displaying and validating data	225
About display formats	227
Working with display formats	228
Defining display formats	231
About edit styles	238
Working with edit styles	240
Defining edit styles	242
Defining a code table	251
About validation rules	255
Working with validation rules	256
Defining validation rules	257
How to maintain extended attributes	263

About displaying and validating data

When DataWindow Designer generates a basic DataWindow object, it uses the extended attributes defined for the data and stored in the extended attribute system tables.

For more information about the extended attribute system tables, see Appendix B, “The Extended Attribute System Tables.”

In the Database painter, you can create the extended attribute definitions that specify a column's display format, edit style, and validation rules.

In the DataWindow painter, you can override these extended attribute definitions for a column in a DataWindow object. These overrides do not change the information stored with the column definition in the extended attribute system tables.

Presenting the data

When you generate a new DataWindow object, DataWindow Designer presents the data according to the properties already defined for a column, such as a column's display format and edit style.

Display formats

Display formats embellish data values while still displaying them as letters, numbers, and special characters. Using display formats, for example, you can:

- Change the color of numbers to display a negative value
- Add parentheses and dashes to format a telephone number
- Add a dollar sign and period to indicate a currency format

For information, see “About display formats” on page 227.

Edit styles

Edit styles usually take precedence over display formats and specify how column data is presented. For example, using edit styles, you can:

- Display valid values in a drop-down list
- Indicate that a single value is selected by a check box
- Indicate which of a group of values is selected with radio buttons

Edit styles affect not only the way data displays, they also affect how the user interacts with the data at runtime.

For more information, see “About edit styles” on page 238.

About display format masks and EditMask masks

The differences between display format masks and EditMask masks can be confusing. A display format mask determines the appearance of the column when the focus is *off* the column, or when the DataWindow object is in print preview mode. When you apply an EditMask edit style, the mask you use determines the appearance of the column when focus is *on* the column.

If you want data to display differently depending on whether the focus is on or off the column, specify an edit mask (on the Edit property page for the column) as well as a display format (on the Format property page for the column), then check the Use Format check box on the Format property page. The Use Format check box displays only when an edit mask has been specified.

If you want the data to display in the same way whether focus is on or off the column and you have defined an edit mask, you do not need to define a display format. The edit mask is used for display if the Use Format box is not checked (the default).

Validating data

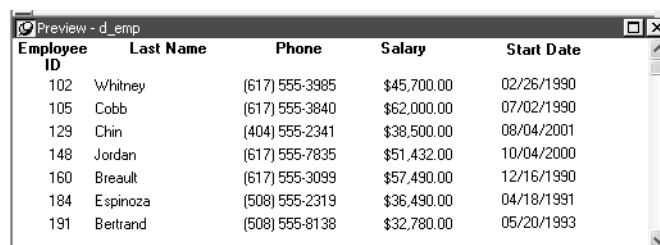
When data is entered in the Database painter or in a DataWindow object, DataWindow Designer evaluates the data against validation rules defined for that column. If the data is valid, DataWindow Designer accepts the entry; otherwise, DataWindow Designer displays an error message and does not accept the entry.

For more information, see “About validation rules” on page 255.

About display formats

You can use display formats to customize the display of column data in a DataWindow object. Display formats are masks in which certain characters have special significance. For example, you can display currency values preceded by a dollar sign, show dates with month names spelled out, and use a special color for negative numbers. DataWindow Designer comes with many predefined display formats. You can use them as is or define your own.

Here the Phone, Salary, and Start Date columns use display formats so the data is easier to interpret:



Employee ID	Last Name	Phone	Salary	Start Date
102	Whitney	(617) 555-3985	\$45,700.00	02/26/1990
105	Cobb	(617) 555-3840	\$62,000.00	07/02/1990
129	Chin	(404) 555-2341	\$38,500.00	08/04/2001
148	Jordan	(617) 555-7835	\$51,432.00	10/04/2000
160	Breault	(617) 555-3099	\$57,490.00	12/16/1990
184	Espinoza	(508) 555-2319	\$36,490.00	04/18/1991
191	Bertrand	(508) 555-8138	\$32,780.00	05/20/1993

Display formats not used for data entry

When users tab to a column containing a display format, DataWindow Designer removes the display format and displays the raw value for users to edit.

If you want to provide formatting used for data entry, you need to specify edit masks, as described in “The EditMask edit style” on page 246.

Working with display formats

You work with display formats in the Database painter and the DataWindow painter.

What you do in the Database painter

In the Database painter, you can:

- Create, modify, and delete named display formats

The named display formats are stored in the extended attribute system tables. When you have defined a display format, it can be used by any column of the appropriate datatype in the database.

- Assign display formats to columns and remove them from columns

These formats are used by default when you place the column in a DataWindow object in the DataWindow painter.

What you do in the DataWindow painter

In the DataWindow painter, you can:

- Accept the default display format assigned to a column in the Database painter
- Override the default display format with another named format stored in the extended attribute system tables
- Create an ad hoc, unnamed format to use with one specific column

Display formats and the extended attribute system tables

When you have placed a column in a DataWindow object and have given it a display format (either the default format from the assignment made in the Database painter for the column or a format assigned in the DataWindow painter), there is no longer any link to the named format in the extended attribute system tables.

If the definition of the display format later changes in the extended attribute system tables, the format for the column in a DataWindow object does not change. If you want to use the modified format, you can reapply it to the column in the DataWindow painter.

Working with display formats in the Database painter

Typically, you define display formats and associate them with columns in the Database painter, because display formats are properties of the data itself. Once you have associated a display format with a column in the Database painter, it is used by default each time the column is placed in a DataWindow object.

Edit style takes precedence

If a column has an associated edit style, the edit style takes precedence over a display format unless you use an EditMask edit style and check the Use Format box on the Format property page.

For more information, see “About edit styles” on page 238.

❖ To create a new display format:

- 1 In the Database painter, select Object>Insert>Display Format from the menu bar.

The Display Format view displays.

- 2 Name the display format and specify a datatype.
- 3 Define the display format using masks.

For information, see “Defining display formats” on page 231.

You can use this display format with any column of the appropriate datatype in the database.

❖ To modify an existing display format:

- 1 In the Database painter, open the Extended Attributes view.
- 2 In the Extended Attributes view, open the list of display formats.
- 3 Position the pointer on the display format you want to modify, display the pop-up menu, and select Properties.
- 4 In the Display Format view, modify the display format as desired.

For information, see “Defining display formats” on page 231.

❖ To associate a display format with a column in the Database painter:

- 1 In the Database painter Objects view, position the pointer on the column, select Properties from the pop-up menu, and select the Display tab in the Properties view.
- 2 Select a format from the list in the Display Format box.

The column now has the selected format associated with it in the extended attribute system tables.

❖ **To remove a display format from a column in the Database painter:**

- 1 In the Database painter Objects view, position the pointer on the column, select Properties from the pop-up menu, and select the Display tab in the Properties view.
- 2 Select (None) from the list in the Display Format box.

The display format is no longer associated with the column.

Working with display formats in the DataWindow painter

Display formats you assign to a column in the Database painter are used by default when you place the column in a DataWindow object. You can override the default format in the DataWindow painter by choosing another format from the extended attribute system tables or defining an ad hoc format for one specific column.

About computed fields

You can assign display formats to computed fields using the same techniques as for columns in a table.

❖ **To specify a display format for a column in the DataWindow painter:**

- 1 In the DataWindow painter, move the pointer to the column, select Properties from the column's pop-up menu, and then select the Format tab.

Information appropriate to the datatype of the selected column displays. The currently used format displays in the Format box. All formats for the datatype defined in the extended attribute system tables are listed in the pop-up list (displayed by clicking the button).

- 2 Do one of the following:
 - Delete the display format.
 - Select a format in the extended attribute system tables from the pop-up list.
 - Create a format for the column by typing it in the Format box. For more information, see "Defining display formats" next.

Format not saved in the extended attribute system tables

If you create a format here, it is used only for the current column and is not saved in the extended attribute system tables.

Shortcuts

To assign the Currency or Percent display format to a numeric column in a report, select the column, then click the Currency or Percent button in the PainterBar or select Format>Currency or Format>Percent from the menu bar.

Customizing the toolbar

You can add buttons to the PainterBar that assign a specified display format to selected columns in reports.

For more information, see “Customizing toolbars” on page 23.

Defining display formats

Display formats are represented through masks, where certain characters have special significance. DataWindow Designer supports four kinds of display formats, each using different mask characters:

- Numbers
- Strings
- Dates
- Times

For example, in a string format mask, each @ represents a character in the string and all other characters represent themselves. You can use the following mask to display phone numbers:

(@@@) @@@-@@@@

Combining formats

You can include different types of display format masks in a single format. Use a space to separate the masks. For example, the following format section includes a date and time format:

mmmm/dd/yyyy h:mm

Using sections

Each type of display format can have multiple sections, with each section corresponding to a form of the number, string, date, or time. Only one section is required; additional sections are optional and should be separated with semicolons (;). You cannot use sections in edit masks. Semicolons can be used only in display formats.

The following format specifies different displays for positive and negative numbers—negative numbers are displayed in parentheses:

\$#,##0;(\$#,##0)

Using keywords Enclose display format keywords in square brackets. For example, you can use the keyword [General] when you want DataWindow Designer to determine the appropriate format for a number.

Using colors You can define a color for each display format section by specifying a color keyword before the format. The color keyword is the name of the color, or a number that represents the color, enclosed in square brackets: [RED] or [255]. The number is usually used only when a color is required that is not provided by name. The named color keywords are:

- [BLACK]
- [BLUE]
- [CYAN]
- [GREEN]
- [MAGENTA]
- [RED]
- [WHITE]
- [YELLOW]

The formula for combining primary color values into a number is:

$$256*256*blue + 256*green + red=number$$

where the amount of each primary color is specified as a value from 0 to 255. For example, to specify cyan, substitute 255 for blue, 255 for green, and 0 for red. The result is 16776960.

If you want to add text to a numeric display format and use a color attribute, you must include the escape character (\) before each literal in the mask. For example:

[red] \D\e\p\t\ : ###

Table 8-1 lists the blue, green, and red values you can use in the formula to create other colors.

Table 8-1: Numeric values used to create colors

Blue	Green	Red	Number	Color
0	0	255	255	Red
0	255	0	65280	Green
0	128	0	32768	Dark green
255	0	0	16711680	Blue
0	255	255	65535	Yellow
0	128	128	32896	Brown
255	255	0	16776960	Cyan
192	192	192	12632256	Light gray

Using special characters

To include a character in a mask that has special meaning in a display format, such as [, precede the character with a backslash (\). For example, to display a single quotation mark, enter \'.

Setting display formats at runtime

In code, you can use the Format property to get and set the format for a column.

Number display formats

A number display format can have up to four sections. Only the first is required. The three other sections determine how the data displays if its value is negative, zero, or NULL. The sections are separated by semi-colons:

Positive-format;negative-format;zero-format>null-format

Special characters

Table 8-2 lists characters that have special meaning in number display formats.

Table 8-2: Characters with special meaning in display formats

Character	Meaning
#	A number
0	A required number; a number will display for every 0 in the mask

Percent signs, decimal points, parentheses, and spaces display as entered in the mask.

Use at least one 0

In general, a number display format should include at least one 0. If users enter 0 in a field with the mask ###, the field will appear to be blank if you do not provide a zero-format section. If the mask is ###.##, only the period displays. If you want two decimal places to display even if both are 0, use the mask ##0.00.

Number keywords

You can use the following keywords as number display formats when you want DataWindow Designer to determine an appropriate format to use:

- [General]
- [Currency]

Note that [Currency(7)] and [Currency(n)] are legal edit masks, but they are *not* legal display formats.

Number and currency settings

To ensure that an application behaves the same in every country where it is deployed, DataWindow expressions and the masks used in display formats and edit masks require U.S. notation for numbers. That is, when you specify a number in a DataWindow expression or in a number mask, a comma always represents the thousands delimiter and a period always represents the decimal place. You should also always use the \$ sign to represent the symbol for currency.

At runtime, the locally correct symbols are displayed for numbers and currency. The comma and period are replaced by the delimiters defined in the user's Number settings in the Regional or International Settings property sheet in the Windows Control Panel. The \$ sign in the mask is replaced by the local currency symbol as defined in the user's Currency setting in the Windows Control Panel. For example, in countries where a comma represents the decimal place and a period represents thousands, users see numbers in those formats.

Percentages

Use caution when defining an edit mask for a percentage. When you enter a number in a column with a percent edit mask and tab off the column, DataWindow Designer divides the number by 100 and stores the result in the buffer. For example, if you enter 23, DataWindow Designer passes .23 to the buffer. When you retrieve from the database, DataWindow Designer multiplies the number by 100 and, if the mask is ##0%, displays 23%.

The datatype for the column must be numeric or decimal to handle the result of a division by 100. If the column has an integer datatype, a percentage entered as 333 is retrieved from the database as 300, and 33 is retrieved as 0.

If you use an edit mask with decimals, such as ##0.00%, the datatype must have enough decimal places to handle the division. For example, if you enter 33.33, the datatype for the column must have at least four decimal places because the result of the division is .3333. If the datatype has only three decimal places, the percentage is retrieved as 33.30.

Examples

Table 8-3 shows how the values 5, -5, and .5 display when different format masks are applied.

Table 8-3: Number display format examples

Format	5	-5	.5
[General]	5	-5	0.5
0	5	-5	1
0.00	5.00	-5.00	0.50
#,##0	5	-5	1
#,##0.00	5.00	-5.00	0.50

Format	5	-5	.5
\$#,##0;(\$#,##0)	\$5	(\$5)	\$1
\$#,##0;-\$#,##0	\$5	-\$5	\$1
\$#,##0;[RED](\$#,##0)	\$5	(\$5)	\$1
[Currency]	\$5.00	(\$5.00)	\$0.50
\$#,##0.00;(\$#,##0.00)	\$5.00	(\$5.00)	\$0.50
\$#,##0.00;[RED](\$#,##0.00)	\$5.00	(\$5.00)	\$0.50
##0%	500%	-500%	50%
##0.00%	500.00%	-500.00%	50.00%
0.00E+00	5.00E+00	-5.00E+00	5.00E-01

String display formats

String display formats can have two sections. The first is required and contains the format for strings; the second is optional and specifies how to represent NULLs:

string-format;null-format

In a string format mask, each at-sign (@) represents a character in the string and all other characters represent themselves.

Example

This format mask:

```
[red] (@@@) @@@-@@@@
```

displays the string 800YESCELT in red as:

```
(800) YES-CELT
```

Date display formats

Date display formats can have two sections. The first is required and contains the format for dates; the second is optional and specifies how to represent NULLs:

date-format;null-format

Special characters

Table 8-4 shows characters that have special meaning in date display formats.

Table 8-4: Characters with special meaning in data display formats

Character	Meaning	Example
d	Day number with no leading zero	9
dd	Day number with leading zero if appropriate	09
ddd	Day name abbreviation	Mon
dddd	Day name	Monday
m	Month number with no leading zero	6
mm	Month number with leading zero if appropriate	06
mmm	Month name abbreviation	Jun
mmmm	Month name	June
yy	Two-digit year	97
yyyy	Four-digit year	1997

Colons, slashes, and spaces display as entered in the mask.

About 2-digit years

If users specify a 2-digit year in a DataWindow object, DataWindow Designer assumes the date is the 20th century if the year is greater than or equal to 50. If the year is less than 50, DataWindow Designer assumes the 21st century. For example:

- 1/1/85 is interpreted as January 1, 1985.
- 1/1/40 is interpreted as January 1, 2040.

Date keywords

You can use the following keywords as date display formats when you want DataWindow Designer to determine an appropriate format to use:

- [ShortDate]
- [LongDate]

The format used is determined by the regional settings for date in the registry. Note that [Date] is not a valid display format.

Examples

Table 8-5 shows how the date Friday, January 30, 1998, displays when different format masks are applied.

Table 8-5: Date display format examples

Format	Displays
[red]m/d/yy	1/30/98 in red
d-mmm-yy	30-Jan-98
dd-mmmm	30-January
mmm-yy	Jan-98
dddd, mmm d, yyyy	Friday, Jan 30, 1998

Time display formats

Time display formats can have two sections. The first is required and contains the format for times; the second is optional and specifies how to represent NULLs:

time-format;null-format

Special characters

Table 8-6 shows characters that have special meaning in time display formats.

Table 8-6: Characters with special meaning in time display formats

Character	Meaning
h	Hour with no leading zero (for example, 1)
hh	Hour with leading zero if appropriate (for example, 01)
m	Minute with no leading zero (must follow h or hh)
mm	Minute with leading zero if appropriate (must follow h or hh)
s	Second with no leading zero (must follow m or mm)
ss	Second with leading zero (must follow m or mm)
ffffff	Microseconds with no leading zeros. You can enter one to six f's; each f represents a fraction of a second (must follow s or ss)
AM/PM	Two-character, uppercase abbreviation (AM or PM as appropriate)
am/pm	Two-character, lowercase abbreviation (am or pm as appropriate)
A/P	One-character, uppercase abbreviation (A or P as appropriate)
a/p	One-character, lowercase abbreviation (a or p as appropriate)

Colons, slashes, and spaces display as entered in the mask.

24-hour format is the default

Times display in 24-hour format unless you specify AM/PM, am/pm, A/P, or a/p.

Time keyword You can use the following keyword as a time display format to specify the format specified in the Windows control panel:

- [Time]

Examples Table 8-7 shows how the time 9:45:33:234567 PM displays when different format masks are applied.

Table 8-7: Time display format examples

Format	Displays
h:mm AM/PM	9:45 PM
hh:mm A/P	09:45 P
h:mm:ss am/pm	9:45:33 pm
h:mm	21:45
h:mm:ss	21:45:33
h:mm:ss:f	21:45:33:2
h:mm:ss:fff	21:45:33:234
h:mm:ss:fffff	21:45:33:234567
m/d/yy h:mm	1/30/98 21:45

About edit styles

You can define edit styles for columns. Edit styles specify how column data is presented in DataWindow objects. Unlike display formats, edit styles do not only affect the display of data; they also affect how users interact with the data at runtime. Once you define an edit style, it can be used by any column of the appropriate datatype in the database.


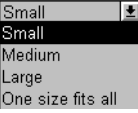
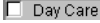



When edit styles are used

If both a display format and an edit style have been assigned to a column, the edit style is always used, with one exception. When you assign an EditMask edit style to a column, you can check the Use Format check box on the Format property page for the column to use the edit mask format when focus is on the column, and the display format mask when focus is off the column.

Edit styles

Table 8-8 shows the available edit styles.

Table 8-8: Edit styles

Edit style	What the edit style does	Example
Edit box (default)	Displays a value in the box For data entry, type a value	
DropDownListBox	Displays a value from the drop-down list For data entry, select or enter a value	
CheckBox	Displays a check box selected or cleared For data entry, select or clear the check box	
RadioButtons	Displays radio buttons, one of which is selected For data entry, select one of the radio buttons	
EditMask	Displays formatted data For data entry, type a value	
DropDownDataWindow	Displays a value from a drop-down DataWindow For data entry, select a value	
InkEdit	On Tablet PCs, displays an InkEdit control so the user can enter data with the stylus.	

For example, suppose you have a column **Status** that takes one of three values: the letters A, T, and L, each representing a status (Active, Terminated, or On Leave). If you assign it the **RadioButton** edit style, users can simply click a button instead of having to type A, T, or L. You do not have to create a validation rule to validate typed input.

Working with edit styles

What you do in the Database painter	<p>You work with edit styles in the Database painter and DataWindow painter.</p> <p>In the Database painter, you can:</p> <ul style="list-style-type: none">• Create, modify, and delete named edit styles <p>The edit styles are stored in the extended attribute system tables. Once you define an edit style, it can be used by any column of the appropriate datatype in the database.</p> <ul style="list-style-type: none">• Assign edit styles to columns <p>These styles are used by default when you place the column in a DataWindow object in the DataWindow painter.</p>
What you do in the DataWindow painter	<p>In the DataWindow painter, you can:</p> <ul style="list-style-type: none">• Accept the default edit style assigned to a column in the Database painter• Override the default edit style with another named style stored in the extended attribute system tables• Create an ad hoc, unnamed edit style to use with one specific column
Edit styles and the extended attribute system tables	<p>When you have placed a column in a DataWindow object and have given it an edit style (either the default style from the assignment made in the Database painter for the column or a style assigned in the DataWindow painter), DataWindow Designer records the name and definition of the edit style in the DataWindow object.</p> <p>However, if the definition of the edit style later changes in the extended attribute system tables, the edit style for the column in a DataWindow object will not change automatically. You can update the column by reassigning the edit style to it in the DataWindow object.</p>

Working with edit styles in the Database painter

Typically, you define edit styles in the Database painter, because edit styles are properties of the data itself. Once defined in the Database painter, the styles are used by default each time the column is placed in a DataWindow object.

❖ To create a new edit style:

- 1 In the Database painter, select Object>Insert>Edit Style from the menu bar.
- 2 In the Edit Style dialog box, select the edit style type from the Style drop-down list.
- 3 Specify the properties of the edit style and click OK.

For information, see “Defining edit styles” on page 242.

You can use the new edit style with any column of the appropriate datatype in the database.

❖ To modify an existing edit style:

- 1 In the Database painter, open the Extended Attributes view.
- 2 In the Extended Attributes view, open the list of edit styles.
- 3 Position the pointer on the Edit style you want to modify, display the pop-up menu, then select Properties.
- 4 In the Edit Style dialog box, modify the edit style as desired and click OK.

For information, see “Defining edit styles” on page 242.

You can use the modified edit style with any column of the appropriate datatype in the database.

❖ To associate an edit style with a column in the Database painter:

- 1 In the Database painter (Objects view), position the pointer on the column, select Properties from the pop-up menu, then select the Edit Style tab in the Properties view.
- 2 Select a style for the appropriate datatype from the list in the Style Name box.

DataWindow Designer associates the selected edit style with the column in the extended attribute system tables.

❖ To remove an edit style from a column in the Database painter:

- 1 In the Database painter (Objects view), position the pointer on the column, select Properties from the pop-up menu, then select the Edit Style tab in the Properties view.
- 2 Select (None) from the list in the Style Name box.

The edit style is no longer associated with the column.

Working with edit styles in the DataWindow painter

An edit style you assign to a column in the Database painter is used by default when you place the column in a DataWindow object. You can override the edit style in the DataWindow painter by choosing another edit style from the extended attribute system tables or defining an ad hoc style for one specific column.

❖ **To specify an edit style for a column:**

- 1 In the DataWindow painter, move the pointer to the column, select Properties from the column's pop-up menu, and then select the Edit tab.
- 2 Select the type of edit style you want from the Style Type drop-down list.
The information on the Edit property page changes to be appropriate to the type of edit style you selected.
- 3 Do one of the following:
 - Select an edit style from the Style Name box.
 - Create an ad hoc edit style for the column, as described in "Defining edit styles" next.

Defining edit styles

This section describes how to specify each type of edit style.

The Edit edit style

By default, columns use the Edit edit style, which displays data in an edit control. You can customize the appearance and behavior of the edit control by modifying a column's Edit edit style. To do so, select Edit in the Style Type drop-down list and specify the properties for that style:

- To restrict the number of characters users can enter, enter a value in the Limit box.
- To convert the case of characters upon display, enter an appropriate value in the Case box.
- To have entered values display as asterisks for sensitive data, check the Password box.

- To allow users to tab to the column but not change the value, check the Display Only box.
- To define a code table to determine which values are displayed to users and which values are stored in the database, check the Use Code Table box and enter display and data values for the code table.

See “Defining a code table” on page 251.

❖ **To use the Edit edit style:**

- 1 Select Edit from the Style Type box, if it is not already selected.
- 2 Select the properties you want.

Date columns and regional settings

Using the Edit edit style, or no edit style, with a date column can cause serious data entry and validation problems if a user’s computer is set up to use a nonstandard date style, such as yyyy/dd/mm. For example, if you enter 2001/03/05 in the Retrieval Arguments dialog box for a date column when the mask is yyyy/dd/mm, the date is interpreted as March 5 instead of May 3. To ensure that the order of the day and month is interpreted correctly, use an EditMask edit style.

The DropDownListBox edit style

You can use the DropDownListBox edit style to have columns display as drop-down lists at runtime:



Typically, this edit style is used with code tables, where you can specify display values (which users see) and shorter data values (which are stored in the database).

In the DropDownListBox edit style, the display values of the code table display in the ListBox portion of the DropDownListBox. The data values are the values that are put in the DataWindow buffer (and sent to the database when an Update is issued) when the user selects an item in the ListBox portion of the drop-down list.

In the preceding example, when users see the value Business Services, the corresponding data value could be 200.

❖ **To use the DropDownListBox edit style:**

- 1 Select DropDownListBox from the Style Type box.
- 2 Select the appropriate properties.
- 3 Enter the value you want to have appear in the Display Value box and the corresponding data value in the Data Value box.

At runtime

You can define and modify a code table for a column in code by using the SetCodeTableValue method at runtime. To obtain the value of a column at runtime, use the GetCodeTableValue method. To clear the code table of values, use the ResetCodeTable method.

For more about code tables, see “Defining a code table” on page 251.

The CheckBox edit style

If a column can take only one of two (or perhaps three) values, you might want to display the column as a check box; users can select or clear the check box to specify a value. In the following entry from a DataWindow object, users can simply check or clear a box to indicate whether an employee has health insurance:

Health Insurance:

❖ **To use the CheckBox edit style:**

- 1 In the Text box, enter the text you want displayed next to the check box.

Using accelerator keys

If the CheckBox has an accelerator key, enter an ampersand (&) before the letter in the text that represents the accelerator key.

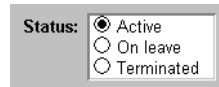
- 2 In the Data Value For boxes, enter the values you want put in the DataWindow buffer when the CheckBox is checked (on) or unchecked (off).

If you selected the 3 States box, an optional third state box (other) appears, for the case when the condition is neither on nor off.

- What happens The value you enter in the Text box becomes the display value, and values entered for On, Off, and Other become the data values.
- When users check or clear the check box at runtime, DataWindow Designer enters the appropriate data value in its buffer. When the UpdateData method is called, DataWindow Designer sends the corresponding data values to the database.
- Centering check boxes without text You may find it useful to center check boxes used for columns of information. First make the text control used for the column header and the column control the same size and left aligned. Then you can center the check boxes and the column header.
- ❖ **To center check boxes without text:**
 - 1 In the Edit property page for the column, make sure the Left Text check box is not selected and that the Text box where you specify associated text is empty.
 - 2 In the General property page, specify centering (Alignment>Center) or specify centering using the StyleBar.

The RadioButtons edit style

If a column can take one of a small number of values, you might want to display the column as radio buttons:



- ❖ **To use the RadioButtons edit style:**
 - 1 Specify how many radio buttons will display in the Columns Across box.
 - 2 Enter a set of display and data values for each button you want to display.

The display values you enter become the text of the buttons; the data values are put in the DataWindow buffer when the button is clicked.

Using accelerator keys

To use an accelerator key on a radio button, enter an ampersand (&) in the Display Value before the letter that will be the accelerator key.

- What happens Users select values by clicking a radio button. When the UpdateData method is issued, the data values are sent to the database.

The EditMask edit style

Sometimes users need to enter data that has a fixed format. For example, in North America phone numbers have a 3-digit area code, followed by three digits, followed by four digits. You can define an edit mask that specifies the format to make it easier for users to enter values:

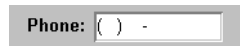


Edit masks consist of special characters that determine what can be entered in the column. They can also contain punctuation characters to aid users.

For example, to make it easier for users to enter phone numbers in the proper format, specify this mask:

(###) ###-####

At runtime, the punctuation characters display in the box and the cursor jumps over them as the user types:



Special characters
and keywords

Edit masks use the same special characters as display formats, and there are special considerations for using numeric, string, date, and time masks.

For information, see “Defining display formats” on page 231.

Semicolons invalid in EditMask edit styles

In a display format, you can use semicolons to separate sections in number, date, time, and string formats. You cannot use semicolons in an EditMask edit style.

Keyboard behavior

Note the following about how certain keystrokes behave in edit masks:

- Both Backspace and Shift + Backspace delete the preceding character.
- Delete deletes everything that is selected.
- Non-numeric edit masks treat any characters that do not match the mask pattern as delimiters.

Also, note certain behavior in Date edit masks:

- Entering zero for the day or month causes the next valid date to be entered. For example, if the edit mask is DD/MM/YY, typing 00/11/01 results in 01/11/01. You can override this behavior in the development environment by adding the following lines to your *DW.INI* file:

```
[Edit Mask Behaviors]
AutocompleteDates=no
```

- The strings 00/00/00 or 00/00/0000 are interpreted as the NULL value for the column.

Using the Mask
pop-up menu

Click the button to the right of the Mask box on the Mask property page to display a list that contains complete masks that you can click to add to the mask box, as well as special characters that you can use to construct your own mask. For example, the menu for a Date edit mask contains complete masks such as mm/dd/yy and dd/mmm/yyyy. It also has components such as dd and jjj (for a Julian day). You might use these to construct a mask like dd-mm-yy, typing in the hyphens as separators.

You cannot use a partial mask, such as dd or mmm, in a date edit mask. Any mask that does not include any characters representing the year will be replaced by a mask that does.

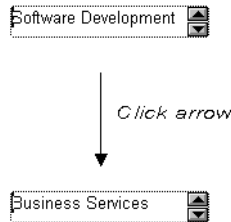
Using masks with “as
is” characters

You can define a mask that contains “as is” characters that always appear in the control or column. For example, you might define a numeric mask such as R\$0000.00 to represent Indian rupees in a currency column. However, you cannot enter a minus sign to represent negative numbers in a mask that contains “as is” characters.

The preferred method of creating a currency editmask is to use the predefined [currency(7)] - International mask. You can change the number in parentheses, which is the number of characters in the mask including two decimal places. When you use this mask, DataWindow Designer uses the currency symbol and format defined in the regional settings section of the Windows control panel. You can enter negative values in a column that uses a currency mask.

Using spin controls

You can define an edit mask as a spin control, a box that contains up and down arrows that users can click to cycle through fixed values. For example, you can set up a code table that provides the valid entries in a column; users simply click an arrow to select an entry. Used this way, a spin control works like a drop-down list that displays one value at a time:



For more about code tables, see “Defining a code table” on page 251.

❖ To use an EditMask edit style:

- 1 Select EditMask in the Style Type box if it is not already selected.
- 2 Define the mask in the Mask box. Click the special characters in the pop-up menu to use them in the mask. To display the pop-up menu, click the button to the right of the Mask box.
- 3 Specify other properties for the edit mask.

When you use your EditMask, check its appearance and behavior. If characters do not appear as you expect, you might want to change the font size or the size of the EditMask.

Using a drop-down calendar

You can use a drop-down calendar option on any DataWindow column with an EditMask edit style and a Date, DateTime, or TimeStamp datatype. The DDCalendar EditMask property option allows for separate selections of the calendar month, year, and date. This option can be set in a check box on the Edit page of the DataWindow painter Properties view when a column with the EditMask edit style is selected. It can also be set in code, as in this example for the birth_date column:

```
dw_1.Modify("birth_date.EditMask.DDCalendar='Yes' ")
```

If you do not include script for client formatting in a Web DataWindow, the drop-down calendar uses a default edit mask to display the column data based on the client computer’s default localization settings. To make sure that dates selected with the drop-down calendar option are displayed with the desired edit mask, specify that the Client Formatting option be included with the static JavaScript generated and deployed for the DataWindow.

To conserve bandwidth, JavaScript for client formatting is not included by default. To include this script, you can select the Client Formatting check box on the Web Generation tab of the DataWindow Properties view.

The drop-down calendar option is supported in all Web DataWindow rendering formats (HTML, XHTML, and XML).

The DropDownDataWindow edit style

Sometimes another data source determines which data is valid for a column.

Consider this situation: the Department table includes two columns, Dept_id and Dept_name, to record your company's departments. The Employee table records your employees. The Department column in the Employee table can have any of the values in the Dept_id column in the Department table.

As new departments are added to your company, you want the DataWindow object containing the Employee table to automatically provide the new departments as choices when users enter values in the Department column.

In situations such as these, you can specify the DropDownDataWindow edit style for a column: it is populated from another DataWindow object. When users go to the column, the contents of the DropDownDataWindow display, showing the latest data:

Department:	Business Services	↓
	Name	Code
Status:	Software Development	100
	Business Services	200
	Corporate Marketing	300
Start date:	Marketing	400
Expiration date:		

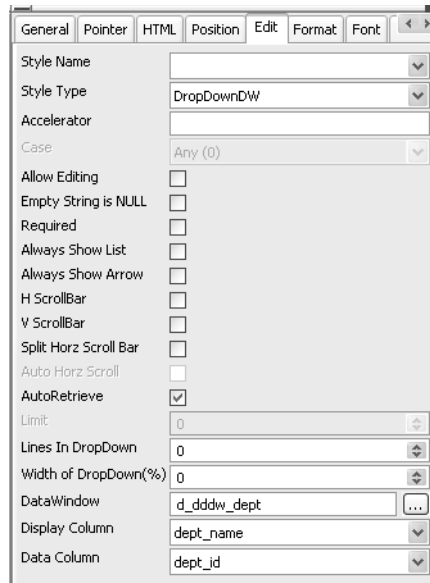
❖ To use the DropDownDataWindow edit style:

- 1 Create a DataWindow object that contains the columns in the detail band whose values you want to use in the column.

You will often choose at least two columns: one column that contains values that the user sees and another column that contains values to be stored in the database. In the example above, you would create a DataWindow object containing the dept_id and dept_name columns in the Department table. Assume this DataWindow object is named d__dddw_dept.

- For the column getting its data from the DataWindow object, select the DropDownDW edit style.

In the example, you would specify the DropDownDataWindow edit style for the dept_name column:



- Click the browse button next to the DataWindow box and select the DataWindow object that contains the data for the column from the drop-down list (in the example, d_dddw_dept).
- In the Display Column box, select the column containing the values that will display in the DataWindow object (in the example, dept_name).
- In the Data Column box, select the column containing the values that will be stored in the database (in the example, dept_id).
- Specify other properties for the edit style and click OK when done.

What happens

At runtime, when data is retrieved into the DataWindow object, the column whose edit style is DropDownDataWindow will itself be populated as data is retrieved into the DataWindow object serving as the drop-down DataWindow object.

When the user goes to the column and drops it down, the contents of the drop-down DataWindow object display. When the user selects a display value, the corresponding data value is stored in the DataWindow buffer and is stored in the database when an UpdateData is issued.

Limit on size of data value

The data value for a column that uses the DropDownDataWindow edit style is limited to 511 characters.

The InkEdit edit style

The InkEdit edit style is designed for use on a Tablet PC and provides the ability to capture ink input from users of Tablet PCs.

You can specify InkEdit as a style type on the Edit page in the Properties view for columns. When the column gets focus, an InkEdit control displays so that the user can enter text with the stylus or mouse. The text is recognized and displayed, then sent back to the database when the column loses focus.

The InkEdit edit style is fully functional on Tablet PCs. On other computers, it behaves like the Edit edit style.

For more information about ink controls and the Tablet PC, and to download the Tablet PC SDK, go to the Microsoft Tablet PC Web site at <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnanchor/html/tabletpc.asp>

Defining a code table

To reduce storage needs, frequently you might want to store short, encoded values in the database, but these encoded values might not be meaningful to users. To make DataWindow objects easy to use, you can define code tables.

Each row in a code table is a pair of corresponding values: a display value and a data value. The display values are those users see at runtime. The data values are those that are saved in the database.

Limit on size of data value

The data value you specify for the Checkbox, DropDownListBox, Edit, EditMask, and RadioButtons edit styles is limited to 255 characters.

How code tables are implemented

You can define a code table as a property of the following column edit styles:

- Edit
- DropDownListBox
- RadioButtons
- DropDownDataWindow
- EditMask, using spin control

The steps to specify the code table property for each edit style are similar: you begin by defining a new edit style in the Database painter. Once you select an edit style, use the specific procedure that follows to define the code table property.

For how to create an edit style, see “About edit styles” on page 238.

Allowing NULL values

An internal DataWindow Designer code, NULL!, indicates null values are allowed. To use this code, specify NULL! as the data value, then specify a display format for nulls for the column.

❖ **To define a code table as a property of the Edit edit style:**

- 1 Select the Use Code Table check box.
- 2 Enter the display and data values for the code table.
- 3 If you want to restrict input in the column to values in the code table, select the Validate check box.

For more information, see “Validating user input” on page 254.

❖ **To define a code table as a property of the DropDownListBox edit style:**

- 1 Enter the display and data values for the code table.
- 2 If you want to restrict input in the column to values in the code table, clear the Allow Editing check box.

For more information, see “Validating user input” on page 254.

❖ **To define a code table as a property of the RadioButtons edit style:**

- Enter the display and data values for the code table.

- ❖ **To define a code table as a property of the DropDownDataWindow edit style:**
 - 1 Specify the column that provides the display values in the Display Column box.
 - 2 Specify the column that provides the data values in the Data Column box.
 - 3 If you want to restrict input to values in the code table, clear the Allow Editing check box.

- ❖ **To define a code table as a property of the EditMask edit style:**
 - 1 Select the Spin Control check box.
 - 2 Select the Code Table check box.
 - 3 Enter the display and data values for the code table.

How code tables are processed

When data is retrieved into a DataWindow object column with a code table, processing begins at the top of the data value column. If the data matches a data value, the corresponding display value displays. If there is no match, the actual value displays.

Consider the example in Table 8-9.

Table 8-9: Data values and display values

Display values	Data values
Massachusetts	MA
Massachusetts	ma
ma	MA
Mass	MA
Rhode Island	RI
RI	RI

If the data is MA or ma, the corresponding display value (Massachusetts) displays. If the data is Ma, there is no match, so Ma displays.

Case sensitivity

Code table processing is case sensitive.

If the code table is in a `DropDownListBox` edit style, and if the column has a code table that contains duplicate display values, then each value displays only once. Therefore, if this code table is defined for a column in a `DataWindow` object that has a `DropDownListBox` edit style, Massachusetts and Rhode Island display in the `ListBox` portion of the `DropDownListBox`.

Validating user input

When users enter data into a column in a `DataWindow` object, processing begins at the top of the display value column of the associated code table.

If the data matches a display value, the corresponding data value is put in the internal buffer. For each display value, the first data value is used. Using the sample code table, if the user enters Massachusetts, ma, or Mass, the data value is MA.

You can specify that *only* the values in the code table are acceptable:

- For a column using the `Edit` edit style, select the `Validate` check box.
If you have selected the `Validate` check box for the `Edit` edit style, an `ItemError` event is triggered whenever a user enters a value not in the code table. Otherwise, the entered value is validated using the column's validation rule, if any, and put in the `DataWindow` buffer.
- For the `DropDownListBox` and `DropDownDataWindow` edit styles, clear the `Allow Editing` check box: users cannot type a value.

Although users cannot type a value when you clear the `Allow Editing` check box, they can search for a row in the drop-down list or `DataWindow` by typing in the initial character for the row display value. The search is case sensitive. For the `DropDownDataWindow` edit style, the initial character for a search cannot be an asterisk or a question mark. This restriction does not apply to the `DropDownListBox` edit style.

When the code table processing is complete, the `ItemChanged` or `ItemError` event is triggered.

Code table data

The data values in the code table must pass validation for the column and must have the same datatype as the column.

About validation rules

When users enter data in a DataWindow object, you want to be sure the data is valid before using it to update the database. Validation rules provide one way to do this.

You usually define validation rules in the Database painter. To use a validation rule, you associate it with a column in the Database painter or DataWindow object painter.

Another technique

You can also perform data validation through code tables, which are implemented through a column's edit style.

For more information, see “About edit styles” on page 238.

Understanding validation rules

Validation rules are criteria that a DataWindow object uses to validate data entered into a column by users. They are DataWindow Designer specific and therefore not enforced by the DBMS.

Validation rules assigned in the Database painter are used by default when you place columns in a DataWindow object. You can override the default rules in the DataWindow object painter.

A validation rule is an expression that evaluates to either TRUE or FALSE. If the expression evaluates to TRUE for an entry into a column, DataWindow Designer accepts the entry. If the expression evaluates to FALSE, the entry is not accepted and the ItemError event is triggered. By default, DataWindow Designer displays a message box to the user. You can customize the message displayed when a value is rejected.

You can also code the ItemError event to cause different processing to happen.

For more information, see the chapter on using DataWindow objects in the *Programmer's Guide*.

At runtime

In code, you can use the `GetColumnValidation` method to obtain the validation rule for a column and the `SetColumnValidation` method to change the validation rule for a column.

For information about the `GetColumnValidation` and `SetColumnValidation` methods, see the online help.

Working with validation rules

You work with validation rules in the Database painter and DataWindow object painter.

What you do in the Database painter

In the Database painter, you can:

- Create, modify, and delete named validation rules

The validation rules are stored in the extended attribute system tables. Once you define a validation rule, it can be used by any column of the appropriate datatype in the database.

- Assign validation rules to columns and remove them from columns

These rules are used by default when you place the column in a DataWindow object in the DataWindow object painter.

What you do in the DataWindow painter

In the DataWindow painter, you can:

- Accept the default validation rule assigned to a column in the Database painter
- Create an ad hoc, unnamed rule to use with one specific column

Validation rules and the extended attribute system tables

Once you have placed a column that has a validation rule from the extended attribute system tables in a DataWindow object, there is no longer any link to the named rule in the extended attribute system tables.

If the definition of the validation rule later changes in the extended attribute system tables, the rule for the column in a DataWindow object will not change.

Defining validation rules

Typically, you define validation rules in the Database painter, because validation rules are properties of the data itself. Once defined in the Database painter, the rules are used by default each time the column is placed in a DataWindow object. You can also define a validation rule in the DataWindow object painter that overrides the rule defined in the Database painter.

Defining a validation rule in the Database painter

This section describes the ways you can manipulate validation rules in the Database painter.

❖ To create a new validation rule

- 1 In the Database painter, select Object>Insert>Validation Rule from the menu bar.

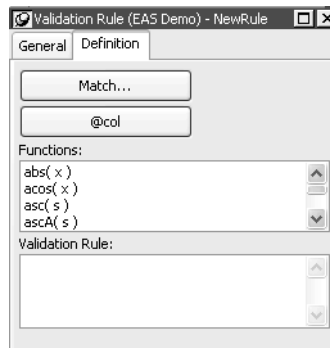
The Validation Rule view displays in the Properties view.

- 2 Assign a name to the rule, select the datatype of the columns to which it applies, and customize the error message (if desired).

For information, see “Customizing the error message” on page 260.

- 3 Click the Definition tab and define the expression for the rule.

For information, see “Defining the expression” on page 258.



You can use this rule with any column of the appropriate datatype in the database.

❖ **To modify a validation rule:**

- 1 In the Database painter, open the Extended Attributes view.
- 2 In the Extended Attributes view, open the list of validation rules.
- 3 Position the pointer on the validation rule you want to modify, display the pop-up menu, and select Properties.
- 4 In the Validation Rule view, modify the validation rule as desired.

For information, see “Defining the expression” on page 258 and “Customizing the error message” on page 260.

❖ **To associate a validation rule with a column in the Database painter:**

- 1 In the Database painter (Objects view), position the pointer on the column, select Properties from the column’s pop-up menu, and select the Validation tab.
- 2 Select a validation rule from the Validation Rule drop-down list.

The column now has the selected validation rule associated with it in the extended attribute system tables. Whenever you use this column in a DataWindow object, it will use this validation rule unless you override it in the DataWindow painter.

❖ **To remove a validation rule from a column in the Database painter:**

- 1 In the Database painter (Objects view), position the pointer on the column, select Properties from its pop-up menu, and select the Validation tab in the Properties view.
- 2 Select (None) from the list in the Validation Rule drop-down list.

The validation rule is no longer associated with the column.

Defining the expression

A validation rule is a boolean expression. DataWindow Designer applies the boolean expression to an entered value. If the expression returns TRUE, the value is accepted. Otherwise, the value is not accepted and an ItemError event is triggered.

What expressions can contain

You can use any valid DataWindow expression in validation rules.

Validation rules can include most DataWindow expression functions. DataWindow expression functions are displayed in the Functions list and can be pasted into the definition.

For information about these functions, see the *DataWindow Object Reference*.

Use the notation *@placeholder* (where *placeholder* is any group of characters) to indicate the current column in the rule. When you define a validation rule in the Database painter, DataWindow Designer stores it in the extended attribute system tables with the placeholder name. At runtime, DataWindow Designer substitutes the value of the column for *placeholder*.

Pasting the placeholder

The `@col` can be easily used as the placeholder. A button in the Paste area is labeled with `@col`. You can click the button to paste the `@col` into the validation rule.

An example

For example, to make sure that both Age and Salary are greater than zero using a single validation rule, define the validation rule as follows:

```
@col > 0
```

Then associate the validation rule with both the Age and Salary columns. At runtime, DataWindow Designer substitutes the appropriate values for the column data when the rule is applied.

Using match values for character columns

If you are defining the validation rule for a character column, you can use the Match button on the Definition page of the Validation Rule view. This button lets you define a match pattern for matching the contents of a column to a specified text pattern (for example, `^[0-9]+$` for all numbers and `^[A-Za-z]+$` for all letters).

❖ To specify a match pattern for character columns:

- 1 Click the Match button on the Definition page of the Validation Rule view. The Match Pattern dialog box displays.
- 2 Enter the text pattern you want to match the column to, or select a displayed pattern.
- 3 (Optional) Enter a test value and click the Test button to test the pattern.
- 4 Click OK when you are satisfied that the pattern is correct.

For more on the Match function and text patterns, see the *DataWindow Object Reference*.

Customizing the error message

When you define a validation rule, DataWindow Designer automatically creates the error message that displays by default when users enter an invalid value:

```
'Item ~' + @Col + '~' does not pass validation test.'
```

You can edit the string expression to create a custom error message.

Different syntax in the DataWindow object painter

If you are working in the DataWindow object painter, you can enter a string expression for the message, but you do not use the @ sign for placeholders. For example, this is the default message:

```
'Item ~' + ColumnName + '~' does not pass validation test.'
```

A validation rule for the Salary column in the Employee table might have the following custom error message associated with it:

```
Please enter a salary greater than $10,000.'
```

If users enter a salary less than or equal to \$10,000, the custom error message displays.

Specifying initial values

As part of defining a validation rule, you can supply an initial value for a column.

- ❖ **To specify an initial value for a column in the Database painter:**
 - 1 Select Properties from the column's pop-up menu and select the Validation tab.
 - 2 Specify a value in the Initial Value box.

Defining a validation rule in the DataWindow painter

Validation rules you assign to a column in the Database painter are used by default when you place the column in a DataWindow object. You can override the validation rule in the DataWindow object painter by defining an ad hoc rule for one specific column.

❖ **To specify a validation rule for a column in the DataWindow painter:**

- 1 In the DataWindow painter, select View>Column Specifications from the menu bar.

The Column Specification view displays.

Name	Type	Prompt	Initial Value	Validation Expression	Validation Message
1 emp_id	number	<input type="checkbox"/>			
2 employee_emp_lname	char(20)	<input type="checkbox"/>			
3 employee_emp_fname	char(20)	<input type="checkbox"/>			
4 salary	decimal(3)	<input type="checkbox"/>		real(gettext()) > 0	'Sorry! The value must be greater than zero.'

- 2 Create or modify the validation expression. To display the Modify Expression dialog box, display the pop-up menu for the box in which you want to enter a Validation Expression and select Expression. Follow the directions in "Specifying the expression" next.
- 3 (Optional) Enter a string or string expression to customize the validation error message.

For more information, see “Customizing the error message” on page 260.

- 4 (Optional) Enter an initial value.

Used for current columns only

If you create a validation rule here, it is used only for the current column and is not saved in the extended attribute system tables.

Specifying the expression

Since a user might have just entered a value in the column, validation rules refer to the current data value, which you can obtain through the `GetText` DataWindow expression function.

Using `GetText` ensures that the most recent data entered in the current column is evaluated.

DataWindow Designer does the conversion for you

If you have associated a validation rule for a column in the Database painter, DataWindow Designer automatically converts the syntax to use `GetText` when you place the column in a DataWindow object.

GetText returns a string. Be sure to use a data conversion function (such as Integer or Real) if you want to compare the entered value with a datatype other than string.

For more on the GetText function and text patterns, see the *DataWindow Object Reference*.

Referring to other columns

You can refer to the values in other columns by specifying their names in the validation rule. You can paste the column names in the rule using the Columns box.

Examples

Here are some examples of validation rules.

Example 1 To check that the data entered in the current column is a positive integer, use this validation rule:

```
Integer(GetText( )) > 0
```

Example 2 If the current column contains the discounted price and the column named Full_Price contains the full price, you could use the following validation rule to evaluate the contents of the column using the Full_Price column:

```
Match(GetText( ), "[0-9]+$") AND  
Real(GetText( )) < Full_Price
```

To pass the validation rule, the data must be all digits (must match the text pattern `[0-9]+$`) and must be less than the amount in the Full_Price column.

Notice that to compare the numeric value in the column with the numeric value in the Full_Price column, the Real function was used to convert the text to a number.

Example 3 In your company, a product price and a sales commission are related in the following way:

- If the price is greater than or equal to \$1000, the commission is between 10 percent and 20 percent
- If the price is less than \$1000, the commission is between 4 percent and 9 percent

The Sales table has two columns, Price and Commission. The validation rule for the Commission column is:

```
(Number(GetText( )) >= If(price >= 1000, .10, .04))  
AND  
(Number(GetText( )) <= If(price >= 1000, .20, .09))
```

A customized error message for the Commission column is:

```
"Price is " + if(price >= 1000,  
"greater than or equal to","less than") +  
" 1000. Commission must be between " +  
If(price >= 1000, ".10", ".04") + " and " +  
If(price >= 1000, ".20.", ".09.")
```

How to maintain extended attributes

DataWindow Designer provides facilities you can use to create, modify, and delete display formats, edit styles, and validation rules independently of their association with columns. The following procedure summarizes how you do this.

❖ **To maintain display formats, edit styles, and validation rules:**

- 1 Open the Database painter.
- 2 Select View>Extended Attributes.

The Extended Attributes view displays listing all the entities in the extended attribute system tables.

- 3 Do one of the following:
 - To create a new entity, display the pop-up menu for the type you want to add, then select New.
 - To modify an entity, display its pop-up menu, then select Properties.
 - To delete an entity, display its pop-up menu, then select Delete.

Caution

If you delete a display format, edit style, or validation rule, it is removed from the extended attribute system tables. Columns in the database are no longer associated with the entity.

Filtering, Sorting, and Grouping Rows

About this chapter

This chapter describes how you can customize your DataWindow object by doing the following in the DataWindow painter:

- Defining filters to limit which of the retrieved rows are displayed in the DataWindow object
- Sorting rows after they have been retrieved from the database
- Displaying the rows in groups and calculating statistics on each group

Contents

Topic	Page
Filtering rows	265
Sorting rows	267
Grouping rows	270

Filtering rows

You can use WHERE and HAVING clauses and retrieval arguments in the SQL SELECT statement for the DataWindow object to limit the data that is retrieved from the database. This reduces retrieval time and space requirements at runtime.

However, you may want to further limit the data that displays in the DataWindow object. For example, you might want to:

- Retrieve many rows and initially display only a subset (perhaps allowing the user to specify a different subset of rows to display at runtime)
- Limit the data that is displayed using DataWindow expression functions (such as If) that are not valid in the SELECT statement

Using filters

In the DataWindow painter, you can define filters to limit the rows that display at runtime. Filters can use most DataWindow expression functions.

Filters do not affect which rows are retrieved

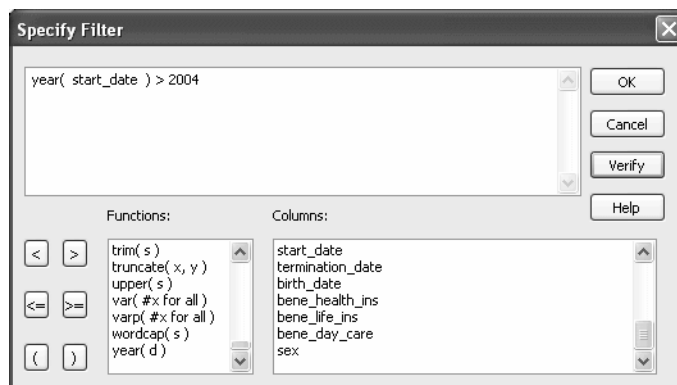
A filter operates against the retrieved data. It does not re-execute the SELECT statement.

Defining a filter

❖ **To define a filter:**

- 1 In the DataWindow painter, select Rows>Filter from the menu bar.

The Specify Filter dialog box displays:



- 2 In the Specify Filter dialog box, enter a boolean expression that DataWindow Designer will test against each retrieved row.

If the expression evaluates to TRUE, the row is displayed. You can specify any valid expression in a filter. You can paste commonly used functions, names of columns, computed fields, retrieval arguments, and operators into the filter.

International considerations

For applications to run the same in any country, filter expressions require U.S. notation for numbers. That is, a comma always represents the thousands delimiter and a period always represents the decimal place when you specify expressions in the development environment.

- 3 (Optional) Click Verify to make sure the expression is valid.
- 4 Click OK.

Only rows meeting the filter criteria are displayed in the Preview view.

Filtered rows and updates

Modifications of filtered rows are applied to the database when you issue an update request.

Removing a filter

❖ **To remove a filter:**

- 1 Select Rows>Filter from the menu bar.
- 2 Delete the filter expression from the Specify Filter dialog box, then click OK.

Examples of filters

Assume that a DataWindow object retrieves employee rows and three of the columns are Salary, Status, and Emp_Lname. Table 9-1 shows some examples of filters you might use.

Table 9-1: Sample filters

To display these rows	Use this filter
Employees with salaries over \$50,000	Salary > 50000
Active employees	Status = 'A'
Active employees with salaries over \$50,000	Salary > 50000 AND Status = 'A'
Employees whose last names begin with H	left(Emp_Lname, 1) = 'H'

Setting filters in a script

You can use the SetFilter and Filter methods in a script to dynamically modify a filter that was set in the DataWindow painter. For information about SetFilter and Filter, see the online help.

Sorting rows

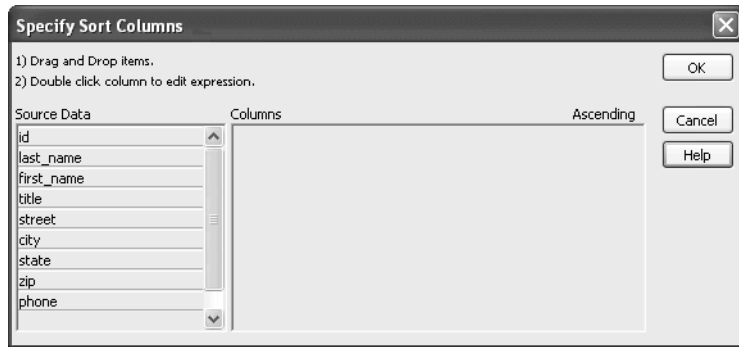
You can use an ORDER BY clause in the SQL SELECT statement for the DataWindow object to sort the data that is retrieved from the database. If you do this, the DBMS itself does the sorting and the rows are brought into DataWindow Designer already sorted.

However, you might want to sort the rows after they are retrieved. For example, you might want to:

- Offload the processing from the DBMS
- Sort on an expression, which might not be allowed in the SELECT statement but is allowed in DataWindow Designer

❖ **To sort the rows:**

- 1 Select Rows>Sort from the menu bar.



- 2 Drag to the Columns box the columns on which you want to sort the rows, and specify whether you want to sort in ascending or descending order.

The order of the columns determines the precedence of the sort. To reorder the columns, drag them up or down in the list. To delete a column from the sort columns list, drag the column outside the dialog box.

- 3 You can also specify expressions to sort on: for example, if you have two columns, Revenues and Expenses, you can sort on the expression *Revenues – Expenses*.

To specify an expression to sort on, double-click a column name in the Columns box, modify the expression in the Modify Expression dialog box, and click OK.

You return to the Specify Sort Columns dialog box with the expression displayed.

If you change your mind

You can remove a column or expression from the sorting specification by simply dragging it and releasing it outside the Columns box.

- 4 Click OK when you have specified all the sort columns and expressions.

Suppressing repeating values

When you sort on a column, there might be several rows with the same value in one column. You can choose to suppress the repeating values in that column.

When you suppress a repeating value, the value displays at the start of each new page and, if you are using groups, each time a value changes in a higher group.

For example, if you have sorted employees by department ID, you can suppress all but the first occurrence of each department ID in the DataWindow object:

Dept	ID	Name	Salary
300	390	Davidson , Jo Ann	\$57,090
	586	Coleman , James	\$42,300
	757	Higgins , Denis	\$43,700
	879	Coe , Kristen	\$36,500
	1293	Shea , Mary Anne	\$138,948
	1336	Bigelow , Janet	\$31,200
	1390	Litton , Jennifer	\$58,930
	1483	Leticq , John	\$75,400
400	184	Espinoza , Melissa	\$36,490
	207	Francis , Jane	\$53,870
	318	Crow , John	\$41,701
	409	Weaver , Bruce	\$46,550
	591	Barletta , Irene	\$45,450
	888	Charlton , Doug	\$28,300
	992	Butterfield , Joyce	\$34,011

❖ To suppress repeating values:

- 1 Select Rows>Suppress Repeating Values from the menu bar.

The Specify Repeating Value Suppression List dialog box displays:



- 2 Drag the columns whose repeated values you want to suppress from the Source Data box to the Suppression List box, and click OK.

If you change your mind

You can remove a column from the suppression list simply by dragging it and releasing it outside the Suppression List box.

Grouping rows

How groups are defined

You can group related rows together and, optionally, calculate statistics for each group separately. For example, you might want to group employee information by department and get total salaries for each department.

Each group is defined by one or more DataWindow object columns. Each time the value in a grouping column changes, a break occurs and a new section begins.

For each group, you can:

- Display the rows in each section
- Specify the information you want to display at the beginning and end of each section
- Specify page breaks after each break in the data
- Reset the page number after each break

Grouping example

The following DataWindow object retrieves employee information. It has one group defined, Dept_ID, so it groups rows into sections according to the value in the Dept_ID column. In addition, it displays:

- Department ID before the first row for that department
- Totals and averages for salary and salary plus benefits (a computed column) for each department
- Grand totals for the company at the end

The following screenshot shows the DataWindow object.

Total Compensation Report				Value of Health Ins. - \$4000 Value of Life Ins. - .543% of salary Value of Day Care - \$5,200			Page 4 of 4 2/27/2004		
Dept. ID	Employee ID	Employee First Name	Employee Last Name	Salary	Health Ins.	Life Ins.	Day Care	Salary Plus Benefits	
400	1191	Matthew	Bucceri	\$45,900.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$55,449.00	
	1507	Ruth	Wetherby	\$35,745.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$45,239.00	
	1576	Scott	Evans	\$68,940.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$78,614.00	
	1607	Mark	Morris	\$61,300.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$70,932.00	
	1643	Elizabeth	Lambert	\$29,384.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$38,843.00	
	1684	Janet	Hildebrand	\$45,829.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$55,377.00	
	1740	Robert	Nielsen	\$34,889.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$44,378.00	
	1751	Alex	Ahmed	\$34,992.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$44,482.00	
	Total:				\$698,250.75			Total:	\$850,842.25
	Average:				\$43,640.67			Average:	\$53,177.17
500	191	Jeannette	Bertrand	\$32,780.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$42,257.00	
	703	Jose	Martinez	\$61,050.88	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$70,681.88	
	750	Jane	Braun	\$37,730.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$47,234.00	
	868	Felicia	Kuo	\$28,200.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$37,653.00	
	921	Charles	Crowley	\$41,700.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$51,226.00	
	1013	Joseph	Barker	\$27,290.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$36,738.00	
	1570	Anthony	Rebeiro	\$34,576.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$44,063.00	
	1615	Sheila	Romero	\$27,500.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$36,949.00	
	1658	Michael	Lynch	\$24,903.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$34,338.00	
	Total:				\$315,729.88			Total:	\$401,144.29
Average:				\$35,081.10			Average:	\$44,571.10	
Grand Total:				\$3,761,106.82			Grand Total:	\$4,479,029.63	
Average:				\$50,148.09			Average:	\$59,719.93	

How to do it

You can create a grouped DataWindow object in three ways:

- Use the Group presentation style to create a grouped DataWindow object from scratch ("Using the Group presentation style" next).
- Take an existing tabular DataWindow object and define grouping ("Defining groups in an existing DataWindow object" on page 276).
- Use the TreeView presentation style (Chapter 13, "Working with TreeViews").

Making the DataWindow control large enough

If a DataWindow object has grouped rows, each page contains all group headers (including zero-height headers) at the top of the page. Your DataWindow control must be large enough to accommodate all the group headers that display on each page of the report.

The last row of a group displays on the same page as that row's group trailer and each applicable higher-level group trailer. If the DataWindow object has a summary band, it displays on the same page as the last row of the report. If the control is not large enough, you might see anomalies when scrolling through the DataWindow object, particularly in the last row of the report, which needs room to display the report's header band, all group headers, all group trailers, the summary band, and the footer band.

If you cannot increase the height of the DataWindow control so that it has room for all the headers and trailers, you can change the design of the DataWindow object so that they require less space.

Scrolling through a grouped DataWindow

When you scroll through a grouped DataWindow object, you might see the group header repeated where you do not expect it. This is because the data is paginated in a fixed layout based on the size of the DataWindow control. You can scroll to a point that shows the bottom half of one page and the top of the next. When you use the arrow keys to page through the data, you scroll one row at a time.

Using the Group presentation style

One of the DataWindow object presentation styles, Group, is a shortcut to creating a grouped DataWindow object. It generates a tabular DataWindow object that has one group level and some other grouping properties defined. You can then further customize the DataWindow object.

❖ To create a basic grouped DataWindow object using the Group presentation style:

- 1 Select File>New from the menu bar.

The New dialog box displays.

- 2 Choose the DataWindow tab page and the Group presentation style, and click OK.
- 3 Choose a data source and define the data.

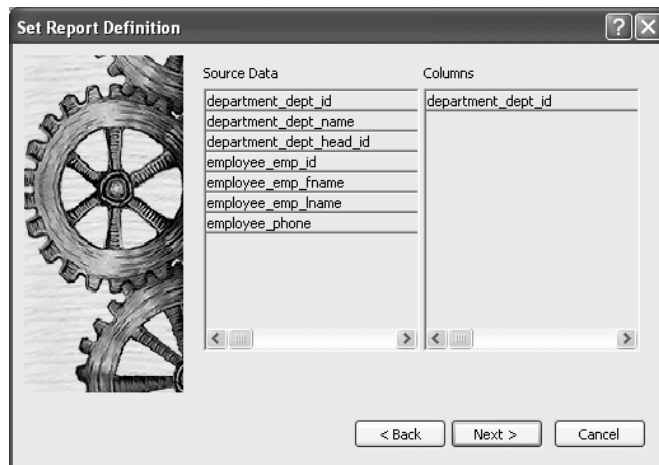
You are prompted to define the grouping column(s).

- 4 Drag the column(s) you want to group on from the Source Data box to the Columns box.

Multiple columns and multiple group levels

You can specify more than one column, but all columns apply to group level one. You can define one group level at this point. Later you can define additional group levels.

In the following example, grouping will be by department, as specified by the dept_id column:



If you want to use an expression, you can define it when you have completed the wizard. See “Using an expression for a group” on page 275.

- 5 Click Next.

DataWindow Designer suggests a header based on your data source. For example, if your data comes from the Employee table, DataWindow Designer uses the name Employee in the suggested header.
- 6 Specify the Page Header text.
- 7 If you want a page break each time a grouping value changes, select the New Page On Group Break box.
- 8 If you want page numbering to restart at 1 each time a grouping value changes, select the Reset Page Number On Group Break box *and* the New Page On Group Break box.
- 9 Click Next.

- 10 Select Color and Border settings and click Next.
- 11 Review your specification and click Finish.

The DataWindow object displays with the basic grouping properties set.

This is an example of a Group style DataWindow object:

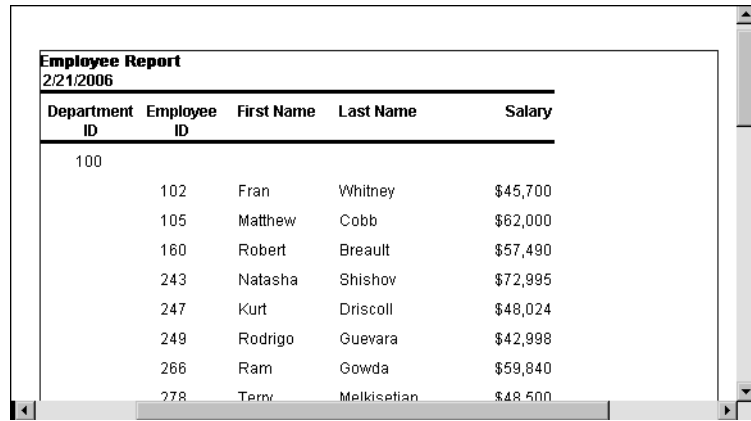
Employee Report				
today()				
Department ID	Employee ID	First Name	Last Name	Salary
Header ↑				
dept_id				
1: Header group dept_id ↑				
emp_id	emp_fname	emp_lname	salary	
Detail ↑				
Total for department:				sum(salary)
1: Trailer group dept_id ↑				
Grand Total:				sum(salary)
Summary ↑				
Page * + page() +				
Footer ↑				

What DataWindow Designer does

As a result of your specifications, DataWindow Designer generates a tabular DataWindow object and:

- Creates group header and trailer bands
- Places the column you chose as the grouping column in the group header band
- Sorts the rows by the grouping column
- Places the page header and the date (as a computed field) in the header band
- Places the page number and page count (as computed fields) in the footer band
- Creates sum-computed fields for all numeric columns (the fields are placed in the group trailer and summary bands)

Here is the preceding DataWindow object in the Preview view:



The screenshot shows a DataWindow object titled "Employee Report" with a date "2/21/2006". The table has five columns: Department ID, Employee ID, First Name, Last Name, and Salary. The data is grouped by Department ID, with a group header for Department ID 100. The rows are as follows:

Department ID	Employee ID	First Name	Last Name	Salary
100				
	102	Fran	Whitney	\$45,700
	105	Matthew	Cobb	\$62,000
	160	Robert	Breault	\$57,490
	243	Natasha	Shishov	\$72,995
	247	Kurt	Driscoll	\$48,024
	249	Rodrigo	Guevara	\$42,998
	266	Ram	Gowda	\$59,840
	278	Terry	Melkisejian	\$48,500

Using an expression for a group

If you want to use an expression for one or more column names in a group, you can enter an expression as the Group Definition on the General page in the Properties view after you have finished using the Group wizard.

❖ **To use an expression for a group:**

- 1 Open the Properties view and select the group header band in the Design view.
- 2 Double-click the ellipsis button next to the Group Definition box on the General page in the Properties view to open the Specify Group Columns dialog box.
- 3 In the Columns box, double-click the column that you want to use in an expression.

The Modify Expression dialog box opens. You can specify more than one grouping item expression for a group. A break occurs whenever the value concatenated from each column/expression changes.

What you can do

You can use any of the techniques available in a tabular DataWindow object to modify and enhance the grouped DataWindow object, such as moving controls, specifying display formats, and so on. In particular, see "Defining groups in an existing DataWindow object" next to learn more about the bands in a grouped DataWindow object and how to add features especially suited for grouped DataWindow objects (for example, add a second group level, define additional summary statistics, and so on).

DataWindow Object is not updatable by default

When you generate a DataWindow object using the Group presentation style, DataWindow Designer makes it not updatable by default. If you want to be able to update the database through the grouped DataWindow object, you must modify its update characteristics. For more information, see Chapter 6, “Controlling Updates in DataWindow Objects.”

Defining groups in an existing DataWindow object

Instead of using the Group presentation style to create a grouped DataWindow object from scratch, you can take an existing tabular DataWindow object and define groups in it.

❖ **To add grouping to an existing DataWindow object:**

- 1 Start with a tabular DataWindow object that retrieves all the columns you need.
- 2 Specify the grouping columns.
- 3 Sort the rows.
- 4 (Optional) Rearrange the DataWindow object.
- 5 (Optional) Add summary statistics.
- 6 (Optional) Sort the groups.

Steps 2 through 6 are described next.

Specifying the grouping columns

❖ **To specify the grouping columns:**

- 1 In the DataWindow painter, Select Rows>Create Group from the menu bar.

The Specify Group Columns dialog box displays.
- 2 Specify the group columns, as described in “Using the Group presentation style” on page 272.
- 3 Set the Reset Page Count and New Page on Group Break properties on the General page in the Properties view.

Creating subgroups

After defining your first group, you can define subgroups, which are groups within the group you just defined.

❖ **To define subgroups:**

- 1 Select Rows>Create Group from the menu bar and specify the column/expression for the subgroup.
- 2 Repeat step 1 to define additional subgroups if you want.

You can specify as many levels of grouping as you need.

How groups are identified

DataWindow Designer assigns each group a number (or level) when you create the group. The first group you specify becomes group 1, the primary group. The second group becomes group 2, a subgroup within group 1, and so on.

For example, suppose you define two groups. The first group uses the dept_id column and the second group uses the status column.

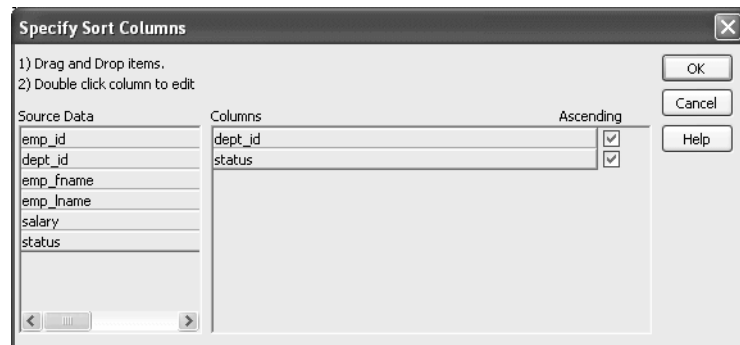
The rows are grouped first by department (group 1). Within department, rows are grouped by status (group 2). If you specify page breaks for the groups, a page break will occur when any of these values changes.

You use the group's number to identify it when defining summary statistics for the group. This is described in "Adding summary statistics" on page 279.

Sorting the rows

DataWindow Designer does not sort the data when it creates a group. Therefore, if the data source is not sorted, you must sort the data by the same columns (or expressions) specified for the groups.

For example, if you are grouping by dept_id then status, select Rows>Sort from the menu bar and specify dept_id and then status as sorting columns:



You can also sort on additional rows. For example, if you want to sort by employee ID within each group, specify `emp_id` as the third sorting column.

For more information about sorting, see “Sorting rows” on page 267.

Rearranging the DataWindow object

When you create a group, DataWindow Designer creates two new bands for each group:

- A group header band
- A group trailer band

The bar identifying the band contains:

- The number of the group
- The name of the band
- The name of each column that defines the group
- An arrow pointing to the band

Dept Id	Emp Id	Emp Fname	Emp Lname	Salary
Header ↑				
1: Header group dept_id ↑				
dept_id	emp_id	emp_fname	emp_lname	salary
Detail ↑				
1: Trailer group dept_id ↑				
Summary ↑				
Footer ↑				

You can include any control in the DataWindow object (such as columns, text, and computed fields) in the header and trailer bands of a group.

Using the group header band

The contents of the group header band display at the top of each page and after each break in the data.

Typically, you use this band to identify each group. You might move the grouping column from the detail band to the group header band, since it now serves to identify one group rather than each row.

For example, if you group the rows by department and include the department in the group header, the department will display before the first line of data each time the department changes.

At runtime, you see this:

Dept Id	Emp Id	Emp Fname	Emp Lname	Salary
100	1090	Susan	Smith	51411.00
	582	Peter	Samuels	37400.00
	529	Dorothy	Sullivan	67890.00
	479	Linda	Siperstein	39875.50
	266	Ram	Gowda	59840.00
	604	Albert	Wang	68400.00
	453	Andrew	Rabkin	64500.00
	445	Kim	Lull	87900.00

Suppressing group headers

If you do not want a group header to display at the top of each page when you print or display a report, select the Suppress Group Header check box on the General property page for the header. If none of the headers are suppressed, they all display at the top of each page. When a page break coincides with a group break, the group header and any group headers that follow it display even if the Suppress Group Header property is set, but higher level headers are suppressed if the property is set for those headers.

For example, suppose a report has three groups: division, sales region, and sales manager. If all three group headers are suppressed, and a sales region group break coincides with a page break, the division header is suppressed but the sales region and sales manager headers display.

Using the group trailer band

The contents of the group trailer display after the last row for each value that causes a break.

In the group trailer band, you specify the information you want displayed after the last line of identical data for each value in the group. Typically, you include summary statistics here, as described next.

Adding summary statistics

One of the advantages of creating a grouped DataWindow object is that you can have DataWindow Designer calculate statistics for each group. To do that, you place computed fields that reference the group. Typically, you place these computed fields in the group's trailer band.

❖ **To add a summary statistic:**

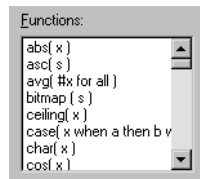
- 1 Select Insert>Control>Computed Field from the menu bar.
- 2 Click in the Design view where you want the statistic.
The Modify Expression dialog box displays.
- 3 Specify the expression that defines the computed field (see below).
- 4 Click OK.

A shortcut to sum values

If you want to sum a numeric column, select the column in Design view and click the Sum button in the Controls drop-down toolbar. DataWindow Designer automatically places a computed field in the appropriate band.

Specifying the expression

Typically, you use aggregate and other functions in your summary statistic. DataWindow Designer lists functions you can use in the Functions box in the Modify Expression dialog box. When you are defining a computed field in a group header or trailer band, DataWindow Designer automatically lists forms of the functions that reference the group:



You can paste these templates into the expression, then replace the #x that is pasted in as the function argument with the appropriate column or expression.

For example, to count the employees in each department (group 1), specify this expression in the group trailer band:

```
Count( Emp_Id for group 1 )
```

To get the average salary of employees in a department, specify:

```
Avg( Salary for group 1 )
```


To get the total salary of employees in a department, specify:

```
Sum( Salary for group 1 )
```

Employee ID	First Name	Last Name	Salary
Header ↑			
Department ID			
dept_id			
1: Header group dept_id ↑			
emp_id	emp_fname	emp_lname	salary
Detail ↑			
			Average Salary: avg(salary for
			Total Salary: sum(salary for
1: Trailer group dept_id ↑			
Summary ↑			
Footer ↑			

At runtime, you see this:

Employee ID	First Name	Last Name	Salary
Department ID			
500			
191	Jeannette	Bertrand	\$29,800
1013	Joseph	Barker	\$27,290
921	Charles	Crowley	\$41,700
868	Felicia	Kuo	\$28,200
1658	Michael	Lynch	\$24,903
1615	Sheila	Romero	\$27,500
750	Jane	Braun	\$34,300
1570	Anthony	Rebeiro	\$34,576
703	Jose	Martinez	\$55,501
Average Salary:			\$33,752
Total Salary:			\$303,770

Sorting the groups

You can sort the groups in a DataWindow object. For example, in a DataWindow object showing employee information grouped by department, you might want to sort the departments (the groups) by total salary.

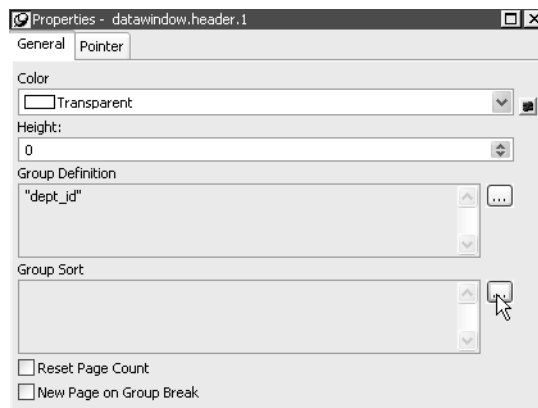
Typically, this involves aggregate functions, as described in “Adding summary statistics” on page 279. In the department salary example, you would sort the groups using the aggregate function Sum to calculate total salary in each department.

❖ **To sort the groups:**

- 1 Place the mouse pointer on the group header bar (not inside the band) until the pointer becomes a double-headed arrow.
- 2 Click.

The General property page for the group displays in the Properties view.

- 3 Click the button next to the Group Sort box.

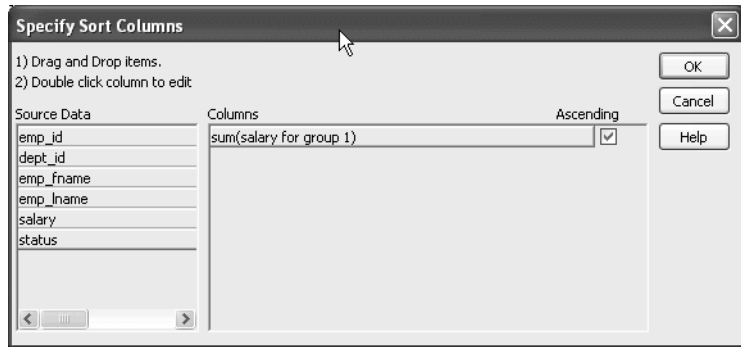


The Specify Sort Columns dialog box displays.

- 4 Drag the column you want to sort the groups by from the Source Data box into the Columns box.

If you chose a numeric column, DataWindow Designer uses the Sum function in the expression; if you chose a non-numeric column, DataWindow Designer uses the Count function.

For example, if you chose the Salary column, DataWindow Designer specifies that the groups will be sorted by the expression `sum(salary for group 1)`:



- 5 Select ascending or descending sort as appropriate.
- 6 If you want to modify the expression to sort on, double-click the column in the Columns box.

The Modify Expression dialog box displays.

- 7 Specify the expression to sort on.

For example, to sort the department group (the first group level) on average salary, specify `avg(salary for group 1)`.

- 8 Click OK.

You return to the Specify Sort Columns dialog box with the expression displayed.

- 9 Click OK again.

At runtime, the groups will be sorted on the expression you specified.

Highlighting Information in DataWindow Objects

About this chapter

This chapter describes how you modify the way information displays in DataWindow objects and reports based on the conditions you specify. The conditions are usually related to data values, which are not available until runtime.

Contents

Topic	Page
Highlighting information	285
Modifying properties conditionally at runtime	289
Supplying property values	295
Specifying colors	314

Highlighting information

Every control in a DataWindow object has a set of properties that determines what the control looks like and where it is located. For example, the values in a column of data display in a particular font and color, in a particular location, with or without a border, and so on.

Modifying properties when designing

You define the appearance and behavior of controls in DataWindow objects in the DataWindow painter. As you do that, you are specifying the controls' properties. For example, when you place a border around a column, you are setting that column's Border property.

In most cases, the appearance and behavior of controls is fixed; you do not want them to change at runtime. When you make headings bold when designing them, you want them to be bold at all times.

In the following DataWindow object, the Salary Plus Benefits column has a Shadow box border around every data value in the column. To display the border, you set the border property for the column:

Health Ins.	Life Ins.	Day Care	Salary Plus Benefits
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$50,748
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$67,137
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$67,802
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$78,191
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$58,284
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$48,031
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$60,165
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$58,763
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$84,905
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$93,177
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$69,650
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$44,892

Modifying properties at runtime

In some cases, however, you might want some properties of controls in DataWindow objects to be driven by the data, which is not known when you are defining the DataWindow object in the painter. For these situations you can define property conditional expressions, which are expressions that are evaluated at runtime.

You can use these expressions to conditionally and dynamically modify the appearance and behavior of your DataWindow object at runtime. The results of the expressions set the values of properties of controls in the DataWindow object.

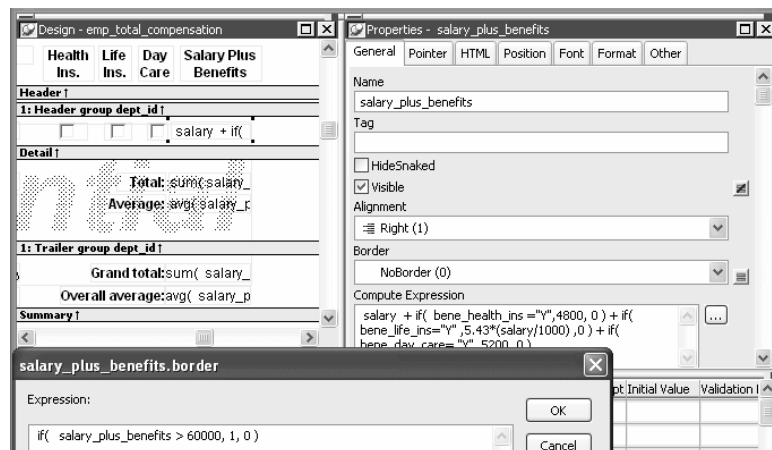
In the following DataWindow object, the Salary Plus Benefits column has a Shadow box border highlighting each data value that is greater than \$60,000:

Health Ins.	Life Ins.	Day Care	Salary Plus Benefits
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$50,748
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$67,137
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$67,802
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$78,191
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$58,284
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$48,031
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$60,165
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$58,763
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$84,905
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$93,177
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$69,650
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$44,892

To control the display of the border, you define a property conditional expression for the column's Border property. When users run the DataWindow object, DataWindow Designer changes the border of individual data values based on the condition (value greater than \$60,000).

Defining an expression

The following illustration shows the Salary_Plus_Benefits column selected in the Design view. To the right of the Design view, the Properties view shows properties for the column, including the Border property. Next to the Border property is a button for accessing the dialog box where you enter the expression. The button displays an equals sign with a slash through it when no expression has been entered, and an equals sign without a slash when it has.



In this example the Border property is set to NoBorder in the Properties view. However, the expression defined for the property overrides that setting at runtime.

A closer look at the expression

The expression you enter almost always begins with `IF`. Then you specify three things: the condition, what happens if it is true, and what happens if it is false. Parentheses surround the three things and commas separate them:

`IF(expression, true, false)`

The following expression is used in the example. Because the expression is for the Border property, the values for true and false indicate particular borders. The value 1 means Shadow box border and the value 0 means no border:

`IF(salary_plus_benefits > 60000, 1, 0)`

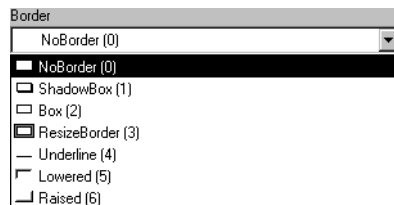
When users run the DataWindow object, DataWindow Designer checks the value in the computed column called salary_plus_benefits to see if it is greater than 60,000. If it is (true), DataWindow Designer displays the value with the Shadow box border. If not (false), DataWindow Designer displays the value with no border.

About specifying properties

Usually you specify a number to indicate what you want for a particular property. For example, the following list shows all of the borders you can specify and the numbers you use. If you want the border property to be Shadow box, you specify 1 in the If statement, for either true or false.

- 0—None
- 1—Shadow box
- 2—Box
- 3—Resize
- 4—Underline
- 5—3D Lowered
- 6—3D Raised

In the Properties view, the list of choices for setting a property includes the values that correspond to choices in parentheses. This makes it easier to define an expression for a property; you do not need to look up the values. For example, if you want to specify the ResizeBorder in your expression, you use the number 3, as shown in the drop-down list.



For details on the values of properties that can be set using expressions, see “Supplying property values” on page 295.

For complete information about what the valid values are for all properties associated with a DataWindow object, see the discussion of DataWindow object properties in the *DataWindow Object Reference* or online Help.

About modifying properties programmatically

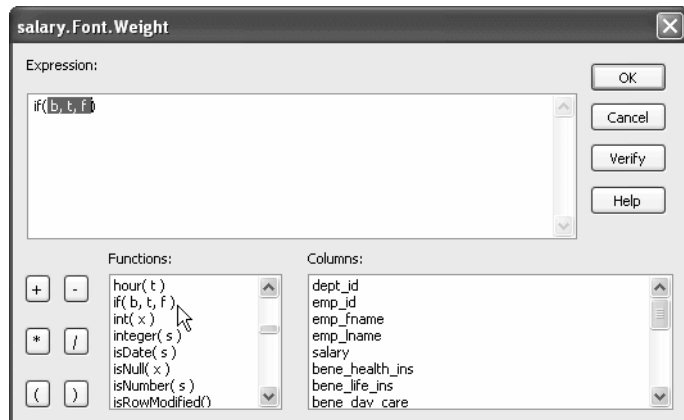
You can also programmatically modify the properties of controls in a DataWindow object at runtime. For more information, see the *Programmer’s Guide*.

Modifying properties conditionally at runtime

“Modifying properties at runtime” on page 286 described how you can use conditional expressions that are evaluated at runtime to highlight information in a DataWindow object. This section presents a procedure for modifying properties at runtime and some examples.

❖ **To modify properties conditionally at runtime:**

- 1 Position the pointer on the control, band, or DataWindow object background whose properties you want to modify at runtime.
- 2 Select Properties from the pop-up menu, then select the page that contains the property you want to modify at runtime.
- 3 Click the button next to the property you want to change.
- 4 Scroll the list of functions in the Functions box until you see the IF function, and then select it:



- 5 Replace the *b* (boolean) with your condition (for example, `salary>40000`).
You can select columns and functions and use the buttons to add the symbols shown on them.
- 6 Replace the *t* (true) with the value to use for the property if the condition is true.

Values to use for properties are usually numbers. They are different for each property. For more information about property values that can be set on the Expressions page, see “Supplying property values” on page 295.

For complete information about what the valid values are for all properties of controls in the DataWindow object, see the discussion of DataWindow object properties in the *DataWindow Object Reference* or online Help.

- 7 Replace the *f* (false) with the value to use for the property if the condition is false.
- 8 Click OK.

For examples, see "Example 1: creating a gray bar effect" next, "Example 2: rotating controls" on page 291, "Example 3: highlighting rows of data" on page 292, and "Example 4: changing the size and location of controls" on page 294.

Example 1: creating a gray bar effect

The following DataWindow object shows alternate rows with a light gray bar. The gray bars make it easier to track data values across the row:

Department ID	Employee ID	Employee First Name	Employee Last Name	Salary	Health Ins.	Life Ins.	Day Care	Salary Plus Benefits
100	102	Fran	Whitney	\$45,700	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$50,748
	105	Matthew	Cobb	\$62,000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$67,137
	160	Robert	Breault	\$57,490	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$67,802
	243	Natasha	Shishov	\$72,995	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$78,191
	247	Kurt	Driscoll	\$48,024	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$58,284
	249	Rodrigo	Guevara	\$42,998	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$48,031

Total Compensation Report
Salary Plus Benefits

Value of health ins. = \$4,800
 Value of life insurance = \$(5.43 x salary)/1,000
 Value of day care = \$5,200

Page 1 of 15
 03/22/03

To create the gray bar effect:

- 1 Add a rectangle control to the detail band and size it so that it surrounds the controls you want highlighted.

To make sure that you have selected the detail band, select the Position tab in the Properties view and select Band from the Layer drop-down list.
- 2 To make it easier to see what you are doing in the Design view, select the General tab and set the Brush Color to White and the Pen Color to Black. A narrow black line forms a boundary around the rectangle.
- 3 Select Send to Back from the rectangle's pop-up menu.
- 4 To hide the border of the rectangle, set the Pen Style to No Visible Line.

- 5 Click the button next to the Brush Color property on the General page.
- 6 In the Modify Expression dialog box, enter the following expression for the Brush.Color property:

```
If (mod (getrow () , 2 ) = 1 , rgb (255 , 255 , 255 ) , rgb (240 , 240 , 240 ))
```

The mod function takes the row number (`getrow()`), divides it by 2, then returns the remainder. The remainder can be either 0 or 1. If the row number is odd, mod returns 1; if the row number is even, mod returns 0.

The expression `mod (getrow () , 2) = 1` distinguishes odd rows from even rows.

The rgb function specifies maximum amounts of red, green, and blue: `rgb (255 , 255 , 255)`. Specifying 255 for red, green, and blue results in the color white.

If the row number is odd (the condition evaluates as true), the rectangle displays as white. If the row number is even (the condition evaluates as false), the rectangle displays as light gray (`rgb (240 , 240 , 240)`).

Example 2: rotating controls

The following DataWindow object shows the column headers for Health Insurance, Life Insurance, and Day Care rotated 45 degrees.

Total Compensation Report Salary Plus Benefits						Page 1 of 19 03/22/03		
Department ID	Employee ID	Employee First Name	Employee Last Name	Salary	Health Ins.	Life Ins.	Day Care	Salary Plus Benefits
	100	Fran	Whitney	\$45,700	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$50,748
	105	Matthew	Cobb	\$62,000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$67,137
	160	Robert	Breault	\$57,490	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$67,802
	243	Natasha	Shishov	\$72,995	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$78,191
	247	Kurt	Driscoll	\$48,024	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$58,284

To rotate each of these three text controls:

- 1 Select one of the controls, then use Ctrl + click to select the other two controls.

The Properties view changes to show the properties that are common to all selected controls.

- 2 On the Font page in the Properties view, click the button next to the Escapement property.
- 3 Enter the number 450 in the Modify Expression dialog box and click OK.

The value entered for font escapement is in tenths of degrees, so the number 450 means 45 degrees. You do not have to specify a condition. Typically, you do not specify a condition for control rotation.

The rotation of the controls does not change in the Design view.

- 4 To see the change, close and reopen the Preview view.

Example 3: highlighting rows of data

The following DataWindow object is an employee phone list for a company in Massachusetts. Out-of-state (not in Massachusetts) employees are shown in bold and preceded by two asterisks (**):

<i>(** means out of state)</i>		Employee Phone List	
		08/20/04	
Employee Name	Phone	Employee Name	Phone
Ahmed, Alex	(617) 555-8748	Pickett, Catherine	(617) 555-3478
Barker, Joseph	(617) 555-8021	Poitras, Kathleen	(617) 555-3920
Barletta, Irene	(617) 555-8345	Powell, Thomas	(617) 555-1956
Bertrand, Jeannette	(508) 555-8138	Preston, Mark	(617) 555-5862
Bigelow, Janet	(617) 555-1493	Rabkin, Andrew	(617) 555-4444
Blaikie, Barbara	(617) 555-9345	Rebeiro, Anthony	(617) 555-5737
Braun, Jane	(617) 555-7857	Romero, Sheila	(617) 555-8138
Breault, Robert	(617) 555-3099	Samuels, Peter	(617) 555-8342
Bucceri, Matthew	(617) 555-5336	** Savarino, Pamela	(310) 555-1857
Butterfield, Joyce	(617) 555-2232	Scott, David	(617) 555-3246
Chao, Shih Lin	(617) 555-5921	Shea, Mary Anne	(617) 555-4616
Charlton, Doug	(508) 555-9246	** Sheffield, John	(713) 555-3877
** Chin, Philip	(404) 555-2341	Shishov, Natasha	(617) 555-2755

This DataWindow object uses newspaper columns. To understand how to create this DataWindow object without highlighting data, see “Printing with newspaper-style columns” on page 176.

In the Design view, the detail band includes four controls: the employee last name, a comma, the employee first name, and the phone number:

Header ↑
emp_lname , emp_fname phone
Detail ↑

To make these controls display in bold with two asterisks if the employee is not from Massachusetts:

- 1 Select one of the controls, then use Ctrl + Click to select the other three controls.

The Properties view changes to show the properties that are common to all selected controls.

- 2 On the Font page in the Properties view, click the button next to the Bold property.
- 3 Enter the following expression in the Modify Expression dialog box and click OK:

```
If (state = 'MA', 400, 700)
```

The expression states that if the value of the state column is MA, use 400 as the font weight. This means employees from Massachusetts display in the normal font. For any state except MA, use 700 as the font weight. This means all other employees display in bold font.

Logic that relies on the state column

To use logic that relies on the state column, you need to include the column in the data source. You can add the column after creating the DataWindow object by modifying the data source. Notice that the state column does not actually appear anywhere in the DataWindow object. Values must be available but do not need to be included in the DataWindow object.

- 4 To insert two asterisks (**) in front of the employee name if the employee is not from Massachusetts, add a text control to the left of the employee name with the two asterisks in bold.
- 5 With the text control selected, click the button next to its Visible property on the General page in the Properties view.
- 6 In the Modify Expression dialog box that displays, enter the following expression and click OK:

```
If (state = 'MA', 0, 1)
```

This expression says that if the state of the employee is MA (the true condition), the Visible property of the ** control is off (indicated by 0). If the state of the employee is not MA (the false condition), the Visible property of the ** control is on (indicated by 1). The asterisks are visible next to that employee's name.

Tip

You can use underlines, italics, strikethrough, borders, and colors to highlight information.

Example 4: changing the size and location of controls

The following DataWindow object shows city and state columns enclosed in a rectangle and underlined. The columns change location if the current row contains data for a customer from the state of New York. The rectangle and the line change both location and size.

Customers from New York An example of changing the size and location of objects

Customer ID	Name	Address	City	State
101	Michaels Devlin	3114 Pioneer Avenue	Rutherford	NJ
102	Beth Reiser	1033 Whippany Road		New York NY
103	Erin Niedringhaus	1990 Windsor Street	Paoli	PA
104	Meghan Mason	550 Dundas Street East	Knoxville	TN
105	Laura McCarthy	1210 Highway 36	Carmel	IN

This example shows how to move the rectangle and line. The process for columns is similar.

In the Design view, the rectangle and line display in one location, with a single set of dimensions. The expressions you specify are used only in Preview view and at runtime and all have the following syntax:

If (state='NY', true value, false value)

The *false value* is the same as the value in Design view. All of the values used in this example are in PowerBuilder Units (PBUs), the default unit of measure used for the DataWindow object.

To change properties of the rectangle and the line for rows with the state column equal to New York:

- 1 Select the rectangle, display the Position page in the Properties view, and specify expressions for the following properties:

Property	Expression
X	<code>if (state = 'NY', 2890, 1865)</code>
Width	<code>if (state = 'NY', 500, 1000)</code>
Height	<code>if (state = 'NY', 160, 120)</code>

- 2 Select the line, display the Position page in the Properties view, and specify expressions for the following properties:

Property	Expression
X1	<code>if (state = 'NY', 2890, 1865)</code>
Y1	<code>if (state = 'NY', 168, 132)</code>
X2	<code>if (state = 'NY', 3400, 2865)</code>
Y2	<code>if (state = 'NY', 168, 132)</code>

- 3 On the General page for the line, specify this expression for Pen Width:

```
if (state = 'NY', 10, 4)
```

At runtime, the rectangle is taller and narrower, and the line is shorter and has a wider pen width.

Supplying property values

Each property has its own set of property values that you can use to specify the true and false conditions in the If expression. Usually you specify a number to indicate what you want. For example, if you are working with the Border property, you use the number 0, 1, 2, 3, 4, 5, or 6 to specify a border.

Table 10-1 summarizes the properties available. A detailed description of each property follows the table. For a complete list of properties for each control, see the online Help.

Valid values of properties are shown in parentheses in the Properties view wherever possible.

For example, the drop-down list showing border selections includes the correct number for specifying each border in parentheses after the name of the border (ShadowBox (1), Underline (4)).

Table 10-1: Properties for controls in the DataWindow painter

Property	Painter option in Properties view	Description
Background.Color	Background Color on General page or Font page	Background color of a control
Border	Border on General page	Border of a control
Brush.Color	Brush Color on General page	Color of a graphic control
Brush.Hatch	Brush Hatch on General page	Pattern used to fill a graphic control
Color	Text Color on Font page; Color on General page; Line Color on General page	Color of text for text controls, columns, and computed fields; background color for the DataWindow object; line color for graphs
Font.Escapement (for rotating controls)	Escapement on Font page	Rotation of a control
Font.Height	Size on Font page	Height of text
Font.Italic	Italic on Font page	Use of italic font for text
Font.Strikethrough	Strikeout on Font page	Use of strikethrough for text
Font.Underline	Underline on Font page	Use of underlining for text
Font.Weight	Bold on Font page	Weight (for example, bold) of text font
Format	Format on Format page	Display format for columns and computed fields
Height	Height on Position page	Height of a control
Pen.Color	Pen Color on General page	Color of a line or the line surrounding a graphic control
Pen.Style	Pen Style on General page	Style of a line or the line surrounding a graphic control
Pen.Width	Pen Width on General page	Width of a line or the line surrounding a graphic control
Pointer	Pointer on Pointer page	Image to be used for the pointer
Protect	Protect on General page	Whether a column can be edited
Timer.Interval	Timer Interval on General page	How often time fields are to be updated
Visible	Visible on General page	Whether a control is visible
Width	Width on Position page	Width of a control
X	X on Position page	X position of a control

Property	Painter option in Properties view	Description
X1, X2	X1, X2 on Position page	X coordinates of either end of a line
Y	Y on Position page	Y position of a control relative to the band in which it is located
Y1, Y2	Y1, Y2 on Position page	Y coordinates of either end of a line

Background.Color

Description	Setting for the background color of a control.
In the painter	Background Color on the General page or Font page in the Properties view.
Value	<p>A number that specifies the control's background color.</p> <p>For information on specifying colors, see “Specifying colors” on page 314.</p> <p>The background color of a line is the color that displays between the segments of the line when the pen style is not solid.</p> <p>If Background.Mode is transparent (1), Background.Color is ignored.</p>
Example	<p>The following statement specifies that if the person represented by the current row uses the day care benefit, the background color of the control is set to light gray (15790320). If not, the background color is set to white (16777215):</p> <pre>IF (bene_day_care = 'Y', 15790320, 16777215)</pre> <p>In this example, the condition is applied to the Background.Color property for three controls: the emp_id column, the emp_fname column, and the emp_lname column.</p>

The following is a portion of the resulting DataWindow object. Notice that the employee ID, first name, and last name have a gray background if the employee uses the day care benefit:

Employee ID	Employee First Name	Employee Last Name	Salary	Health Ins.	Life Ins.	Day Care	Salary Plus Benefits
102	Fran	Whitney	\$45,700	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$50,748
105	Matthew	Cobb	\$62,000	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$67,137
180	Robert	Breault	\$57,490	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$67,802
243	Natasha	Shishov	\$72,995	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$78,191
247	Kurt	Driscoll	\$48,024	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$58,284
249	Rodrigo	Guevara	\$42,998	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$48,031
266	Ram	Gowda	\$59,840	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$60,165
278	Terry	Melkisetian	\$48,500	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$58,763
316	Lynn	Pastor	\$74,500	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	\$84,905
445	Kim	Lull	\$87,900	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	\$93,177

Border

Description

The type of border for the control.

In the painter

Border on the General page in the Properties view.

Value

A number that specifies the type of border. Values are:

- 0—None
- 1—Shadow box
- 2—Box
- 3—Resize
- 4—Underline
- 5—3D Lowered
- 6—3D Raised

Example

The following statement specifies that if the person represented by the current row has a status of L (on leave), the status column displays with a Shadow box border:

```
If(status = 'L', 1, 0)
```

In this example, the condition is applied to the Border property of the status column.

The following is a portion of the resulting DataWindow object. Notice that the status On Leave displays with a Shadow box border:

Emp ID	Employee First Name	Employee Last Name	Street	City	State	Zip Code	Phone	Status	Soc. Sec. No.
102	Fran	Wahnev	49 East Washington Street	Needham	MA	02192-	(617) 555-3985	Active	017-34-9033
105	Mathew	Cobb	77 Pleasant Street	Waltham	MA	02154-	(617) 555-3840	Active	052-34-5739
129	Philip	Chin	59 Pond Street	Atlanta	GA	30339-	(404) 555-2341	Active	024-60-8923
148	Julie	Jordan	144 Great Plain Avenue	Winchester	MA	01890-	(617) 555-7835	Active	501-70-4733
160	Robert	Breault	58 Cherry Street	Milton	MA	02186-	(617) 555-3099	Active	025-48-7623
184	Melissa	Espinosa	112 Apple Tree Way	Stow	MA	01775-	(508) 555-2319	Active	025-48-1943
191	Jeanette	Berband	209 Concord Street	Acton	MA	01720-	(508) 555-8138	Active	017-34-8821
195	Marc	Dill	89 Hancock Street	Milton	MA	02186-	(617) 555-2144	Active	079-48-6634
207	Jane	Francis	12 Hawthorne Drive	Concord	MA	01742-	(508) 555-9022	Active	501-70-8992
243	Natasha	Shishov	15 Milk Street	Waltham	MA	02154-	(617) 555-2755	Active	043-21-6799
247	Kurt	Driscoll	154 School Street	Waltham	MA	02154-	(617) 555-1234	On Leave	024-60-1768
249	Rodrigo	Guevara	East Main Street	Frammingham	MA	01701-	(508) 555-0029	Active	084-32-9990
266	Ram	Gowda	79 Page Street	Natick	MA	01760-	(508) 555-8722	Active	017-34-6122

About the value L and the value On Leave

The status column uses an edit style. The internal value for on leave is L and the display value is On Leave. The conditional expression references the internal value L, which is the actual value stored in the database. The DataWindow object shows the value On Leave, which is the display value assigned to the value L in the code table for the Status edit style.

Brush.Color

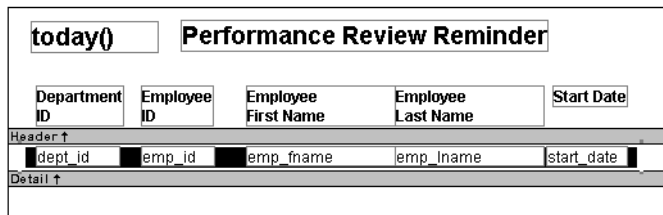
Description	Setting for the fill color of a graphic control.
In the painter	Brush Color on the General page in the Properties view.
Value	A number that specifies the color that fills the control. For information on specifying colors, see “Specifying colors” on page 314.
Example	See the example for "Brush.Hatch" next.

Brush.Hatch

- Description** Setting for the fill pattern of a graphic control.
- In the painter** Brush Hatch on the General page in the Properties view.
- Value** A number that specifies the pattern that fills the control. Values are:
- 0—Horizontal
 - 1—Bdiagonal (lines from lower left to upper right)
 - 2—Vertical
 - 3—Cross
 - 4—Fdiagonal (lines from upper left to lower right)
 - 5—DiagCross
 - 6—Solid
 - 7—Transparent

Example In this example, statements check the employee's start date to see if the month is the current month or the month following the current month. Properties of a rectangle control placed behind the row of data are changed to highlight employees with months of hire that match the current month or the month following the current month.

The Design view includes columns of data and a rectangle behind the data. The rectangle has been changed to black in the following picture to make it stand out:



The following statement is for the Brush.Color property of the rectangle. If the month of the start date matches the current month or the next one, Brush.Color is set to light gray (12632256). If not, it is set to white (16777215), which means it will not show:

```

If(month( start_date ) = month(today())
or month( start_date ) = month(today())+1
or (month(today()) = 12 and month(start_date)=1) ,
12632256, 16777215)
    
```

The following statement is for the Brush.Hatch property of the rectangle. If the month of the start date matches the current month or the next one, Brush.Hatch is set to Bdiagonal (1). If not, it is set to Transparent (7), which means it will not show:

```
If (month( start_date ) = month(today())  
or month( start_date ) = month(today())+1  
or (month(today()) = 12 and month(start_date)=1) ,  
1, 7)
```

Expressions are also provided for Pen.Color and Pen.Style.

For more about these properties and a picture, see “Pen.Style” on page 308.

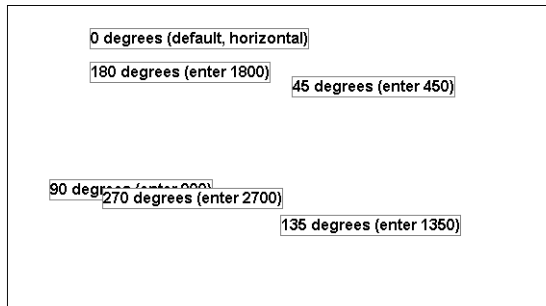
Color

Description	The color of text for text controls, columns, and computed fields; background color for the DataWindow object; line color for graphs.
In the painter	In the Properties view, Text Color on the Font property page; Color on the General property page; Line Color on the General property page.
Value	A number that specifies the color used for text. For information on specifying colors, see “Specifying colors” on page 314.
Example	The following statement is for the Color property of the emp_id, emp_fname, emp_lname, and emp_birth_date columns: <pre>If (month(birth_date) = month (today()), 255, 0)</pre> If the employee has a birthday in the current month, the information for the employee displays in red (255). Otherwise, the information displays in black (0). The Font.Underline property has the same conditional expression defined for it so that the example shows clearly on paper when printed in black and white.

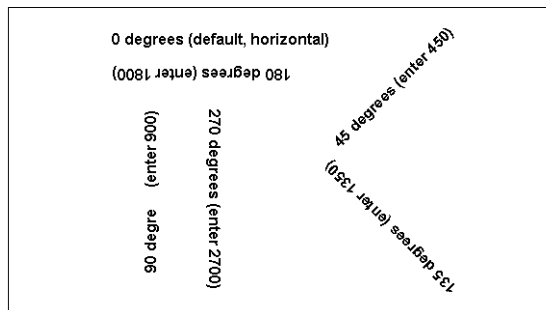
Font.Escapement (for rotating controls)

Description	The angle of rotation from the baseline of the text.
In the painter	Escapement on the Font page in the Properties view.
Value	An integer in tenths of degrees. For example, 450 means 45 degrees. 0 is horizontal.
Example	To enter rotation for a control, select the control in the Design view and click the button next to the Escapement property in the Properties view. In the dialog box that displays, enter the number of tenths of degrees.

The following picture shows the Design view with a number of text controls. Each text control shows the Font.Escapement value entered and the number of degrees of rotation. In the Design view, you do not see rotation; it looks as if the controls are all mixed up. Two controls seem to overlie each other:



The next picture shows the same controls at runtime. Each control is rotated appropriately:



How to position controls that are rotated

Make the controls movable. To do so, display each control and select the Moveable check box in the Position page. Then in the Preview view, click the rotated text control until a gray box displays (try the center of the text). Drag the rotated control where you want it. In the Design view, the controls will be wherever you dragged them. They may look incorrectly positioned in the Design view, but they will be correctly positioned when you run the DataWindow object. When you are satisfied with the positioning, you can clear the Moveable check box for the controls to ensure that they stay where you want them.

Font.Height

Description	The height of the text.
In the painter	Size on the Font page in the Properties view.
Value	An integer in the unit of measure specified for the DataWindow object. Units of measure include PowerBuilder units, thousandths of an inch (1000 = 1 inch), thousandths of a centimeter (1000 = 1 centimeter), or pixels. To specify size in points, specify a negative number.
Example	<p>The following statement is specified for the Font.Height property of a text control. Note that the DataWindow object is defined as using thousandths of an inch as its unit of measure. The statement says that if the control is in the first row, show the text 1/2-inch high (500 1/1000ths of an inch) and if it is not the first, show the text 1/5-inch high (200 1/1000ths of an inch):</p> <pre>If(GetRow() = 1, 500, 200)</pre> <p>The boundaries of the control might need to be extended to allow for the increased size of the text. At runtime, the first occurrence of the text control is big (1/2 inch); subsequent ones are small (1/5 inch).</p>

Font.Italic

Description A number that specifies whether the text should be italic.

In the painter Italic on the Font page in the Properties view.

Value Values are:

0—Not italic

1—Italic

Example The following statements are specified for the Font.Italic, Font.Underline, and Font.Weight properties, respectively. If the employee has health insurance, the employee's information displays in italics. If not, the employee's information displays in bold and underlined:

```
If (bene_health_ins = 'Y', 1, 0)
```

```
If (bene_health_ins = 'N', 1, 0)
```

```
If (bene_health_ins = 'N', 700, 400)
```

Statements are specified in this way for four controls: the emp_id column, the emp_fname column, the emp_lname column, and the emp_salary column. In the resulting DataWindow object, those with health insurance display in italics. Those without health insurance are emphasized with bold and underlining:

				Health insurance		
129	<i>Phillip</i>	<i>Chin</i>	<i>\$38,500.00</i>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
195	<i>Marc</i>	<i>Dill</i>	<i>\$54,800.00</i>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
299	<i>Rollin</i>	<i>Overbey</i>	<i>\$39,300.00</i>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
467	<i>James</i>	<i>Klobucher</i>	<i>\$49,500.00</i>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
641	<i>Thomas</i>	<i>Powell</i>	<i>\$54,600.00</i>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
667	Mary	Garcia	\$39,800.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
690	Kathleen	Poitras	\$46,200.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
856	<i>Samuel</i>	<i>Singer</i>	<i>\$34,892.00</i>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
902	<i>Moira</i>	<i>Kelly</i>	<i>\$87,500.00</i>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
913	Ken	Martel	\$55,700.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
930	<i>Ann</i>	<i>Taylor</i>	<i>\$46,890.00</i>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Font.Strikethrough

Description A number that specifies whether the text should be crossed out.

In the painter Strikeout on the Font page in the Properties view.

Value Values are:

- 0—Not crossed out
- 1—Crossed out

Example The following statement is for the Font.Strikethrough property of the emp_id, emp_fname, emp_lname, and emp_salary columns. The status column must be included in the data source even though it does not appear in the DataWindow object itself. The statement says that if the employee's status is L, which means On Leave, cross out the text in the control:

```
If(status = 'L', 1, 0)
```

An extra text control is included to the right of the detail line. It becomes visible only if the status of the row is L (see “Visible” on page 311).

The following is a portion of the resulting DataWindow object. It shows two employees who are On Leave. The four columns of information show as crossed out:

102	Fran	Whitney	\$45,700.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
105	Matthew	Cobb	\$62,000.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
160	Robert	Breault	\$57,490.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
243	Natasha	Shishov	\$72,995.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
247	Kurt	Driscoll	\$48,023.69	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<i>On leave</i>
249	Rodrigo	Guevara	\$42,998.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
266	Ram	Gowda	\$59,840.00	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
278	Terry	Melkisetian	\$48,500.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
316	Lynn	Pastor	\$74,500.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	
445	Kim	Lull	\$87,900.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
453	Andrew	Rabkin	\$64,500.00	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	
479	Linda	Siperstein	\$39,875.50	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<i>On leave</i>

Font.Underline

Description	A number that specifies whether the text should be underlined.
In the painter	Underline on the Font page in the Properties view.
Value	Values are: 0—Not underlined 1—Underlined
Example	The following statement, when applied to the Font.Underline property of columns of employee information, causes the information to be underlined if the employee does not have health insurance: <pre>If(bene_health_ins = 'N', 1, 0)</pre> For pictures of this example, see “Font.Italic” on page 304.

Font.Weight

Description	The weight of the text.
In the painter	Bold on the Font page in the Properties view.
Value	Values are: 100—Thin 200—Extra light 300—Light 400—Normal 500—Medium 600—Semibold 700—Bold 800—Extrabold 900—Heavy

Most commonly used values

The most commonly used values are 400 (Normal) and 700 (Bold). Your printer driver might not support all of the settings.

Example	The following statement, when applied to the Font.Weight property of columns of employee information, causes the information to be displayed in bold if the employee does not have health insurance: <pre>If(bene_health_ins = 'N', 700, 400)</pre> For pictures of this example, see “Font.Italic” on page 304.
---------	---

Format

Description	The display format for a column.
In the painter	Format on the Format page in the Properties view.
Values	A string specifying the display format.
Example	The following statement, when applied to the Format property of the Salary column, causes the column to display the word <i>Overpaid</i> for any salary greater than \$60,000 and <i>Underpaid</i> for any salary under \$60,000:

```
if (salary>60000, 'Overpaid', 'Underpaid')
```

Edit Mask edit style change

The Edit Mask edit style assigned to the salary column had to be changed. Because edit styles take precedence over display formats, it was necessary to change the edit style assigned to the salary column (an Edit Mask edit style) to the Edit edit style.

Height

Description	The height of the column or other control.
In the painter	Height on the Position page in the Properties view.
Value	An integer in the unit of measure specified for the DataWindow object. Units of measure include PowerBuilder units, thousandths of an inch (1000 = 1 inch), thousandths of a centimeter (1000 = 1 centimeter), or pixels.
Example	The following statement causes the height of a rectangle to be 160 PowerBuilder units if the state column for the row has the value NY. Otherwise, the rectangle is 120 PowerBuilder units high:

```
if (state = 'NY', 160, 120)
```

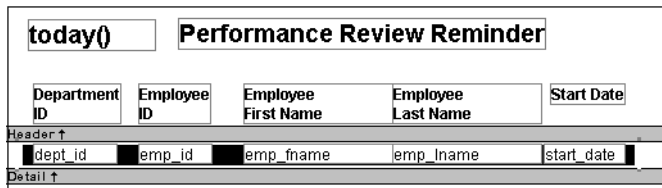
For more details and pictures, see “Example 4: changing the size and location of controls” on page 294.

Pen.Color

Description	The color of the line or the outline of a graphic control.
In the painter	Pen Color on the General page in the Properties view.
Value	A number that specifies the color of the line or outline. For information on specifying colors, see “Specifying colors” on page 314.
Example	See the example for the Pen.Style property, next.

Pen.Style

Description	The style of the line or the outline of a graphic control.
In the painter	Pen Style on the General page in the Properties view.
Value	Values are: <ul style="list-style-type: none"> 0—Solid 1—Dash 2—Dotted 3—Dash-dot pattern 4—Dash-dot-dot pattern 5—Null (no visible line)
Example	<p>In this example, statements check the employee's start date to see if the month is the current month or the month following the current month. Properties of a rectangle control placed behind the row of data are changed to highlight employees with months of hire that match the current month or the month following the current month.</p> <p>The Design view includes columns of data and a rectangle behind the data. The rectangle has been changed to black in the following picture to make it stand out:</p>



The following statement is for the Pen.Color property of the line around the edge of the rectangle. If the month of the start date matches the current month or the next one, Pen.Color is set to light gray (12632256). If not, it is set to white (16777215), which means it will not show:

```
If (month( start_date ) = month(today())
or month( start_date ) = month(today())+1
or (month(today()) = 12 and month(start_date)=1) ,
12632256, 16777215)
```

The following statement is for the Pen.Style property of the rectangle. If the month of the start date matches the current month or the next one, Pen.Style is set to Solid (0). If not, it is set to NULL (5), which means it will not show:

```
If (month( start_date ) = month(today())
or month( start_date ) = month(today())+1
or (month(today()) = 12 and month(start_date)=1) ,
0, 5)
```

Expressions are also defined for Brush.Color and Brush.Hatch.

For more about these properties, see “Brush.Hatch” on page 300.

The following is a portion of the resulting DataWindow object. A rectangle with light gray cross-hatching highlights employees whose reviews are due soon. The line enclosing the rectangle is Light Gray and uses the pen style Solid (0):

01/11/01		Performance Review Reminder		
Department ID	Employee ID	Employee First Name	Employee Last Name	Start Date
400	888	Doug	Charlton	03/10/1991
200	902	Moira	Kelly	04/01/1991
200	913	Ken	Martel	04/16/1991
500	921	Charles	Crowley	04/22/1991
200	930	Ann	Taylor	05/08/1991
200	949	Pamela	Savarino	05/08/1991
100	958	Thomas	Sisson	07/16/1991
400	992	Joyce	Butterfield	08/13/1991
500	1013	Joseph	Barker	09/10/1991
200	1021	Paul	Sterling	10/28/1991
200	1039	Shih Lin	Chao	11/11/1991
400	1062	Barbara	Blaikie	11/20/1991
100	1090	Susan	Smith	12/13/1991
200	1101	Mark	Preston	01/09/1992
200	1142	Alison	Clark	01/19/1992
100	1157	Hing	Soo	01/29/1992
200	1162	Kevin	Goggin	02/03/1992
400	1191	Matthew	Bucceri	02/12/1992

Pen.Width

Description	The width of the line or the outline of a graphic control.
In the painter	Pen Width on the General page in the Properties view.
Value	An integer in the unit of measure specified for the DataWindow object. Units of measure include PowerBuilder units, thousandths of an inch (1000 = 1 inch), thousandths of a centimeter (1000 = 1 centimeter), or pixels.
Example	<p>The following statement causes the width of a line to be 10 PowerBuilder units if the state column for the row has the value NY. Otherwise, the line is 4 PowerBuilder units wide:</p> <pre> If (state = 'NY', 10, 4)</pre> <p>For more details and pictures, see “Example 4: changing the size and location of controls” on page 294.</p>

Pointer

Description	The image used for the mouse pointer when the pointer is over the specified control.
In the painter	Pointer on the Pointer page in the Properties view.
Value	<p>A string that specifies a value of the Pointer enumerated data type or the name of a cursor file (CUR) used for the pointer.</p> <p>Values of the Pointer enumerated data type are:</p> <ul style="list-style-type: none">Arrow!Cross!HourGlass!IBeam!Icon!Size!SizeNESW!SizeNS!SizeNWSE!SizeWE!UpArrow!

Example The following condition, entered for the Pointer property of every control in a row of expense data, changes the pointer to a column every time the value in the expense column exceeds \$100,000. Note that the pointer has no meaning in a printed report. The pointer is for use on the screen display of a DataWindow object:

```
If (expense 100000, 'pbcolumn.cur', 'arrow!')
```

Protect

Description The protection setting of a column.

In the painter Protect on the General page in the Properties view.

Value Values are:

- 0—False, the column is not protected
- 1—True, the column is protected

Timer_Interval

Description The number of milliseconds between the internal timer events.

In the painter Timer Interval on the General page in the Properties view.

Value The default is 0 (which is defined to mean 60,000 milliseconds or one minute).

Visible

Description Whether the control is visible in the DataWindow object.

In the painter Visible on the General page in the Properties view.

Value Values are:

- 0—Not visible
- 1—Visible

Example The following statement is for the Visible property of a text control with the words On Leave located to the right of columns of employee information. The statement says that if the current employee's status is L, which means On Leave, the text control is visible. Otherwise, it is invisible:

```
If (status = 'L', 1, 0)
```

The status column must be retrieved

The status column must be included in the data source even though it does not appear in the DataWindow object itself.

The Design view includes the text control at the right-hand end of the detail line. The text control is visible at runtime only if the value of the status column for the row is L.

In the resulting DataWindow object, the text control is visible only for the two employees on leave. For a picture, see “Font.Strikethrough” on page 305.

Width

Description	The width of the control.
In the painter	Width on the Position page in the Properties view.
Value	An integer in the unit of measure specified for the DataWindow object. Units of measure include PowerBuilder units, thousandths of an inch (1000 = 1 inch), thousandths of a centimeter (1000 = 1 centimeter), or pixels.
Example	<p>The following statement causes the width of a rectangle to be 500 PowerBuilder units if the state column for the row has the value NY. Otherwise, the rectangle is 1000 PowerBuilder units wide:</p> <pre>if (state = 'NY', 500, 1000)</pre> <p>For more details and pictures, see “Example 4: changing the size and location of controls” on page 294.</p>

X

Description	The distance of the control from the left edge of the DataWindow object. At runtime, the distance from the left edge of the DataWindow object is calculated by adding the margin to the x value.
In the painter	X on the Position page in the Properties view.
Value	An integer in the unit of measure specified for the DataWindow object. Units of measure include PowerBuilder units, thousandths of an inch (1000 = 1 inch), thousandths of a centimeter (1000 = 1 centimeter), or pixels.

Example The following statement causes a rectangle to be located 6.250 inches from the left if the state column for the row has the value NY. Otherwise, the rectangle is 4.000 inches from the left:

```
If (state = 'NY', 6250, 4000)
```

For more details and pictures, see “Example 4: changing the size and location of controls” on page 294.

X1, X2

Description The distance of each end of the line from the left edge of the DataWindow object as measured in the Design view. At runtime, the distance from the left edge of the DataWindow object is calculated by adding the margin to the x1 and x2 values.

In the painter X1, X2 on the Position page in the Properties view.

Value Integers in the unit of measure specified for the DataWindow object. Units of measure include PowerBuilder units, thousandths of an inch (1000 = 1 inch), thousandths of a centimeter (1000 = 1 centimeter), or pixels.

Example The following statements for the X1 and X2 properties of a line cause the line to extend from 6.250 to 7.150 inches from the left if the state column for the row has the value NY. Otherwise, the line extends from 4.000 to 6.000 inches from the left:

```
If (state = 'NY', 6250, 4000)
If (state = 'NY', 7150, 6000)
```

For more details and pictures, see “Example 4: changing the size and location of controls” on page 294.

Y

Description The distance of the control from the top of the band in which the control is located.

In the painter Y on the Position page in the Properties view.

Value An integer in the unit of measure specified for the DataWindow object. Units of measure include PowerBuilder units, thousandths of an inch (1000 = 1 inch), thousandths of a centimeter (1000 = 1 centimeter), or pixels.

Example For information, see “Example 4: changing the size and location of controls” on page 294.

Y1, Y2

Description The distance of each end of the specified line from the top of the band in which the line is located.

In the painter Y1, Y2 on the Position page in the Properties view.

Value Integers in the unit of measure specified for the DataWindow object. Units of measure include PowerBuilder units, thousandths of an inch (1000 = 1 inch), thousandths of a centimeter (1000 = 1 centimeter), or pixels.

Example The following statements for the Y1 and Y2 properties of a line cause the line to be located .400 inches (Y1 and Y2 equal .400 inches) from the top of the detail band, if the state column for the row has the value NY. Otherwise, the line is located .250 inches (Y1 and Y2 equal .250 inches) from the top of the detail band:

```
If (state = 'NY', 400, 250)
If (state = 'NY', 400, 250)
```

For more details and pictures, see “Example 4: changing the size and location of controls” on page 294.

Specifying colors

You specify a color by specifying a number that represents the color. You can specify the number explicitly or by using an expression that includes the RGB (*r, g, b*) function.

For the numbers and expressions that specify common colors, see Table 10-2 on page 315.

How the number is calculated

The formula for combining color values into a number is:

$$red + 256 * green + 256 * 256 * blue$$

where the amount of each primary color (red, green, and blue) is specified as a value from 0 to 255.

The RGB function calculates the number from the amounts of red, green, and blue specified.

Sample numeric calculation

To create cyan, you use blue and green, but no red. If you wanted to create the most saturated (bright) cyan, you would use maximum amounts of blue and green in the formula, which is indicated by the number 255 for each. The following statements show the calculation:

```
red + 256*green + 256*256*blue
0 + 256*255 + 256*256*255
0 + 65280 + 16711680
16776960
```

Sample expression using the RGB function

The following expression specifies the brightest cyan:

```
RGB (0,255,255)
```

Notice that the expression specifies the maximum for green and blue (255) and 0 for red. The expression returns the value 16776960. To specify cyan, entering the expression `RGB(0, 255, 255)` is the same as entering the number 16776960.

Numbers and expressions to enter for the common colors

Table 10-2 shows the numbers and expressions to enter for some common colors. The number and expression for a color are equivalent. You can use either.

Table 10-2: Numbers and expressions for common colors

Color	Expression to enter	Number to enter	How the number is calculated
Black	RGB (0, 0, 0)	0	0 (no color)
Blue	RGB (0, 0, 255)	16711680	256*256*255 (blue only)
Cyan	RGB (0, 255, 255)	16776960	256*255 + 256*256*255 (green and blue)
Dark Green	RGB (0, 128, 0)	32768	256*128 (green only)
Green	RGB (0, 255, 0)	65280	256*255 (green only)
Light Gray	RGB (192, 192, 192)	12632256	192 + 256*192 + 256*256*192 (some red, green, and blue in equal amounts)
Lighter Gray	RGB (224, 224, 224)	14737632	224 + 256*224 + 256*256*224 (some red, green, and blue in equal amounts)
Lightest Gray	RGB (240, 240, 240)	15790320	240 + 256*240 + 256*256*240 (some red, green, and blue in equal amounts)
Magenta	RGB (255, 0, 255)	16711935	255 + 256*256*255 (red and blue)
Red	RGB (255, 0, 0)	255	255 (red only)
White	RGB (255, 255, 255)	16777215	255 + 256*255 + 256*256*255 (red, green, and blue in equal amounts at the maximum of 255)
Yellow	RGB (255, 255, 0)	65535	255 + 256*255 (red and green)

Using Nested Reports

About this chapter

This chapter provides information about creating reports that have other reports nested in them.

Contents

Topic	Page
About nested reports	317
Creating a report using the Composite presentation style	319
Placing a nested report in another report	321
Working with nested reports	325

About reports and DataWindow objects

A report is the same as a nonupdatable DataWindow object.

This chapter shows the process of nesting reports using the Report painter in InfoMaker, but you can do the same things in the DataWindow painter with the same results.

About nested reports

A nested report is a report within another report.

There are two ways to create reports containing nested reports:

- Create a composite report using the Composite presentation style
- Place a nested report in another report

About creating a composite report

You can choose the Composite presentation style to create a new report that consists entirely of one or more nested reports. This type of report is called a composite report. A composite report is a container for other reports.

You can use composite reports to print more than one report on a page.

About placing a nested report within another report

You can place one or more reports within another report. The report you place is called the nested report. You can place a nested report in any type of report except crosstab. Most of the time you will place nested reports in freeform or tabular reports.

Often, the information in the nested report depends on information in the report in which it is placed (the base report). The nested report and the base report are related to each other by some common data. The base report and the nested report have a master/detail relationship.

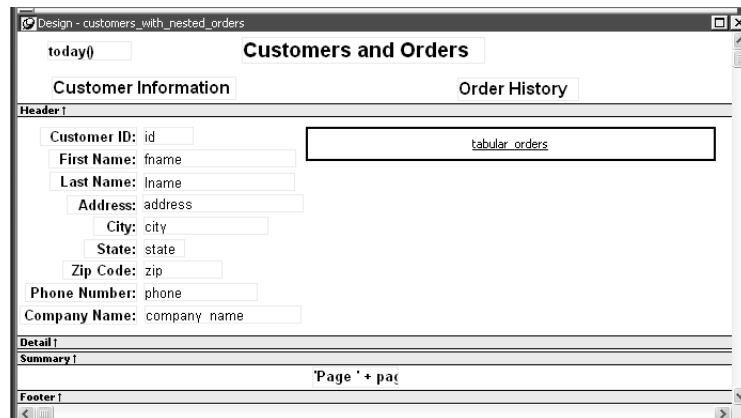
Freeform report with a related nested report

For example, the following freeform report lists all information about a customer and then includes a related nested report (which happens to be a tabular report). The related nested report lists every order that the customer has ever placed. The base report supplies the customer ID to the nested report, which requires a customer ID as a retrieval argument. This is an example of a master/detail relationship—one customer has many orders:

2/21/2001		Customers and Orders					
Customer Information			Order History				
Customer ID:	105						
First Name:	Laura						
Last Name:	McCarthy						
Address:	1210 Highway 36						
City:	Carmel						
State:	IN						
Zip Code:	46032-						
Phone Number:	(317) 555-8437						
Company Name:	Arno & Sons						
Sales Order ID	Order Date	Sales Rep ID	Line #	Product ID	Quantity	Date Shipped	
2006	09/28/98	299	1	300	48	09/28/98	
2344	03/30/98	195	1	501	36	03/31/98	
2454	06/18/98	299	1	501	36	06/17/98	
2568	09/21/98	856	1	600	36	09/22/98	
			2	601	36	09/22/98	

What you see in the Design view

In the Design view, you see everything in the base report plus a box that represents the related nested report:



The difference between nested and composite reports

There are two important differences between nesting using the Composite style and nesting a report within a base report.

Data sources The composite report does not have a data source—it is just a container for nested reports. In contrast, a base report with a nested report in it has a data source. The nested report has its own data source.

Related nesting The composite report cannot be used to relate reports to each other in the database sense. One report cannot feed a value to another report, which is what happens in a master/detail report. If you want to relate reports to each other so that you can create a master/detail report, you need to place a nested report within a base report.

How retrieval works

When you preview (run) a composite report, DataWindow Designer retrieves all the rows for one nested report, and then for another nested report, and so on until all retrieval is complete. Your computer must have a default printer specified, because composite reports are actually displayed in print preview mode.

When you preview (run) a report with another related report nested in it, DataWindow Designer retrieves all the rows in the base report first. Then DataWindow Designer retrieves the data for all nested reports related to the first row. Next, DataWindow Designer retrieves data for nested reports related to the second row, and so on, until all retrieval is complete for all rows in the base report.

For information about efficiency and retrieval, see “Supplying retrieval arguments to relate a nested report to its base report” on page 327.

Limitations on nesting reports

You cannot place a crosstab with retrieval arguments within another report as a related nested report. However, you can include a crosstab in a Composite report.

Creating a report using the Composite presentation style

❖ **To create a report using the Composite presentation style:**

- 1 Select File>New from the menu bar.

The New Report dialog box displays.

- 2 Choose the DataWindow tab page and the Composite presentation style, and click OK.

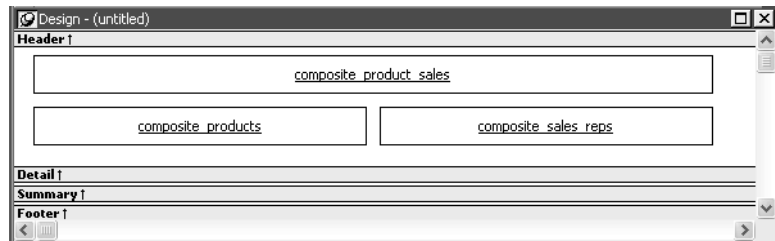
The wizard displays all reports (DataWindow objects) that are in the current project's library search path.

- Click the reports you want to include in the composite report and then click Next.

The wizard lists your choices.

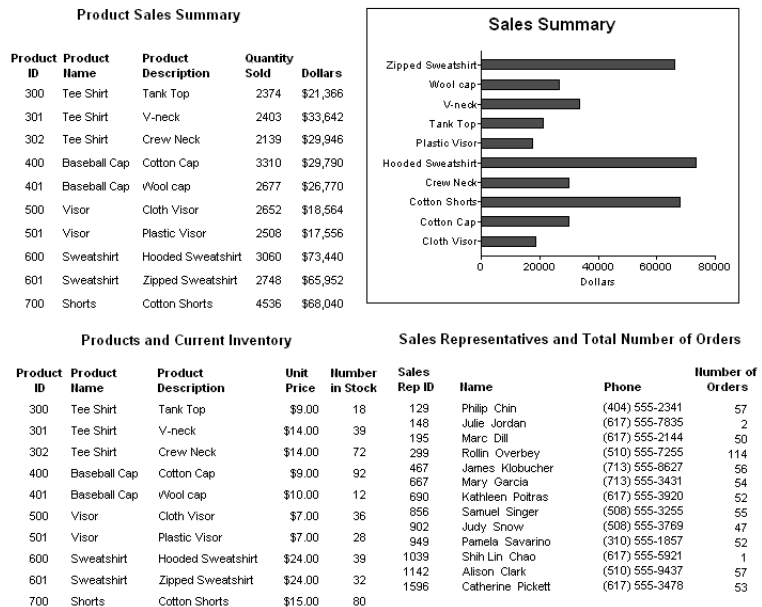
- Click Finish.

DataWindow Designer places boxes for the selected reports in the Design view. In this example, you see three reports:



- Select File>Save from the menu bar and assign a name to the composite report.

- Look at the Preview view of the report:



Notice that you are in print preview (which is read-only).

Working with composite reports

Many of the options available for working with reports, such as Rows>Filter, Rows>Import, and Rows>Sort, are disabled for a composite report. If you want to use any of these options, you need to access the nested report(s), where these options are available.

- 7 Continue to enhance the composite report (for example, add a date and title).

Placing a nested report in another report

When you place a nested report in another report, the two reports can be independent of each other, or they can be related in the database sense by sharing some common data such as a customer number or a department number. If the reports are related, you need to do some extra things to both the base report and the related nested report.

Usually, when you place a report within a report rather than create a composite report, you want to relate the reports. Those instructions are first.

Placing a related nested report in another report

Typically, a related nested report provides the details for a master report. For example, a master report might provide information about customers. A related nested report placed in the master report could provide information about all the orders that belong to each customer.

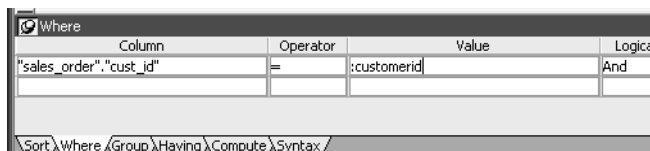
❖ **To place a related nested report in another report:**

- 1 Create the nested report (DataWindow object) that you plan to place in the base report.
- 2 Define a retrieval argument for the nested report.

For example, suppose the nested report lists orders and you want to list orders for a particular customer. To define a retrieval argument, you would:

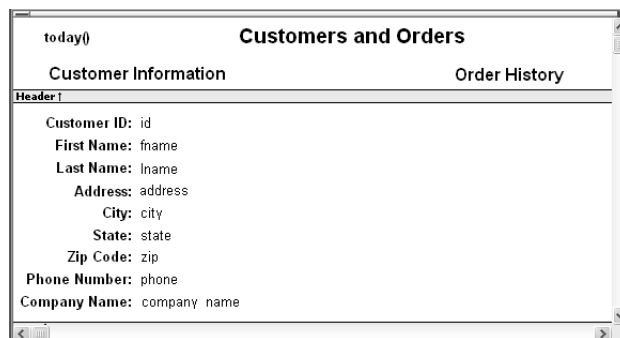
- Select Design>Data Source to go to the Select painter.
 - Select Design>Retrieval Arguments from the menu bar in the Select painter.
 - Define a retrieval argument in the Specify Retrieval Arguments dialog box. In the example, *customerID* is the name assigned to the retrieval argument.
- 3 Specify the retrieval argument in a WHERE clause for the SELECT statement.

The WHERE clause in this example tells the DBMS to retrieve rows where the value in the column *cust_id* equals the value of the argument *:customerid*:



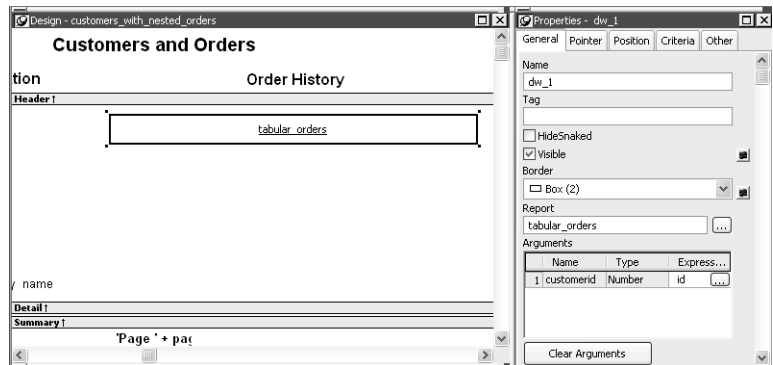
At this point, when you run the report to retrieve data, you are prompted to enter a value for *:customerid*. Later in these steps, you will specify that the base report supply the values for *:customerid* instead of prompting for values.

- 4 Open or create the report you want to have as the base report.
- In the example, the base report is one that lists customers and has a place for the order history of each customer:



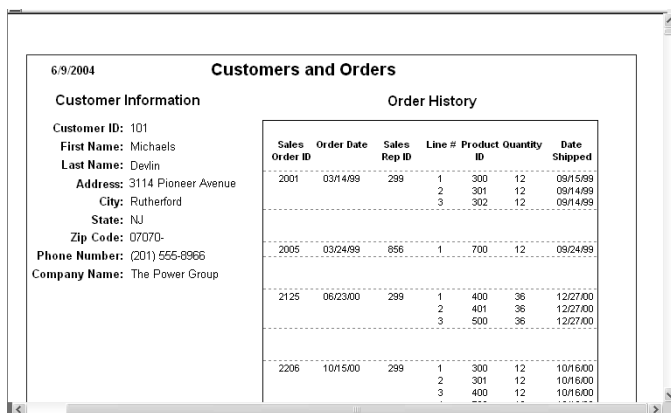
- 5 Select Insert>Control>Report from the menu bar.
- 6 In the Design view, click where you want to place the report.
The Select Report dialog box displays, listing defined reports (DataWindow objects) in the current project's library search path.
- 7 Select the report you want, and click OK.
A box representing the report displays in the Design view.
- 8 With the report still selected, select the General page of the Properties view.

The Arguments box lists arguments defined for the nested report and provides a way for you to specify how information from the base report will be used to supply the values of arguments to the nested report.



- 9 Supply the base report column or the expression that will supply the argument's value. To do this, click the button in the Expression column.
The Modify Expression dialog box displays. In this dialog box, you can easily select one of the columns or develop an expression. In the example, the column named id from the base report will supply the value for the argument :customerid in the nested report.
- 10 Select File>Save from the menu bar and assign a name to the report.

- 11 In the Preview view, you can see what your report looks like:



6/9/2004

Customers and Orders

Customer Information		Order History						
Customer ID:	101	Sales Order ID	Order Date	Sales Rep ID	Line #	Product ID	Quantity	Date Shipped
First Name:	Michaels	2001	03/14/99	299	1	300	12	09/15/99
Last Name:	Devlin				2	301	12	09/14/99
Address:	3114 Pioneer Avenue				3	302	12	09/14/99
City:	Rutherford							
State:	NJ							
Zip Code:	07070-							
Phone Number:	(201) 555-8966	2005	03/24/99	856	1	700	12	09/24/99
Company Name:	The Power Group							
		2125	06/23/00	299	1	400	36	12/27/00
					2	401	36	12/27/00
					3	500	36	12/27/00
		2206	10/15/00	299	1	300	12	10/16/00
					2	301	12	10/16/00
					3	400	12	10/16/00

Placing an unrelated nested report in another report

When you place an unrelated nested report in a base report, the entire nested report appears with *each* row of the base report.

❖ **To place an unrelated nested report in another report:**

- 1 Create or open the report you want as the base report.
- 2 Select Insert>Control>Report from the menu bar.
- 3 In the Design view, click where you want to place the report.

The Select Report dialog box displays, listing defined reports (DataWindow objects) in the current project's library search path.

- 4 Select the report you want to nest in the base report, and click OK.

A box representing the nested report displays in the Design view.

- 5 Select File>Save from the menu bar and if the base report is newly created, assign a name to it.

Working with nested reports

When you use nested reports either in composite reports or in other base reports, several enhancements and options are available. An easy way to see what you can do is to select the nested report and look at the Properties view for it.

Many of the options in the Properties view are described in Chapter 5, “Enhancing DataWindow Objects.” For example, using borders on nested reports is like using borders on any control.

This section describes activities that apply only to nested reports or that have special meaning for nested reports. It covers:

- "Adjusting nested report width and height" next
- “Changing a nested report from one report to another” on page 326
- “Modifying the definition of a nested report” on page 326
- “Adding another nested report to a composite report” on page 327
- “Supplying retrieval arguments to relate a nested report to its base report” on page 327
- “Specifying criteria to relate a nested report to its base report” on page 329
- “Using options for nested reports” on page 330

Adjusting nested report width and height

When you preview a report with nested reports, the width of the nested report may be unacceptable. This can happen, for example, if you change the design of the nested report or if you use newspaper columns in a nested report. The width of the nested report is not adjusted to fit its contents at runtime; if the report is too narrow, some columns may be truncated. For example, if the size of the nested report is set to 6 inches wide in the parent report, columns in the nested report that exceed that width are not displayed in the parent report.

❖ **To adjust report width:**

- 1 In the Design view, position the pointer near a vertical edge of the nested report and press the left mouse button.
- 2 Drag the edge to widen the nested report.

- 3 Check the new width in the Preview view.

When you Print preview a DataWindow that contains a nested N-Up report with newspaper columns across the page, you might find that blank pages display (and print) when the nested report in the detail band fills the page. This is because any white space at the bottom of the band is printed to a second page. You can usually solve this problem by dragging up the detail band to eliminate the white space between the nested report and the band, or even to overlap the bottom of the representation of the nested report.

Changing a nested report from one report to another

You can change the nested report that is used. For example, you may work on several versions of a nested report and need to update the version of the nested report that the composite or base report uses.

❖ To change the nested report to a different report:

- 1 Select the nested report in the Design view.
- 2 In the Properties view, General property page, click the button next to the Report box.
- 3 Select the report you want to use, and click OK.

The name of the report that displays in the box in the Design view changes to the new one.

Modifying the definition of a nested report

You can modify the definition of the nested report. You can do this directly from the composite report or base report that contains the nested report.

❖ To modify the definition of a nested report from the composite report or base report:

- 1 Position the pointer on the nested report whose definition you want to modify, and display the pop-up menu.
- 2 Select Modify Report from the pop-up menu.

The nested report opens and displays in the painter. Both the composite or base report and the nested report are open.

- 3 Modify the report.

- 4 Select File>Close from the menu bar.
You are prompted to save your changes.
- 5 Click OK.
You return to the composite report or to the base report that includes the nested report.

Adding another nested report to a composite report

After you have created a composite report, you might want to add another report. The following procedure describes how. For information on adding a nested report to a report that is *not* a composite report, see “Placing a related nested report in another report” on page 321 or “Placing an unrelated nested report in another report” on page 324.

❖ To add another nested report to a composite report:

- 1 Open the composite report.
- 2 Select Insert>Control>Report from the menu bar.
- 3 Click in the Design view where you want to place the report.
The Select Report dialog box displays, listing defined reports (DataWindow objects) in the current project’s library search path.
- 4 Select the report you want and click OK.
A box representing the report displays in the Design view.

Supplying retrieval arguments to relate a nested report to its base report

The most efficient way to relate a nested report to its base report is to use retrieval arguments. If your nested report has arguments defined, you use the procedure described in this section to supply the retrieval argument value from the base report to the nested report. (The procedure described is part of the whole process covered in “Placing a related nested report in another report” on page 321.)

Why retrieval arguments are efficient

Some DBMSs have the ability to bind input variables in the WHERE clause of the SELECT statement. When you use retrieval arguments, a DBMS *with this capability* sets up placeholders in the WHERE clause and compiles the SELECT statement *once*. DataWindow Designer retains this compiled form of the SELECT statement for use in subsequent retrieval requests.

Requirements for reusing the compiled SELECT statement

To enable DataWindow Designer to retain and reuse the compiled SELECT statement:

- The database interface must support binding of input variables.
- You must enable binding support by setting the DisableBind DBParm to 0, which is the default.
- You must enable caching in the database profile. Set the SQLCache DBParm parameter to the number of levels of nesting plus 5.

For more information, see the description of the SQLCache and DisableBind DBParm parameters in the online Help.

Nested reports in composite reports

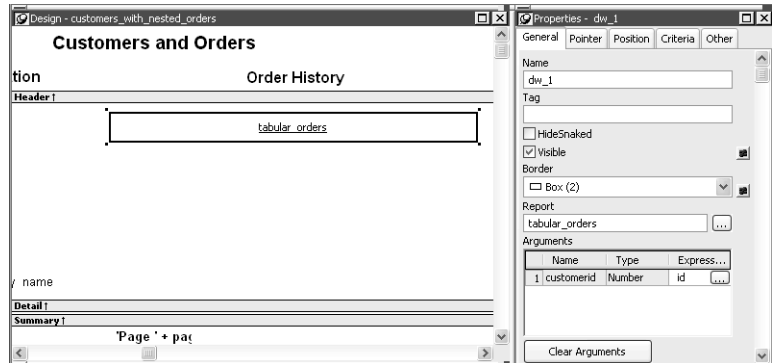
If the base report is a composite report, you need to define retrieval arguments for the composite report before you can supply them to the nested report.

In the Properties view for the composite report, select the General page. Then define the retrieval arguments that the nested report needs, taking care to specify the correct type.

❖ To supply a retrieval argument value from the base report to the nested report:

- 1 Make sure that the nested report has been set up to take one or more retrieval arguments.
See “Placing a nested report in another report” on page 321.
- 2 Select the nested report and then select the General page of the Properties view.

The Arguments box lists arguments defined for the nested report and provides a way for you to specify how information from the base report will supply the value of the argument to the nested report.



- 3 Supply the base report column that will supply the argument's value. To do this, click the button in the Expression column.

The Modify Expression dialog box displays. In this dialog box, you can easily select one of the columns or develop an expression. In the example, the column named `id` from the base report will supply the value for the argument `:customerid` in the nested report.

When you run the report now, you are not prompted for retrieval argument values for the nested report. The base report supplies the retrieval argument values automatically.

Specifying criteria to relate a nested report to its base report

If you do not have arguments defined for the nested report and if database efficiency is not an issue, you can place a nested report in another report and specify criteria to pass values to the related nested report.

How the DBMS processes SQL if you use the specify criteria technique

If you use the specify criteria technique, the DBMS repeatedly recompiles the SELECT statement and then executes it. The recompilation is necessary for each possible variation of the WHERE clause.

❖ **To specify criteria to relate a nested report to its base report:**

- 1 Select the nested report and then select the Criteria page in the Properties view.

The Criteria property page provides a way for you to specify how information from the base report will supply the retrieval criteria to the nested report.

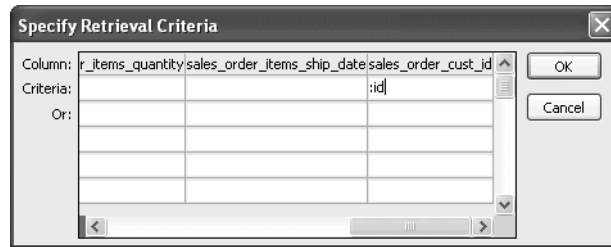
- 2 Click the button next to the criteria box.

The Specify Retrieval Criteria dialog box displays.

- 3 Enter the retrieval criteria and click OK.

The rules for specifying criteria are the same as for specifying criteria in the Quick Select data source. Multiple criteria in one line are ANDed together. Criteria entered on separate lines are ORed together.

In this example, the customer ID (the id column) is the retrieval criterion being supplied to the nested report. Notice that the id column is preceded by a colon (:), which is required:



When you run the report now, DataWindow Designer retrieves rows in the nested report based on the criteria you have specified. In the example, the customer ID column in the base report determines which rows from the sales_order table are included for each customer.

Using options for nested reports

Using the Autosize Height option

Autosize Height should be on for all nested reports except graphs. This option ensures that the height of the nested report can change to accommodate the rows that are returned.

This option is on by default for all nested reports except graphs. Usually there is no reason to change it. If you do want to force a nested report to have a fixed height, you can turn this option off.

Note that all bands in the DataWindow also have an Autosize Height option. The option is off by default and must be on for the Autosize Height option for the nested report to work properly.

❖ **To change the Autosize Height option for a nested report:**

- 1 In the Design view, select the nested report.
- 2 In the Properties view, select the Position properties page.
- 3 Select/clear the Autosize Height check box.

Handling large rows

To avoid multiple blank pages or other anomalies in printed reports, never create a DataWindow object with a data row greater than the size of the target page. To handle large text-string columns, break the large string into a series of small strings. The smaller strings are used to populate individual data rows within a nested report instead of using a single AutoSize Height text column.

Using the Slide option

DataWindow Designer determines the appropriate Slide options when positioning the nested report(s) and assigns default values. Usually, you should not change the default values:

- The Slide Left option is on by default for grid and crosstab style reports and off by default for all others. Having Slide Left on for grid and crosstab ensures that these reports break horizontally on whole columns and not in the middle of a column.
- The Slide Up All Above/Directly Above options ensure that the nested report uses just as much vertical space as it needs. One of these options is on by default for all nested reports.

For more information, see “Sliding controls to remove blank space in a DataWindow object” on page 219.

Using the New Page option (composite only)

The New Page option forces a new page for a nested report used *in a composite report*. By default, this option is off.

❖ **To specify that a nested report in a composite report should begin on a new page:**

- 1 In the Design view, select the nested report.
- 2 In the Properties view, select the General page.
- 3 Select the New Page check box.

A check mark displays, indicating the option is selected.

Using the Trail Footer option (composite only)

The Trail Footer option controls the placement of the footer for the last page of a nested report *in a composite report*. By default, this option is on. The footer appears directly under the contents of the nested report and not at the bottom of the page.

❖ **To specify that the footer should appear at the bottom of the page:**

- 1 In the Design view, select the nested report.
- 2 In the Properties view, select the General page.
- 3 Clear the Trail Footer check box.

The check mark next to the option disappears, indicating the option is no longer selected. The footer appears at the bottom of the page on all pages of the nested report, including the last page. Note that if another nested report begins on the same page, the footer from the earlier report might be misleading or confusing.

About this chapter

This chapter describes how to build crosstabs.

Contents

Topic	Page
About crosstabs	333
Creating crosstabs	337
Associating data with a crosstab	338
Previewing crosstabs	344
Enhancing crosstabs	344

About crosstabs

Cross tabulation is a useful technique for analyzing data. By presenting data in a spreadsheet-like grid, a crosstab lets users view summary data instead of a long series of rows and columns. For example, in a sales application you might want to summarize the quarterly unit sales of each product.

In DataWindow Designer, you create crosstabs by using the Crosstab presentation style. When data is retrieved into the DataWindow object, the crosstab processes all the data and presents the summary information you have defined for it.

An example

Crosstabs are easiest to understand through an example. Consider the Printer table in the EAS Demo DB. It records quarterly unit sales of printers made by sales representatives in one year. (This is the same data used to illustrate graphs in Chapter 15, “Working with Graphs.”)

Table 12-1: The Printer table in the EAS Demo DB

Rep	Quarter	Product	Units
Simpson	Q1	Stellar	12
Jones	Q1	Stellar	18
Perez	Q1	Stellar	15
Simpson	Q1	Cosmic	33
Jones	Q1	Cosmic	5
Perez	Q1	Cosmic	26
Simpson	Q1	Galactic	6
Jones	Q1	Galactic	2
Perez	Q1	Galactic	1
.	.	.	.
.	.	.	.
.	.	.	.
Simpson	Q4	Stellar	30
Jones	Q4	Stellar	24
Perez	Q4	Stellar	36
Simpson	Q4	Cosmic	60
Jones	Q4	Cosmic	52
Perez	Q4	Cosmic	48
Simpson	Q4	Galactic	3
Jones	Q4	Galactic	3
Perez	Q4	Galactic	6

This information can be summarized in a crosstab. Here is a crosstab that shows unit sales by printer for each quarter:

This report summarizes the unit sales of printers in each quarter.
11/9/98

Sum Of Units	Quarter				Grand Total
Product	Q1	Q2	Q3	Q4	Grand Total
Cosmic	64	104	134	160	462
Galactic	9	21	13	12	55
Stellar	45	51	60	90	246
Grand Total	118	176	207	262	763

The first-quarter sales of Cosmic printers displays in the first data cell. (As you can see from the data in the Printer table shown before the crosstab, in Q1 Simpson sold 33 units, Jones sold 5 units, and Perez sold 26 units—totaling 64 units.) DataWindow Designer calculates each of the other data cells the same way.

To create this crosstab, you only have to tell DataWindow Designer which database columns contain the raw data for the crosstab, and DataWindow Designer does all the data summarization automatically.

What crosstabs do

Crosstabs perform two-dimensional analysis:

- The first dimension is displayed as columns across the crosstab.

In the preceding crosstab, the first dimension is the quarter, whose values are in the Quarter column in the database table.

- The second dimension is displayed as rows down the crosstab.

In the preceding crosstab, the second dimension is the type of printer, whose values are in the Product column in the database table.

Each cell in a crosstab is the intersection of a column (the first dimension) and a row (the second dimension). The numbers that appear in the cells are calculations based on both dimensions. In the preceding crosstab, it is the sum of unit sales for the quarter in the corresponding column and printer in the corresponding row.

Crosstabs also include summary statistics. The preceding crosstab totals the sales for each quarter in the last row and the total sales for each printer in the last column.

How crosstabs are implemented in DataWindow Designer

Crosstabs in DataWindow Designer are implemented as grid DataWindow objects. Because crosstabs are grid DataWindow objects, users can resize and reorder columns at runtime (if you let them).

Import methods return empty result

A crosstab report takes the original result set that was retrieved from the database, sorts it, summarizes it, and generates a new summary result set to fit the design of the crosstab. The `ImportFile`, `ImportClipboard`, and `ImportString` methods can handle only the original result set, and they return an empty result when used with a crosstab report.

Two types of crosstabs

There are two types of crosstabs:

- Dynamic
- Static

Dynamic crosstabs

With dynamic crosstabs, DataWindow Designer builds all the columns and rows in the crosstab dynamically when you run the crosstab. The number of columns and rows in the crosstab match the data that exists at runtime.

Using the preceding crosstab as an example, if a new printer was added to the database after the crosstab was saved, there would be an additional row in the crosstab when it is run. Similarly, if one of the quarter's results was deleted from the database after the crosstab was saved, there would be one less column in the crosstab when it is run.

By default, crosstabs you build are dynamic.

Static crosstabs

Static crosstabs are quite different from dynamic crosstabs. With static crosstabs, DataWindow Designer establishes the columns in the crosstab based on the data in the database when you *define* the crosstab. (It does this by retrieving data from the database when you initially define the crosstab.) No matter what values are in the database when you later run the crosstab, the crosstab always has the same columns as when you defined it.

Using the preceding crosstab as an example, if there were four quarters in the database when you defined and saved the crosstab, there would always be four columns (Q1, Q2, Q3, and Q4) in the crosstab at runtime, even if the number of columns changed in the database.

Advantages of dynamic crosstabs

Dynamic crosstabs are used more often than static crosstabs, for the following reasons:

- You can define dynamic crosstabs very quickly because no database access is required at definition time.
- Dynamic crosstabs always use the current data to build the columns and rows in the crosstab. Static crosstabs show a snapshot of columns as they were when the crosstab was defined.
- Dynamic crosstabs are easy to modify: all properties for the dynamically built columns are replicated at runtime automatically. With static crosstabs, you must work with one column at a time.

Creating crosstabs

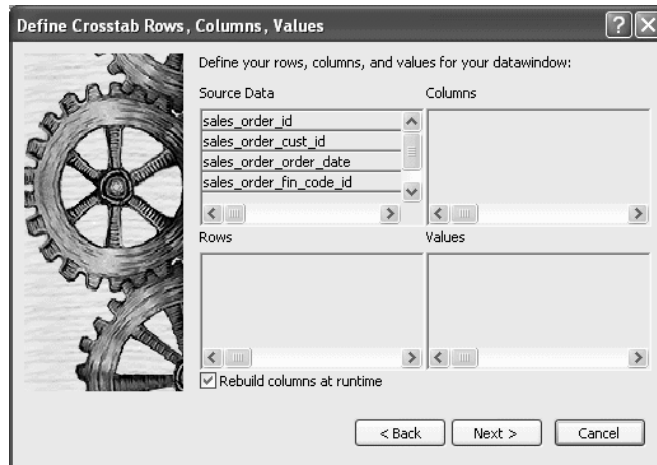
- ❖ **To create a crosstab:**
 - 1 Select File>New from the menu bar.
The New dialog box displays.
 - 2 Select the DataWindow tab.
 - 3 Select the Crosstab presentation style, then click OK.
You are prompted to specify the data.
 - 4 Specify the data you want retrieved into the DataWindow object.
For more information, see Chapter 4, “Defining DataWindow Objects.”
You are prompted to specify the data for the columns, rows, and cell values in the crosstab.
 - 5 Enter the definitions for the columns, rows, and cell values in the crosstab.
See “Associating data with a crosstab” on page 338.
 - 6 Click Next.
 - 7 Click Color and Border settings and click Next.
 - 8 Review your specifications and click Finish.
DataWindow Designer creates the crosstab.
 - 9 (Optional) Specify other properties of the crosstab.
See “Enhancing crosstabs” on page 344.
 - 10 Save the DataWindow object in a library.

Associating data with a crosstab

You associate crosstab columns, rows, and cell values with columns in a database table or other data source.

❖ **To associate data with a crosstab:**

- 1 If you are defining a new crosstab, the Define Crosstab Rows, Columns, Values dialog box displays after you specify the data source.



- 2 Specify the database columns that will populate the columns, rows, and values in the crosstab, as described below.
- 3 To build a dynamic crosstab, make sure the Rebuild Columns At Runtime box is selected.

For information about static crosstabs, see “Creating static crosstabs” on page 353.

- 4 Click Next.

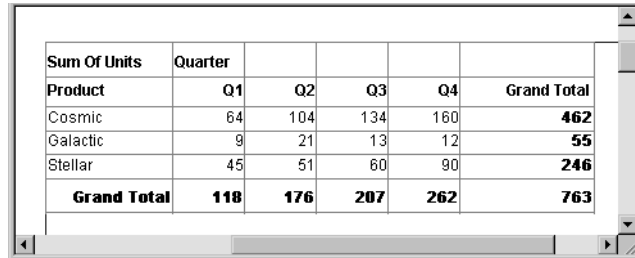
Specifying the information

To define the crosstab, drag the column names from the Source Data box in the Crosstab Definition dialog box (or Wizard page) into the Columns, Rows, or Values box, as appropriate.

If you change your mind or want to edit the DataWindow object later, select Design>Crosstab from the menu bar and drag the column name out of the Columns, Row, or Values box and drop it. Then specify a different column.

Dynamic crosstab example

The process is illustrated using the following dynamic crosstab. The columns in the database are Rep, Quarter, Product, and Units. The crosstab shows the number of printers sold by Quarter:



Sum Of Units	Quarter					
Product	Q1	Q2	Q3	Q4	Grand Total	
Cosmic	64	104	134	160	462	
Galactic	9	21	13	12	55	
Stellar	45	51	60	90	246	
Grand Total	118	176	207	262	763	

Specifying the columns

You use the Columns box to specify one or more of the retrieved columns to provide the columns in the crosstab. When users run the crosstab, there is one column in the crosstab for each unique value of the database column(s) you specify here.

❖ To specify the crosstab's columns:

- Drag the database column from the Source Data box into the Columns box.

Using the printer example, to create a crosstab where the quarters form the columns, specify Quarter as the Columns value. Because there are four values in the table for Quarter (Q1, Q2, Q3, and Q4), there are four columns in the crosstab.

Specifying the rows

You use the Rows box to specify one or more of the retrieved columns to provide the rows in the crosstab. When users run the crosstab, there is one row in the crosstab for each unique value of the database column(s) you specify here.

❖ To specify the crosstab's rows:

- Drag the database column from the Source Data box into the Rows box.

Using the printer example, to create a crosstab where the printers form the rows, specify Product as the Rows value. Because there are three products (Cosmic, Galactic, and Stellar), at runtime there are three rows in the crosstab.

Columns that use code tables

If you specify columns in the database that use code tables, where data is stored with a data value but displayed with more meaningful display values, the crosstab uses the column's display values, not the data values. For more information about code tables, see Chapter 8, "Displaying and Validating Data."

Specifying the values

Each cell in a crosstab holds a value. You specify that value in the Values box. Typically you specify an aggregate function, such as Sum or Avg, to summarize the data. At runtime, each cell has a calculated value based on the function you provide here and the column and row values for the particular cell.

❖ **To specify the crosstab's values:**

- 1 Drag the database column from the Source Data box into the Values box. DataWindow Designer displays an aggregate function for the value. If the column is numeric, DataWindow Designer uses Sum. If the column is not numeric, DataWindow Designer uses Count.
- 2 If you want to use an aggregate function other than the one suggested by DataWindow Designer, double-click the item in the Values box and edit the expression. You can use any of the other aggregate functions supported in the DataWindow painter, such as Max, Min, and Avg.

Using the printer example, you would drag the Units column into the Values box and accept the expression `sum(units for crosstab)`.

Using expressions

Instead of simply specifying database columns, you can use any valid DataWindow expression to define the columns, rows, and values used in the crosstab. You can use any DataWindow expression function in the expression.

For example, say a table contains a date column named `SaleDate`, and you want a column in the crosstab for each month. You could enter the following expression for the Columns definition:

```
Month(SaleDate)
```

The `Month` function returns the integer value (1–12) for the specified month. Using this expression, you get columns labeled 1 through 12 in the crosstab. Each database row for January sales is evaluated in the column under 1, each database row for February sales is evaluated in the column under 2, and so on.

❖ **To specify an expression for columns, rows, or values:**

- 1 In the Crosstab Definition dialog box (or wizard page), double-click the item in the Columns, Rows, or Values box.

The Modify Expression dialog box displays.

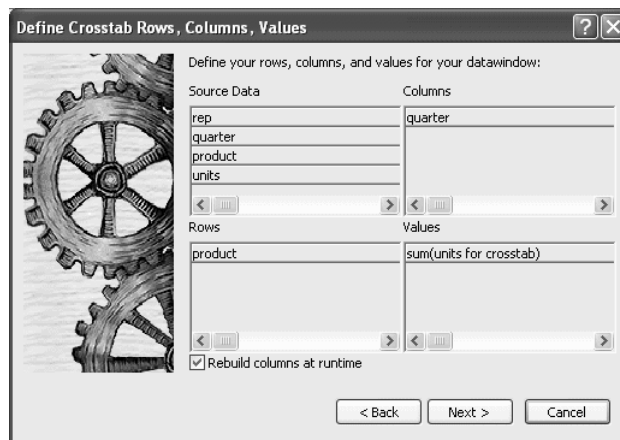
- 2 Specify the expression and click OK.

Viewing the crosstab

After you have specified the data for the crosstab's columns, rows, and values, DataWindow Designer displays the crosstab definition in the Design view.

For example, to create the dynamic crosstab shown as the “Dynamic crosstab example” on page 339, you would:

- 1 Drag the quarter column from the Source Data box to the Columns box.
- 2 Drag the product column from the Source Data box to the Rows box.
- 3 Drag the units column from the Source Data box to the Values box and accept the expression `sum(units for crosstab)`.
- 4 Select the Rebuild Columns at Runtime check box.



In the Design view, the crosstab looks like this:

Sum Of Units	Quarter		
Header [1] ↑			
Product	@quarter	Grand Total	
Header [2] ↑			
product	units	crosstabsum(1)	
Detail ↑			
"Grand Total"	sum(units	sum(grand_sum_units for	
Summary ↑			
Footer ↑			

Notice that in the Design view, DataWindow Designer shows the quarter entries using the symbolic notation @quarter (with dynamic crosstabs, the actual data values are not known at definition time). @quarter is resolved into the actual data values (in this case, Q1, Q2, Q3, and Q4) when the crosstab runs.

The crosstab is generated with summary statistics: the rows and columns are totaled for you.

At this point, the crosstab looks like this in the Preview view with data retrieved:

Sum Of Units	Quarter				
Product	Q1	Q2	Q3	Q4	Grand Total
Cosmic	64	104	134	160	462
Galactic	9	21	13	12	55
Stellar	45	51	60	90	246
Grand Total	118	176	207	262	763

- Because quarter was selected as the Columns definition, there is one column in the crosstab for each unique quarter (Q1, Q2, Q3, and Q4).
- Because product was selected as the Rows definition, there is one row in the crosstab for each unique product (Cosmic, Galactic, and Stellar).
- Because sum(units for crosstab) was selected as the Values definition, each cell contains the total unit sales for the corresponding quarter (the Columns definition) and product (the Rows definition).
- DataWindow Designer displays the grand totals for each column and row in the crosstab.

Specifying more than one row or column

Typically you specify one database column as the Columns definition and one database column for the Rows definition, as in the printer crosstab. But you can specify as many columns (or expressions) as you want.

For example, consider a crosstab that has the same specification as the crosstab in “Viewing the crosstab” on page 341, except that two database columns, quarter and rep, have been dragged to the Columns box.

DataWindow Designer displays this in the Design view:

Sum Of Units	Quarter	Rep		
Header [1] ↑				
	@quarter	@quarter	Sum Of Units	
Header [2] ↑				
Product	@rep		Grand Total	
Header [3] ↑				
product	units	crosstabsum(1, 2, "@quar/crosstabsum(1)		
Detail ↑				
"Grand Total"	sum(unitssum(sum_units for all)		sum(grand_sum_units for	
Summary ↑				
Footer ↑				

This is what you see at runtime:

Sum Of Units	Q1			Q1 Sum Of Units	Q2			Q2 Sum Of Units
	Jones	Perez	Simpson		Jones	Perez	Simpson	
Cosmic	5	26	33	64	36	28	40	104
Galactic	2	1	6	9	6	3	12	21
Stellar	18	15	12	45	15	20	16	51
Grand Total	25	42	51	118	57	51	68	176

For each quarter, the crosstab shows sales of each printer by each sales representative.

Previewing crosstabs

When you have defined the crosstab, you can see it with data in the Preview view.

❖ **To preview the crosstab:**

- 1 Select View>Preview from the menu bar to display the Preview view.
- 2 Click on the Preview view to be sure it is current.
- 3 Select Rows>Retrieve from the menu bar.

DataWindow Designer retrieves the rows and performs the cross tabulation on the data.

Retrieve on Preview makes retrieval happen automatically

If the crosstab definition specifies Retrieve on Preview, retrieval happens automatically when the Preview view first displays.

- 4 Continue enhancing your DataWindow object and retrieve again when necessary to see the results of your enhancements.

Enhancing crosstabs

When you have provided the data definitions, the crosstab is functional, but you can enhance it before using it. Because a crosstab is a grid DataWindow object, you can enhance a crosstab using the same techniques you use in other DataWindow objects. For example, you can:

- Sort or filter rows
- Change the column headers
- Specify fonts, colors, mouse pointers, and borders
- Specify column display formats

For more on these and the other standard enhancements you can make to DataWindow objects, see Chapter 5, “Enhancing DataWindow Objects.”

The rest of this section covers topics either unique to crosstabs or especially important when working with crosstabs:

- "Specifying basic properties" next
- "Modifying the data associated with the crosstab" on page 346
- "Changing the names used for the columns and rows" on page 346
- "Defining summary statistics" on page 347
- "Cross-tabulating ranges of values" on page 350
- "Creating static crosstabs" on page 353
- "Using property conditional expressions" on page 354

Specifying basic properties

Crosstabs are implemented as grid DataWindow objects, so you can specify the following grid properties for a crosstab:

- When grid lines are displayed
 - How users can interact with the crosstab at runtime
- ❖ **To specify the crosstab's basic properties:**
- 1 In the Properties view, select the General tab.
 - 2 Specify basic crosstab properties.

Table 12-2 lists basic crosstab properties.

Table 12-2: Basic properties for crosstabs

Option	Result
Display	<p><i>On</i> — Grid lines always display.</p> <p><i>Off</i> — Grid lines never display (columns cannot be resized at runtime).</p> <p><i>Display Only</i> — Grid lines display only when the crosstab displays online.</p> <p><i>Print Only</i> — Grid lines display only when the contents of the crosstab are printed.</p>
Column Moving	Columns can be moved at runtime.
Mouse Selection	Data can be selected at runtime (and, for example, copied to the clipboard).
Row Resize	Rows can be resized at runtime.

Modifying the data associated with the crosstab

When you initially define the crosstab, you associate the crosstab rows and columns with columns in a database table or other data source. You can change the associated data anytime in the Crosstab Definition dialog box.

❖ **To open the Crosstab Definition dialog box:**

- 1 Position the mouse below the footer band in the workspace and display the pop-up menu.
- 2 Select Crosstab from the pop-up menu.

The Crosstab Definition dialog box displays.

❖ **To modify the data associated with a crosstab:**

- 1 In the Crosstab Definition dialog box, fill in the boxes for Columns, Rows, and Values as described in “Associating data with a crosstab” on page 338.
- 2 Click OK.

Changing the names used for the columns and rows

Sometimes names of columns in the database might not be meaningful. You can change the names that are used to label rows and columns in crosstabs so that the data is easier to understand.

❖ **To change the names used in crosstabs:**

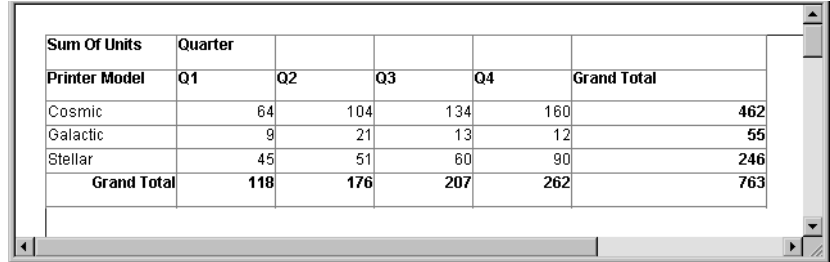
- 1 In the Crosstab Definition dialog box, double-click the name of the column in the Source Data box.
The New Name dialog box displays.
- 2 Specify the name you want used to label the corresponding column. You can have multiple-word labels by using underscores: underscores are replaced by spaces in the Design view and at runtime.
- 3 Click OK.

DataWindow Designer changes the column name in the Source Data box and anywhere else the column is used.

Example

For example, if you want the product column to be labeled *Printer Model*, double-click product in the Crosstab Definition dialog box and specify `printer_model` in the New Name dialog box.

When the crosstab runs, you see this:



Sum Of Units	Quarter				
Printer Model	Q1	Q2	Q3	Q4	Grand Total
Cosmic	64	104	134	160	462
Galactic	9	21	13	12	55
Stellar	45	51	60	90	246
Grand Total	118	176	207	262	763

Defining summary statistics

When you generate a crosstab, the columns and rows are automatically totaled for you. You can include other statistical summaries in crosstabs as well. To do that, you place computed fields in the workspace.

❖ **To define a column summary:**

- 1 Enlarge the summary band to make room for the summaries.
- 2 Select Insert>Control>Computed Field from the menu bar.
- 3 Click the cell in the summary band where you want the summary to display.

The Modify Expression dialog box displays.

- 4 Define the computed field.

For example, if you want the average value for a column, specify `avg(units for all)`, where `units` is the column providing the values in the crosstab.

This is a crosstab that has been enhanced to show averages and maximum values for each column. This is the Design view:

This is the crosstab at runtime:

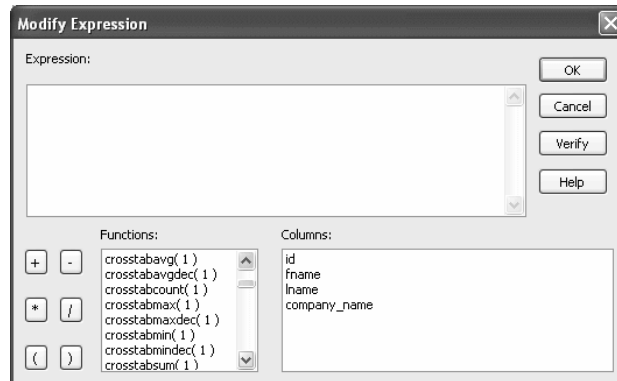
❖ To define a row summary:

- 1 Select Insert>Control>Computed Field from the menu bar.
- 2 Click the empty cell to the right of the last column in the detail band.
The Modify Expression dialog box displays.
- 3 Define the computed field. You should use one of the crosstab functions, described next.

Using crosstab functions

There are nine special functions you can use only in crosstabs: CrosstabAvg, CrosstabAvgDec, CrosstabCount, CrosstabMax, CrosstabMaxDec, CrosstabMin, CrosstabMinDec, CrosstabSum, and CrosstabSumDec.

These functions are listed in the Functions box when you define a computed field in a crosstab:



Each of these functions returns the corresponding statistic about a row in the crosstab (average, count, maximum value, minimum value, or sum). You place computed fields using these functions in the detail band in the Design view. Use the functions with the Dec suffix when you want to return a decimal datatype.

By default, DataWindow Designer places CrosstabSum and CrosstabSumDec in the detail band, which returns the total for the corresponding row.

How to specify the functions

Each of these functions takes one numeric argument, which refers to the expression defined for Values in the Crosstab Definition dialog box. The first expression for Values is numbered 1, the second is numbered 2, and so on.

Generally, crosstabs have only one expression for Values, so the argument for the crosstab functions is 1. So, for example, if you defined `sum(units for crosstab)` as your Values expression, DataWindow Designer places `CrosstabSum(1)` in the detail band.

If you want to cross-tabulate both total unit sales and a projection of future sales, assuming a 20 percent increase in sales (that is, sales that are 1.2 times the actual sales), you define two expressions for Values:

```
sum(units for crosstab)
sum(units * 1.2 for crosstab)
```

Here `CrosstabSum(1)` returns the total of `sum(units for crosstab)` for the corresponding row. `CrosstabSum(2)` returns the total for `sum(units * 1.2 for crosstab)`.

For more information

For complete information about defining computed fields, see Chapter 5, “Enhancing DataWindow Objects.”

For more about the crosstab functions, see the *DataWindow Object Reference* in the online Help.

Cross-tabulating ranges of values

You can build a crosstab where each row tabulates a *range* of values, instead of one discrete value, and you can make each column in the crosstab correspond to a range of values.

For example, in cross-tabulating departmental salary information, you might want one row in the crosstab to count all employees making between \$30,000 and \$40,000, the next row to count all employees making between \$40,000 and \$50,000, and so on.

❖ To cross-tabulate ranges of values:

- 1 Determine the expression that results in the raw values being converted into one of a small set of fixed values.

Each of those values will form a row or column in the crosstab.

- 2 Specify the expression in the Columns or Rows box in the Crosstab Definition dialog box.

You choose the box depending on whether you want the columns or rows to correspond to the range of values.

- 3 In the Values column, apply the appropriate aggregate function to the expression.

Example

This is best illustrated with an example.

You want to know how many employees in each department earn between \$30,000 and \$40,000, how many earn between \$40,000 and \$50,000, how many earn between \$50,000 and \$60,000, and so on. To do this, you want a crosstab where each row corresponds to a \$10,000 range of salary.

The first step is to determine the expression that, given a salary, returns the next smaller salary that is a multiple of \$10,000. For example, given a salary of \$34,000, the expression would return \$30,000, and given a salary of \$47,000, the expression would return \$40,000. You can use the `Int` function to accomplish this, as follows:

```
int (salary/10000) * 10000
```

That expression divides the salary by 10,000 and takes the integer portion, then multiplies the result by 10,000. So for \$34,000, the expression returns \$30,000, as follows:

```
34000/10000 = 3.4  
int (3.4) = 3  
3 * 10000 = 30000
```

With this information you can build the crosstab. The following uses the `Employee` table in the EAS Demo DB:

- 1 Build a crosstab and retrieve the `dept_id` and `salary` columns.
- 2 In the Crosstab Definition dialog box, drag the `dept_id` column to the Columns box.
- 3 Drag the `salary` column to the Rows box *and* to the Values box and edit the expressions.

In the Rows box, use:

```
int (salary/10000) * 10000
```

In the Values box, use:

```
count (int (salary/10000) * 10000 for crosstab)
```

For more on providing expressions in a crosstab, see “Using expressions” on page 340.

- 4 Click OK.

This is the result in the Design view:

Number of employees by department and salary		Total number of employees making the salary	
\$30,000 includes up to \$39,999			
Header [1] ↑			
		Department id	
	Salary display(@		
Header [2] ↑			
row_column	val	crosstabsum(1)	
Detail ↑			
Total number of employees in the department	sum(val for		
Summary ↑			
Footer ↑			

This is the crosstab at runtime:

Number of employees by department and salary							Total number of employees making the salary
\$30,000 includes up to \$39,999							
		Department id					
	Salary	100	200	300	400	500	
	\$20,000				2	5	7
	\$30,000	3	8	2	5	2	20
	\$40,000	6	5	2	5	1	19
	\$50,000	4	3	3	2	1	13
	\$60,000	4	1		2		7
	\$70,000	2	1	1			4
	\$80,000	2	1				3
	\$90,000	1					1
	\$130,000			1			1
	Total number of employees in the department	22	19	9	16	9	

You can see, for example, that 2 people in department 400 and 5 in department 500 earn between \$20,000 and \$30,000.

Displaying blank values as zero

In the preceding crosstab, several of the cells in the grid are blank. There are no employees in some salary ranges, so the value of those cells is null. To make the crosstab easier to read, you can add a display format to fields that can have null values so that they display a zero.

❖ To display blank values in a crosstab as zero:

- 1 Select the column you want to modify and click the Format tab in the Properties view.
- 2 Replace [General] in the Format box with ###0;###0;0;0.
The fourth section in the mask causes a null value to be represented as zero.

Creating static crosstabs

By default, crosstabs are dynamic: when you run them, DataWindow Designer retrieves the data and dynamically builds the columns and rows based on the retrieved data. For example, if you define a crosstab that computes sales of printers and a new printer type is entered in the database after you define the crosstab, you want the new printer to be in the crosstab. That is, you want DataWindow Designer to build the rows and columns dynamically based on current data, not the data that existed when the crosstab was defined.

Occasionally, however, you might want a crosstab to be static. That is, you want its columns to be established when you define the crosstab. You do not want additional columns to display in the crosstab at runtime; no matter what the data looks like, you do not want the number of columns to change. You want only the updated statistics for the predefined columns. The following procedure shows how to do that.

❖ To create a static crosstab:

- 1 In the wizard page or in the Crosstab Definition dialog box, clear the Rebuild Columns At Runtime check box.
- 2 Define the data for the crosstab as usual, and click OK.

What happens

With the check box cleared, instead of immediately building the crosstab's structure, DataWindow Designer first retrieves the data from the database. Using the retrieved data, DataWindow Designer then builds the crosstab structure and displays the workspace. It places all the values for the column specified in the Columns box in the workspace. These values become part of the crosstab's definition.

For example, in the following screenshot, the four values for Quarter (Q1, Q2, Q3, and Q4) are displayed in the Design view:

Sum Of Units	Quarter					
Header [1] ↑						
Product	Q1	Q2	Q3	Q4	Grand Total	
Header [2] ↑						
product	units	units_1	units_2	units_3	crosstabsum(1)	
Detail ↑						
"Grand Total"	sum(units	sum(units	sum(units	sum(units	sum(grand_sum_units for	
Summary ↑						
Footer ↑						

At runtime, no matter what values are in the database for the column, the crosstab shows only the values that were specified when the crosstab was defined. In the printer example, the crosstab always has the four columns it had when it was first defined.

Making changes

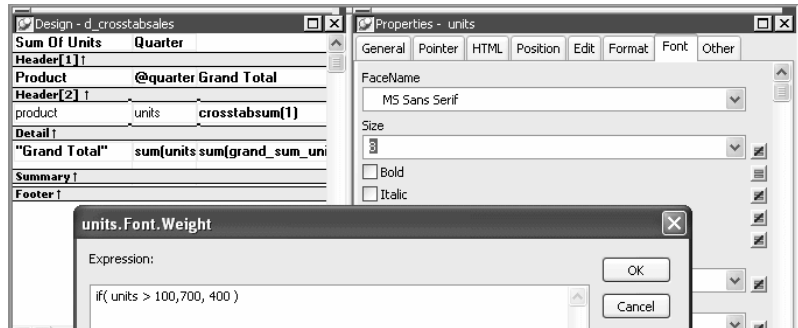
You can modify the properties of any of the columns in a static crosstab. You can modify the properties of each column individually, since each column is displayed in the workspace as part of the crosstab's definition. For example, in the printer crosstab you can directly modify the way values are presented in each individual quarter, since each quarter is represented in the Design view. (The values are shown as `units`, `units_1`, `units_2`, and `units_3`.)

Using property conditional expressions

As with other DataWindow objects, you can specify property conditional expressions to modify properties at runtime. You can use them with either dynamic or static crosstabs. With dynamic crosstabs, you specify an expression once for a column or value, and DataWindow Designer assigns the appropriate properties when it builds the individual columns at runtime. With static crosstabs, you have to specify an expression for each individual column or value, because the columns are already specified at definition time.

Example

In the following crosstab, an expression has been specified for Units:



The expression is for the Font.Weight property of the units column:

```
if (units > 100, 700, 400)
```

The expression specifies to use bold font (weight = 700) if the number of units is greater than 100. Otherwise, use normal font (weight = 400).

This is the crosstab at runtime:

Sum Of Units	Quarter				
Product	Q1	Q2	Q3	Q4	Grand Total
Cosmic	64	104	134	160	462
Galactic	9	21	13	12	55
Stellar	45	51	60	90	246
Grand Total	118	176	207	262	763

Values larger than 100 are shown in bold.

For more information about property conditional expressions, see Chapter 10, "Highlighting Information in DataWindow Objects."

About this chapter

This chapter describes how to build and use DataWindow objects in DataWindow Designer using the TreeView presentation style.

Contents

Topic	Page
TreeView presentation style	357
Creating a new TreeView DataWindow	359
Adding and deleting TreeView levels	363
Selecting a tree node and navigating the tree	364
Sorting rows in a TreeView DataWindow	365
TreeView DataWindow Design view	366
Setting properties for the TreeView DataWindow	367
TreeView DataWindow samples	371

TreeView presentation style

The TreeView presentation style provides an easy way to create DataWindow objects that display hierarchical data in a TreeView, where the rows are divided into groups that can be expanded and collapsed. A TreeView DataWindow displays a hierarchy of nodes, similar to the way the left pane of Windows Explorer displays folders and files.

In the TreeView DataWindow, each parent node contains other nodes called child nodes. You can display parent nodes—nodes that contain child nodes—in expanded or collapsed form.

With the TreeView DataWindow presentation style, you can group data in a hierarchy that allows users to browse the data and expand nodes to view details. Each TreeView level or node has an icon that users can click to expand or collapse the node.

You use the TreeView DataWindow wizard to create a TreeView DataWindow object. For information, see “Creating a new TreeView DataWindow” on page 359.

Example

This sample TreeView DataWindow uses the department and employee tables in the EAS Demo DB database and has two TreeView levels. The first level is the department name. The second level is the city where each employee resides. The detail data for each employee is grouped in TreeView leaf nodes under these two levels.

Employee ID	First Name	Last Name	Status	Salary	Start Date	Health Insurance	Sex
+ Finance		Total Employees: 5					
- Marketing		Total Employees: 4					
+ Concord							
1576	Scott	Evans	Active	\$68,940.00	12/30/1999	<input checked="" type="checkbox"/> Health Insurance	<input type="radio"/> M <input type="radio"/> F
207	Jane	Francis	Active	\$53,870.00	08/04/1998	<input checked="" type="checkbox"/> Health Insurance	<input type="radio"/> M <input type="radio"/> F
+ Needham							
+ Waltham							
- R & D		Total Employees: 13					
+ Belmont							
+ Burlington							
453	Andrew	Rabkin	Active	\$64,500.00	12/13/1992	<input checked="" type="checkbox"/> Health Insurance	<input type="radio"/> M <input type="radio"/> F
316	Lynn	Pastor	Active	\$74,500.00	10/24/1992	<input checked="" type="checkbox"/> Health Insurance	<input type="radio"/> M <input type="radio"/> F
+ Concord							
445	Kim	Lull	Active	\$87,900.00	12/13/1992	<input checked="" type="checkbox"/> Health Insurance	<input type="radio"/> M <input type="radio"/> F
+ Houston							
+ Lexington							
529	Dorothy	Sullivan	Active	\$67,890.00	08/03/1993	<input checked="" type="checkbox"/> Health Insurance	<input type="radio"/> M <input type="radio"/> F
+ Milton							
+ Natick							
+ Waltham							
+ Wellesley							
+ Sales		Total Employees: 6					
+ Shipping		Total Employees: 1					

Similarities to the Group presentation style

Creating and using a TreeView DataWindow is similar to creating and using a Group DataWindow. However, with the TreeView DataWindow, you can click the state icon to expand and collapse nodes.

The state icon in a TreeView DataWindow is a plus sign (+) when the node is collapsed and a minus sign (-) when the node is expanded. When a node is expanded, connecting lines display by default to show more detail and indicate how the parent data connects with the child data. When a node is collapsed, only the parent data displays; the detail data does not.

Creating a new TreeView DataWindow

You use the TreeView wizard and the DataWindow painter to create a TreeView DataWindow.

TreeView creation process

A TreeView DataWindow has multiple levels, each of which is a node in the TreeView. You use the TreeView wizard to create a TreeView DataWindow, but the wizard produces a DataWindow that includes only the top level of the TreeView.

Creating a complete TreeView DataWindow involves three steps:

- 1 Using the TreeView DataWindow wizard to create the top level (level 1) of the TreeView DataWindow.
- 2 Using the DataWindow painter to add additional levels to the TreeView DataWindow.
- 3 Setting TreeView DataWindow properties to customize the TreeView style.

For information about adding and deleting TreeView levels, see “Adding and deleting TreeView levels” on page 363. For information about setting properties in the DataWindow painter, see “Setting properties for the TreeView DataWindow” on page 367.

Creating a TreeView DataWindow

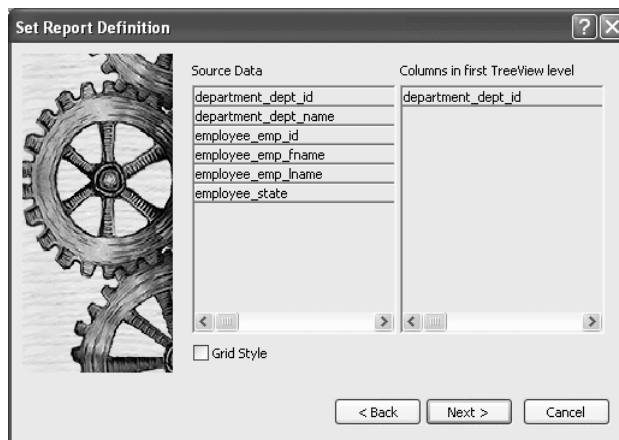
❖ **To create a TreeView DataWindow:**

- 1 Select File>New from the menu bar and select the DataWindow tab.
- 2 Choose the TreeView presentation style for the DataWindow and click OK.
- 3 Select the data source you want to use: Quick Select, SQL Select, Query, External, ADO DataSet, or Stored Procedure.
You are prompted to specify the data.
- 4 Define the tables and columns you want to use.
You are prompted to specify the TreeView grouping columns.

Multiple columns and multiple TreeView levels

You can specify more than one column, but all columns apply to TreeView level one. At this point, you can define only one TreeView level. You define additional levels later.

In the following example, TreeView grouping will be by department, as specified by the dept_id column:



If you want to use an expression, you can define it when you have completed the wizard. See “Using an expression for a column name” on page 362.

The sample DataWindow shown in “Example” on page 358 uses the department and employee tables in the EAS Demo DB database.

- 5 Specify the column or columns that will be at the top level (level 1) of the TreeView DataWindow.

The sample DataWindow uses the department name as the top level. If you want to display both the department ID and department name, you specify that both columns are at the top level.

- 6 If you want the TreeView DataWindow to display grid lines, select the Grid Style check box.

When you select the Grid Style check box, the TreeView DataWindow displays grid lines for rows and columns. You can drag the grid lines to resize rows and columns.

- 7 Click Next.

- 8 Modify the default color and border settings if needed, and then click Next.
- 9 Review the TreeView DataWindow characteristics.
- 10 Click Finish.

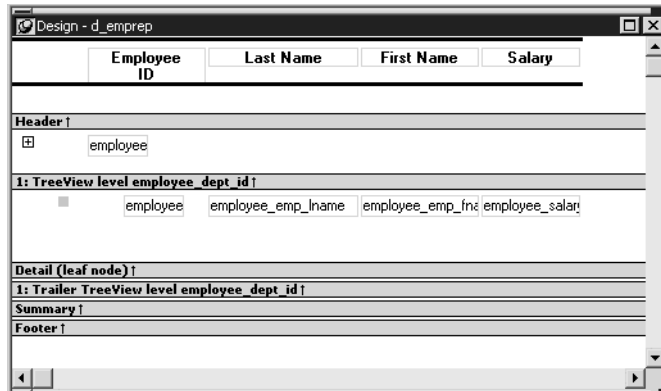
The DataWindow painter Design view displays. For information about the Design view, see “TreeView DataWindow Design view” on page 366. For information about adding additional levels, see “Adding and deleting TreeView levels” on page 363.

What DataWindow Designer does

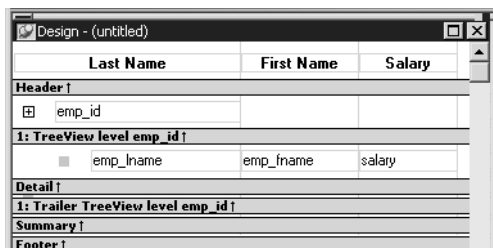
As a result of your specifications, DataWindow Designer generates a TreeView DataWindow object and creates:

- A TreeView header band with controls that include the heading text of the detail band columns
- The first TreeView level band with the TreeView level columns you chose in the wizard
- The detail (leaf node) band that includes all the column controls except for first-level columns you selected in the wizard
- A level 1 trailer band.
- A summary band, and a footer band.

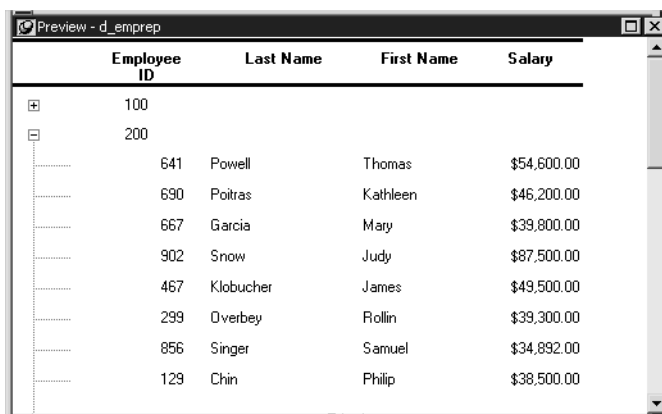
Here is the sample TreeView DataWindow object in the Design view:



If you selected the Grid Style check box, vertical and horizontal grid lines display:



Here is the sample TreeView DataWindow object in the Preview view:



Using an expression for a column name

If you want to use an expression for one or more column names in a TreeView, you can enter it as the TreeView definition on the General page in the Properties view after you finish using the TreeView wizard.

❖ **To use an expression for a TreeView column name:**

- 1 Open the Properties view and click the TreeView level band in the Design view.
- 2 Click the ellipsis button next to the TreeView Level Definition box on the General page in the Properties view to open the Specify Group Columns dialog box.
- 3 In the Columns box, double-click the column you want to use in an expression.

The Modify Expression dialog box opens. You can specify more than one grouping item expression for a group. A break occurs whenever the value concatenated from each column/expression changes.

What you can do

All of the techniques available in a tabular DataWindow object, such as moving controls and specifying display formats, are available for modifying and enhancing TreeView DataWindow objects. See "Adding and deleting TreeView levels" next to read more about the bands in a TreeView DataWindow object and see how to add features especially suited for TreeView DataWindow objects, such as additional TreeView levels or summary statistics.

DataWindow object is not updatable by default

When you generate a DataWindow object using the TreeView presentation style, DataWindow Designer makes it not updatable by default. If you want to be able to update the database through the TreeView DataWindow object, you must modify its update characteristics. For more information, see Chapter 6, "Controlling Updates in DataWindow Objects."

Adding and deleting TreeView levels

You add and delete TreeView levels using the Rows menu in the DataWindow painter.

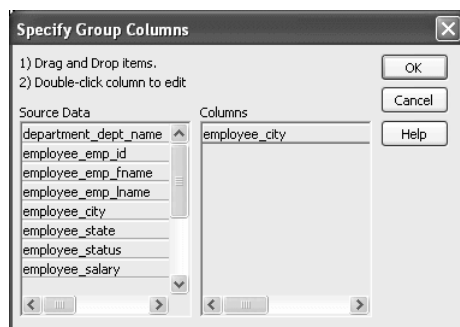
❖ To create an additional level in a TreeView DataWindow:

- 1 Open the TreeView DataWindow if it is not already open.
- 2 Select Rows>Create TreeView Level from the menu bar.

The Specify Group Columns dialog box displays.

- 3 Specify the columns you want to set as the next TreeView level by dragging them from the Source Data pane to the Columns pane.

In the sample DataWindow shown in “Example” on page 358, the second level has a single column, the `employee_city` column.



- 4 Click OK.

The new TreeView level and a Trailer band for that level are created in the TreeView Design view. For information on how to set properties for a TreeView level, see “Setting TreeView level properties” on page 369.

❖ **To delete a level in a TreeView DataWindow:**

- 1 Select Rows>Delete TreeView Level from the menu bar.
- 2 Select the number of the level to delete from the list of levels that displays.
The level in the TreeView DataWindow is deleted immediately.

If you delete a level by mistake

If you unintentionally delete a level, close the TreeView DataWindow without saving changes, then reopen it and continue working.

Selecting a tree node and navigating the tree

You can select a tree node in the TreeView DataWindow by setting the Select Node By Mouse property to true and then clicking a tree node to select it with the mouse.

After you select a tree node in the TreeView DataWindow, you can navigate the tree using the up, down, left, and right keys.

Use this key	To do this
Up	Select a tree node prior to the currently selected node.
Down	Select a tree node next to the currently selected node.
Left	Collapse the currently selected node. If the current tree node is a leaf node or the node has been collapsed, the DataWindow just scrolls to the left, which is its normal behavior.
Right	Expand the currently selected node. If the current tree node is a leaf node or the node has been expanded, the DataWindow just scrolls to the right, which is its normal behavior.

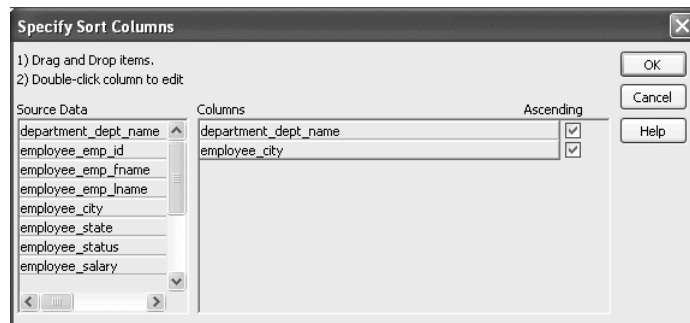
Sorting rows in a TreeView DataWindow

❖ To sort the rows within levels in a TreeView DataWindow:

- 1 Select Rows>Sort from the menu bar.
- 2 Drag the columns that you want to sort the rows on from the Source Data box to the Columns box.

The order of the columns determines the precedence of the sort. The sort order is ascending by default. To sort in descending order, clear the Ascending check box.

For example, the sample DataWindow shown in “Example” on page 358 has department name as the first level and the employee’s city of residence as the second level.

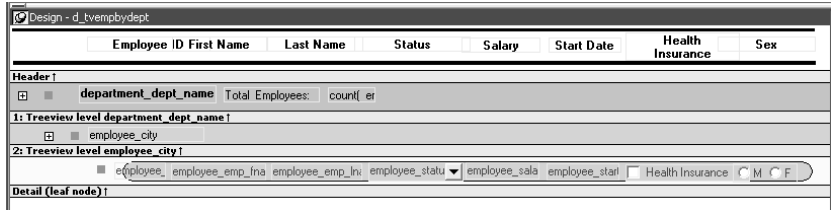


Other actions you can take

To reorder the columns, drag them up or down in the list. To delete a column from the sort columns list, drag the column outside the dialog box. To specify an expression to sort on, double-click a column name in the Columns box and modify the expression in the Modify Expression dialog box.

TreeView DataWindow Design view

The Design view for the TreeView DataWindow differs from the traditional Design view for most DataWindow presentation styles.



The Design view has a header band, a TreeView level band for each added level, a detail band, a Trailer band for each level, a summary band, and a footer band.

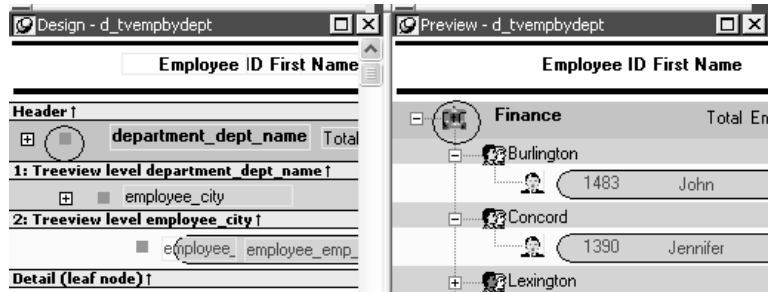
By default, the controls in the header band are the heading text of the detail band columns, and the controls in the detail (leaf node) band are all the column controls except for the first-level columns (in the 1:Treeview level band) that you selected when you used the TreeView wizard. Columns that you specify as additional levels remain in the detail band.

The minimum height of each TreeView level band is the height of the tree node icon.

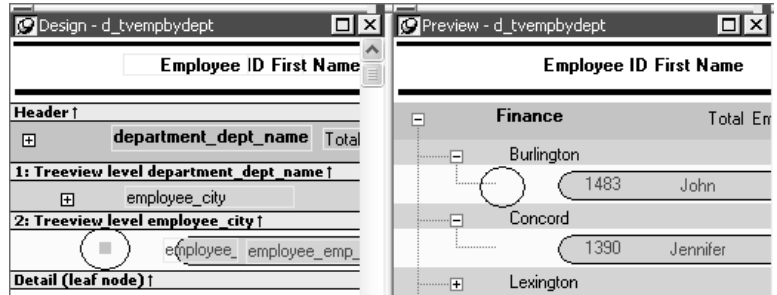
Icons in the Design view

There are three icons in the Design view that represent the locations of nodes, icons, and connecting lines in the tree to help you design the DataWindow. Columns must always display to the right of the state and tree node icons:

- A square icon with a plus sign (+) in each TreeView level band represents the position of the state icon, the icon that indicates whether a node is expanded or collapsed. On the XP platform, the plus (+) and minus (-) icons have the Windows XP style.
- A shaded square icon in the detail band and in each TreeView level band represents the position of the image you specify as a tree node icon.



- When there is no tree node icon specified, a shaded square icon in the detail band and in each TreeView level band represents where the connecting line ends.



The position of all the icons changes when you change the indent value.

For more information about specifying icons and the indent value, see “Setting properties for the TreeView DataWindow.”

Setting properties for the TreeView DataWindow

You can set three types of properties for the TreeView DataWindow:

- General properties
- TreeView level properties
- Detail band properties

Specifying images for tree node icons

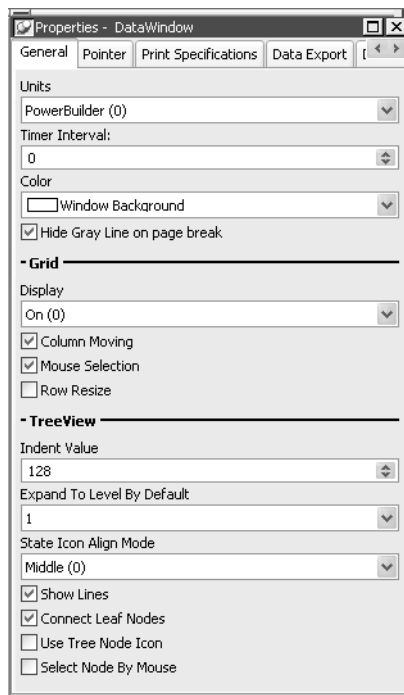
In the sample DataWindow shown in “Creating a new TreeView DataWindow” on page 359, different tree node icons display for collapsed and expanded levels. The icons are also different for each level. You specify images for these icons as TreeView level band properties.

The sample DataWindow also displays a tree node icon next to every row in the detail band. You specify an image for this icon as a detail band property.

Tree node icons do not display by default. After specifying images for icons, select the Use Tree Node Icon general property.

Setting general TreeView properties

You set most TreeView DataWindow properties on the General page in the DataWindow Properties view.



The properties that are specific to a TreeView DataWindow are the TreeView properties and the Grid properties in the General page in the Properties view. The Grid properties display in the Properties view only if you select the Grid Style check box when you define the TreeView DataWindow:

Property	Description
Display On (0)	Grid lines always display.
Display Off (1)	Grid lines never display, so users cannot resize columns at runtime.
Display Only (2)	Grid lines display only when the DataWindow object displays online.
Display Print Only (3)	Grid lines display only when the contents of the DataWindow object are printed.
Column Moving	Columns can be moved at runtime.
Mouse Selection	Data can be selected at runtime and, for example, copied to the clipboard.

Property	Description
Row Resize	Rows can be resized at runtime.
Indent Value	The indent value of the child node from its parent in the units specified for the DataWindow. The indent value defines the position of the state icon. The X position of the state icon is the X position of its parent plus the indent value.
Expand To Level By Default	Expand to TreeView level 1, 2, or 3.
State Icon Align Mode	Align the state icon in the middle (0), at the top (1), or at the bottom (2).
Show Lines	Whether lines display that connect parent nodes and child nodes. If you want to display lines that connect the rows in the detail band to their parent, select Connect Leaf Nodes.
Connect Leaf Nodes	Whether lines display that connect the leaf nodes in the detail band rows.
Use Tree Node Icon	Whether an icon for the tree node displays. This applies to icons in the level and detail bands. For how to specify icon images, see "Setting TreeView level properties" and "Setting detail band properties" next.
Select Node By Mouse	Whether a Tree node is selected by clicking the Tree node with the mouse.

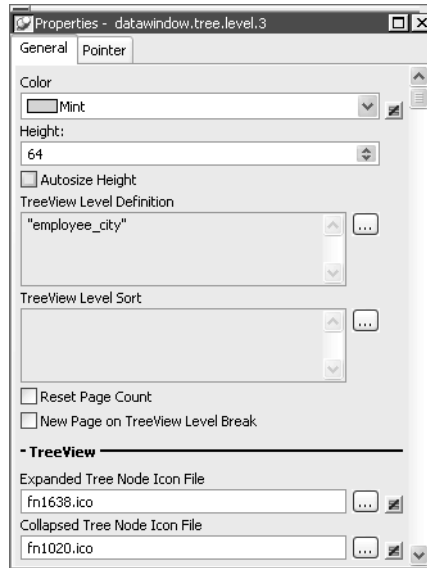
Setting TreeView level properties

In the Properties view, you can specify expanded and collapsed icons for each TreeView level. You access the Properties view by clicking the bar identifying the band for that level in the Design view in the DataWindow painter. You can also access the Properties view from the Rows menu, or by clicking any of the icons in the Design view that represent the locations of nodes, icons, and connecting lines. (See "Icons in the Design view" on page 366.)

❖ To modify properties for a level in a TreeView DataWindow:

- 1 Select Rows>Edit TreeView Level from the menu bar and then select the number of the level from the list of levels, or click the bar identifying the band for that level or any of the icons in that band.

- 2 Use the DataWindow TreeView Level properties view that displays to edit the properties for the level you selected.



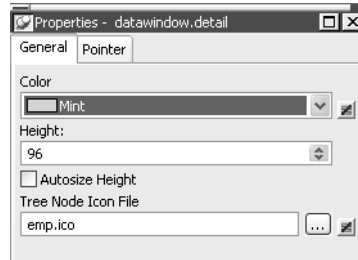
The properties that are specific to a TreeView level band are at the bottom of the Properties view:

Property	Description
Expanded Tree Node Icon File	The file name of the tree node icon in a TreeView level band when it is in the expanded state.
Collapsed Tree Node Icon File	The file name of the tree node icon in a TreeView level band when it is in the collapsed state.

You set the tree node icon file name separately for each TreeView level band. You can use a quoted expression for the tree node icon file.

Setting detail band properties

You can specify an icon for the rows in the detail band by clicking the detail band in the DataWindow painter to display the Properties view.



The only property that is specific to the detail band is located at the bottom of the Properties view:

Property	Description
Tree Node Icon File	The file name of the tree node icon in the detail band. You can use a quoted expression.

TreeView DataWindow samples

This section provides you with two samples that demonstrate use of the TreeView DataWindow.

What the samples do

The two TreeView DataWindow samples are:

- Data Explorer

The Data Explorer uses a TreeView DataWindow to display sales-related data in a Windows Explorer-like interface and allows users to update the data.

- Data Linker

The Data Linker uses a TreeView DataWindow on the left for data navigation, linked to four DataWindows on the right for updating the data. The Data Linker demonstrates populating a TreeView DataWindow with data and linking each TreeView level to a separate DataWindow.

Tables and database

Both samples use the employee, sales_order, sales_order_items, customer, and product tables in the EAS Demo DB database.

TreeView DataWindows

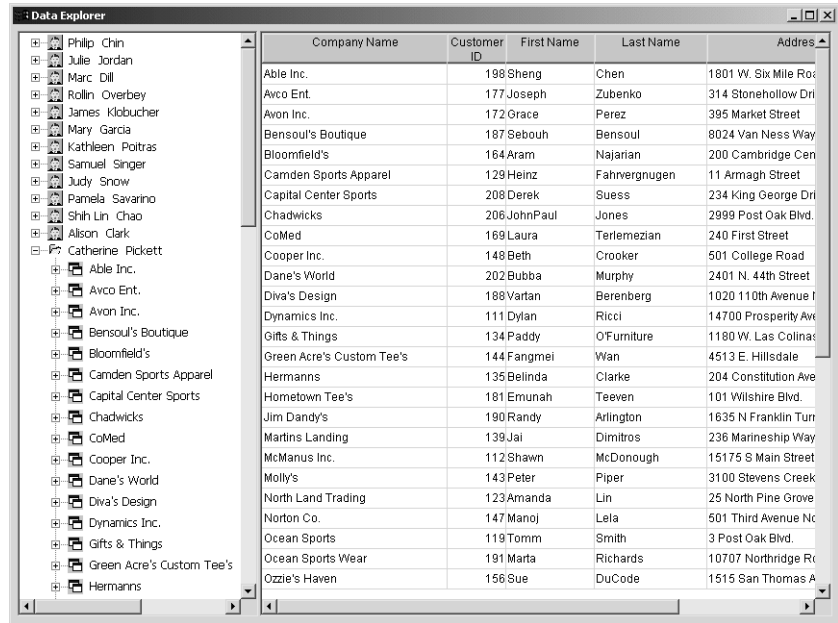
The TreeView DataWindows are *d_sales_report* and *d_sales_report2*. Each TreeView DataWindow has three TreeView levels:

- The first level (level 1) is the sales representative's name.
You create the first level using the TreeView DataWindow wizard.
- The second level (level 2) is the name of the customer's company.
You create the second level using the Rows>Create TreeView Level menu item in the DataWindow painter.
- The third level (level 3) is the sales order ID.
You also create the third level using the Rows>Create TreeView Level menu item in the DataWindow painter.

Data Explorer sample

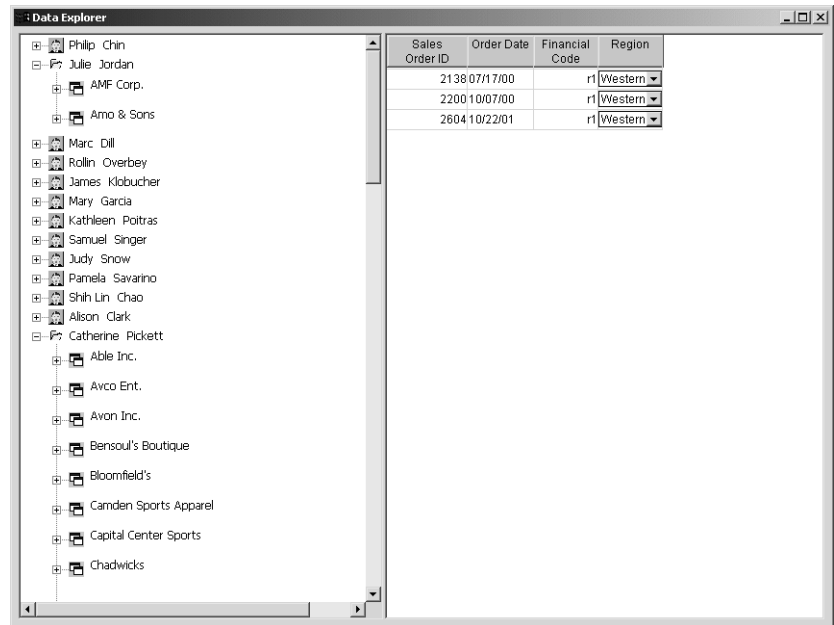
What the Data Explorer does

Clicking on each TreeView level displays details in a DataWindow on the right. For example, if you click a name in the TreeView DataWindow on the left, detailed customer data displays in the DataWindow on the right.

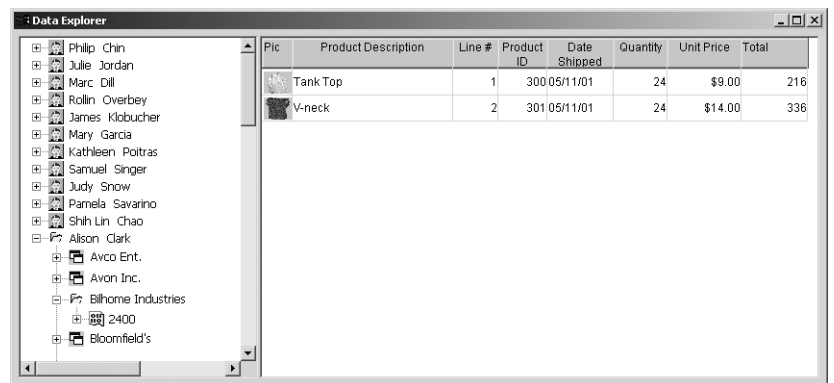


Using the Data Explorer

You can click on any TreeView level in the Data Explorer. If you click a company name in the TreeView DataWindow on the left (for example, Able Inc., under Catherine Pickett), order information displays on the right.

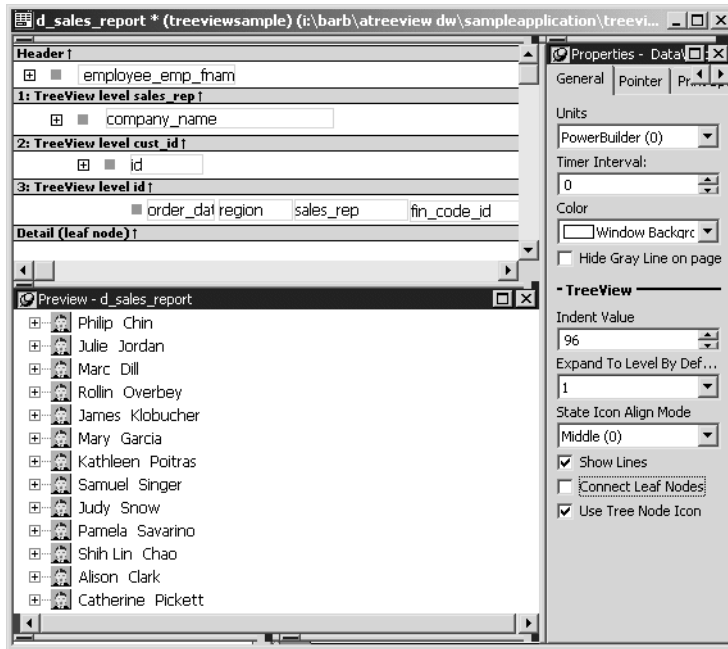


If you click an order ID in the TreeView DataWindow on the left (for example, order ID 2400, under Bilhome Industries, under Alison Clark), the customer order information displays on the right.



Data Explorer
TreeView
DataWindow

Here is the TreeView DataWindow used in the Data Explorer.



One TreeView DataWindow

The Data Explorer uses one TreeView DataWindow, but DataWindows that are not TreeView DataWindows also support the Data Explorer's functionality.

Data Explorer code

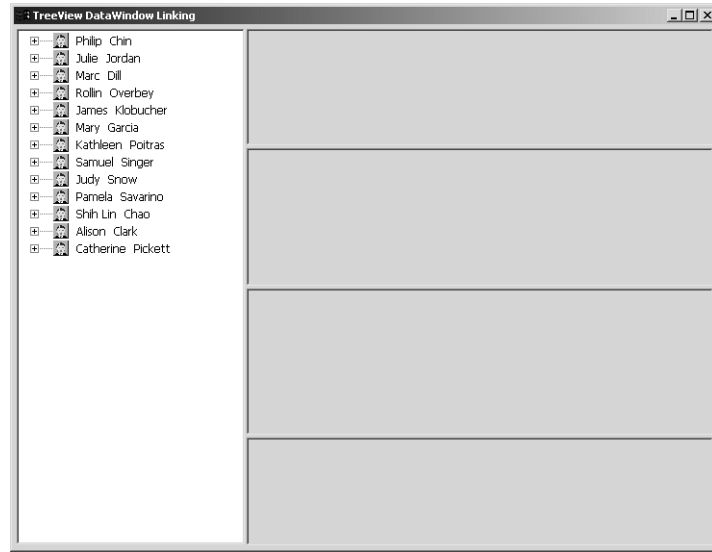
The code in the Clicked event uses `GetBandAtPointer` to determine which DataWindow to display. Clicking on some editable items in the detail DataWindow opens a window in which you can manipulate the data.

The `PopupMenu` menu object has two menu items that call the `CollapseAll` and `ExpandAll` methods to collapse or expand all the nodes in the TreeView.

Data Linker sample

What the Data Linker does

When you first run the Data Linker, no data displays on the right side of the window.



Using the Data Linker

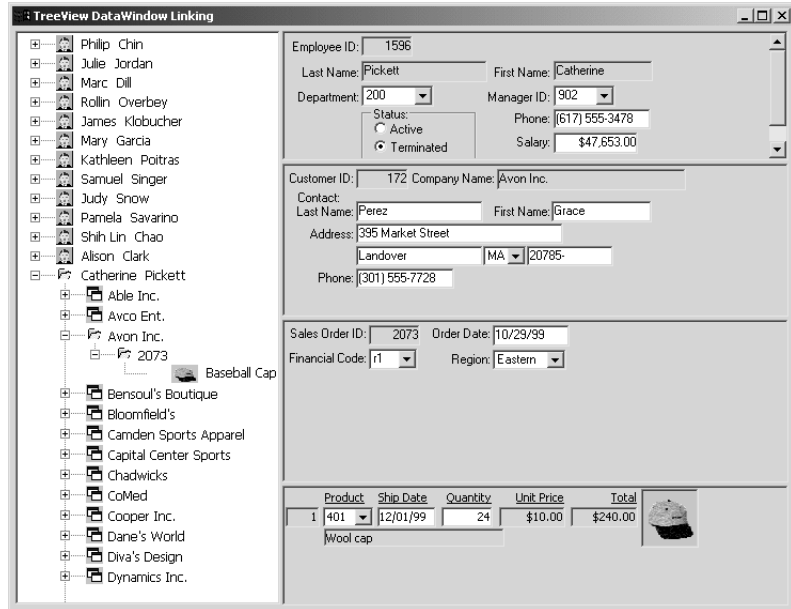
To use the Data Linker, you first expand an employee name and a company's data in the TreeView DataWindow.



Expanding the TreeView displays the company names, the orders for the company you select, and in the detail band, the icon and name for each item in the order.

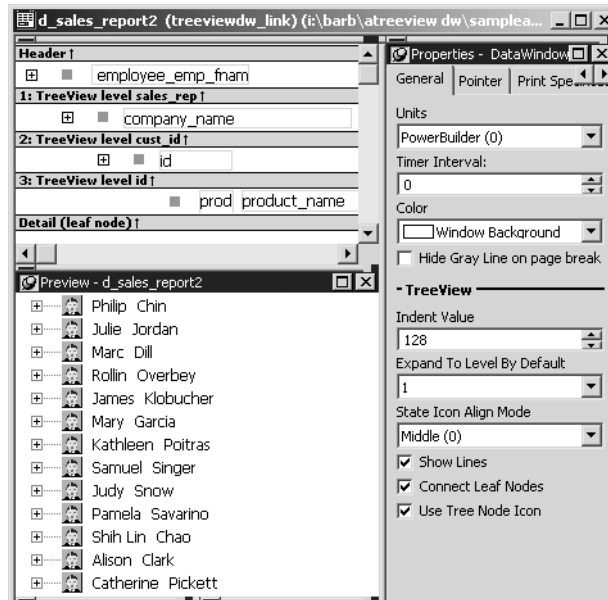
You can click on each of the TreeView levels in order, and then click in the detail band to display the details in the four DataWindows on the right.

For example, if you click first on Catherine Pickett, then on Avon Inc., then on 2073, and last on Baseball Cap, the data in each of the related DataWindows displays on the right. You can also update the data in each of the DataWindows.



Data Linker TreeView
DataWindow

Here is the TreeView DataWindow used in the Data Linker sample.

**One TreeView DataWindow**

The Data Linker uses one TreeView DataWindow, but other DataWindows that are not TreeView DataWindows also support the Data Linker's functionality.

Data Linker code

The code in the Clicked event uses `GetBandAtPointer` to determine which DataWindow to display.

Exporting and Importing XML Data

About this chapter

The row data in a DataWindow can be exported and imported in the Extensible Markup Language (XML). This chapter describes how to create and use templates that control the export and import of data in XML format.

Contents

Topic	Page
About XML	379
XML support in the DataWindow painter	383
The Export/Import Template view for XML	384
Editing XML templates	390
Exporting to XML	398
Importing XML	408

About XML

Like Hypertext Markup Language (HTML), Extensible Markup Language (XML) is a subset of Standardized General Markup Language (SGML) and has been designed specifically for use on the Web. XML is defined in the W3C Recommendation published by the World Wide Web Consortium. The latest version of this document is available at <http://www.w3.org/TR/REC-xml>.

XML is more complete and disciplined than HTML, and it is also a framework for creating markup languages—it allows you to define your own application-oriented markup tags.

XML provides a set of rules for structuring data. Like HTML, XML uses tags and attributes, but the tags are used to delimit pieces of data, allowing the application that receives the data to interpret the meaning of each tag. These properties make XML particularly suitable for data interchange across applications, platforms, enterprises, and the Web. The data can be structured in a hierarchy that includes nesting.

An XML document is made up of declarations, elements, comments, character references, and processing instructions, indicated in the document by explicit markup.

The simple XML document that follows contains an XML declaration followed by the start tag of the root element, `<d_dept_list>`, nested row and column elements, and finally the end tag of the root element. The root element is the starting point for the XML processor.

```
<?xml version="1.0">
<d_dept_list>
  <d_dept_list_row>
    <dept_id>100</dept_id>
    <dept_name>R &D</dept_name>
    <dept_head_id>501</dept_head_id>
  </d_dept_list_row>
  ...
</d_dept_list>
```

This section contains a brief overview of XML rules and syntax. For a good introduction to XML, see XML in 10 points at <http://www.w3.org/XML/1999/XML-in-10-points>. For more detailed information, see the W3C XML page at <http://www.w3.org/XML/>, the O'Reilly page XML from the inside out at <http://www.xml.com>, the XML Cover Pages at <http://xml.coverpages.org/xml.html>, or one of the many books about XML.

Valid and well-formed XML documents

An XML document must be valid, well-formed, or both.

Valid documents

To define a set of tags for use in a particular application, XML uses a separate document named a document type definition (DTD). A DTD states what tags are allowed in an XML document and defines rules for how those tags can be used in relation to each other. It defines the elements that are allowed in the language, the attributes each element can have, and the type of information each element can hold. Documents can be verified against a DTD to ensure that they follow all the rules of the language. A document that satisfies a DTD is said to be valid.

If a document uses a DTD, the DTD must immediately follow the declaration.

XML Schema provides an alternative mechanism for describing and validating XML data. It provides a richer set of datatypes than a DTD, as well as support for namespaces, including the ability to use prefixes in instance documents and accept unknown elements and attributes from known or unknown namespaces. For more information, see the W3C XML Schema page at <http://www.w3.org/XML/Schema>.

Well-formed documents

The second way to specify XML syntax is to assume that a document is using its language properly. XML provides a set of generic syntax rules that must be satisfied, and as long as a document satisfies these rules, it is said to be well-formed. All valid documents must be well-formed.

Processing well-formed documents is faster than processing valid documents because the parser does not have to verify against the DTD or XML schema. When valid documents are transmitted, the DTD or XML schema must also be transmitted if the receiver does not already possess it. Well-formed documents can be sent without other information.

XML documents should conform to a DTD or XML schema if they are going to be used by more than one application. If they are not valid, there is no way to guarantee that various applications will be able to understand each other.

XML syntax

There are a few more restrictions on XML than on HTML; they make parsing of XML simpler.

Tags cannot be omitted

Unlike HTML, XML does not allow you to omit tags. This guarantees that parsers know where elements end.

The following example is acceptable HTML, but not XML:

```
<table>
  <tr>
    <td>Dog</td>
    <td>Cat
    <td>Mouse
  </table>
```

To change this into well-formed XML, you need to add all the missing end tags:

```
<table>
  <tr>
    <td>Dog</td>
    <td>Cat</td>
    <td>Mouse</td>
  </tr>
</table>
```

Representing empty elements

Empty elements cannot be represented in XML in the same way they are in HTML. An empty element is one that is not used to mark up data, so in HTML, there is no end tag. There are two ways to handle empty elements:

- Place a dummy tag immediately after the start tag. For example:

```
<img href="picture.jpg"></img>
```

- Use a slash character at the end of the initial tag:

```
<img href="picture.jpg"/>
```

This tells a parser that the element consists only of one tag.

XML is case sensitive

XML is case sensitive, which allows it to be used with non-Latin alphabets. You must ensure that letter case matches in start and end tags: `<MyTag>` and `</Mytag>` belong to two different elements.

White space

White space within tags in XML is unchanged by parsers.

All elements must be nested.

All XML elements must be properly nested. All child elements must be closed before their parent elements close.

XML parsing

There are two major types of application programming interfaces (APIs) that can be used to parse XML:

- Tree-based APIs map the XML document to a tree structure. The major tree-based API is the Document Object Model (DOM) maintained by W3C. A DOM parser is particularly useful if you are working with a deeply-nested document that must be traversed multiple times.

For more information about the DOM parser, see the W3C Document Object Model page at <http://www.w3c.org/DOM>.

- Event-based APIs use callbacks to report events, such as the start and end of elements, to the calling application, and the application handles those events. These APIs provide faster, lower-level access to the XML and are most efficient when extracting data from an XML document in a single traversal.

For more information about the best-known event-driven parser, SAX (Simple API for XML), see the SAX page at <http://sax.sourceforge.net/>.

Xerces parser

DataWindow Designer includes software developed by the Apache Software Foundation (<http://www.apache.org/>). The XML services for DataWindow objects are built on the Apache Xerces-C++ parser, which conforms to both DOM and SAX specifications. For more information about SAX, see the Xerces C++ Parser page at <http://xml.apache.org/xerces-c/index.html>.

XML support in the DataWindow painter

DataWindow Designer supports both the export and import of XML in DataStore and DataWindow objects using XML template objects. You construct XML templates for export and import graphically in the Export/Import Template view for XML. Each template you create is encapsulated in the DataWindow object. A template enables you to specify the XML logical structure of how the row data iterates inside the root element of the XML document.

Export templates

An XML export template lets you customize the XML that is generated.

You can specify optional XML and document type declarations that precede the root element in the exported XML, as well as the logical structure and nesting level of iterative DataWindow row data inside the root element. The children of the root element can contain elements, character references, and processing instructions as well as the row data, using explicit markup. For more information, see “Header and Detail sections” on page 387.

If the exported XML is used by different applications or processes, you can define a separate export template for each use.

Import templates

You need to create an import template if you want to import data that does not match the DataWindow column definition or is associated with a schema, or if you want to import attribute values.

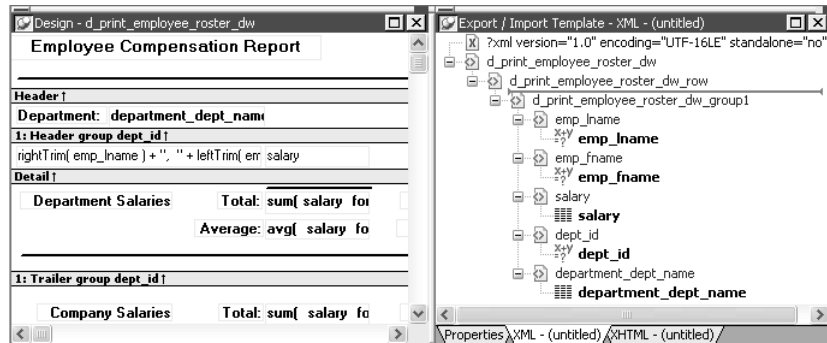
Only the mapping of column names to element and attribute names is used for import. All other information in the template is ignored.

The Export/Import Template view for XML

You define and edit templates for export and import in the Export/Import Template view for XML in the DataWindow painter. The view uses a tree view to represent the template.

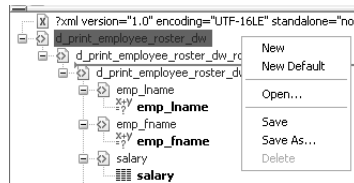
When you create a new DataWindow object, DataWindow Designer displays a default template in the Export/Import Template view. You can edit only one template at a time in the view, but you can create multiple templates and save them with the DataWindow object. Each template is uniquely associated with the DataWindow object open in the painter.

The default template has one element for each column in the DataWindow object.



Creating, opening, and saving templates

From the pop-up menu for the Export/Import Template view (with nothing selected), you can create new templates with or without default contents, open an existing template, save the current template, or delete the current template. You can only open and edit templates that are associated with the current DataWindow object.














Representing tree view items

Each item in the template displays as a single tree view item with an image and font color that denotes its type. Elements are represented by a yellow icon that resembles a luggage tag. The end tags of elements and the markup delimiters used in an XML document do not display.

Table 14-1 shows the icons used in the Export/Import Template view.

Table 14-1: Icons used in the Export/Import Template view

Icon	Description
	XML declaration or document type declaration
	Root or child element
	Group header element
	DataWindow column reference
	Static text control reference
	Computed field or DataWindow expression reference
	Literal text
	Comment
	Processing instruction
	CDATA section
	Nested report

Creating templates

Creating new base templates

To create a template, select the New menu item or the New Default menu item from the pop-up menu in the Export/Import Template view.

The New menu item creates a template that is empty except for the XML declaration, the root element, and the first element of the row data section, referred to as the Detail Start element. The name of the root element is the same as the name of the DataWindow object, and the default name for the Detail Start element is the name of the root element with `_row` appended.

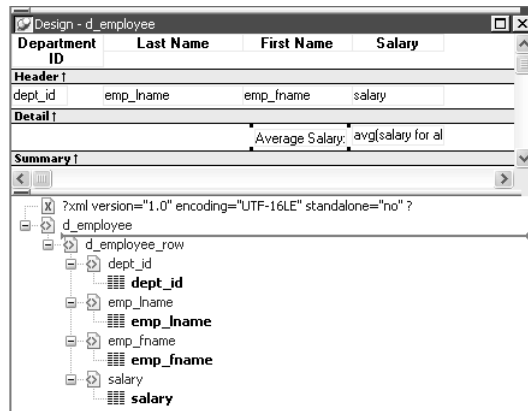
For example, if the DataWindow object is named `d_name`, the default template has this structure:

```
<?xml version="1.0"?>
<d_name>
  <d_name_row>
  </d_name_row>
</d_name>
```

Creating new default templates

The New Default menu item creates a template with the same contents as the New menu item, as well as a flat structure of child elements of the Detail Start element. A child element is created for each DataWindow column name, in the order in which the columns appear in the SELECT statement, with the exception of blob and computed columns. The default tag for the element is the column's name.

If the names of the column and the control are the same, the content of the child element displays with a control reference icon. If there is no control name that matches the column name, the content of the child element displays using the DataWindow expression icon. For example, consider a DataWindow object in which the dept_id column is used as a retrieval argument and does not display:



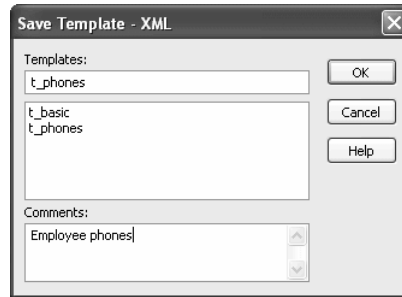
The SQL syntax is:

```
SELECT "employee"."dept_id",
       "employee"."emp_lname",
       "employee"."emp_fname",
       "employee"."salary"
FROM "employee"
WHERE employee.dept_id = :deptnum
ORDER BY "employee"."emp_lname" ASC
```

In the default template, dept_id uses the DataWindow expression icon. All the other columns used the column control reference icon.

Saving templates

To save a new template, select Save from the pop-up menu in the Export/Import Template view, and give the template a name and optionally a comment that identifies its use.



The template is stored inside the DataWindow object in the PBL. After saving a template with a DataWindow object, you can see its definition in the Source editor for the DataWindow object. For example, this is part of the source for a DataWindow that has two templates. The templates have required elements only:

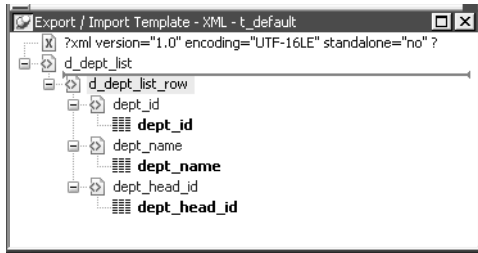
```
export.xml (usetemplate="t_address"
  template=(comment="Employee Phone Book"
    name="t_phone" xml="<d_emplist><d_emplist_row
      __pbband=~"detail~"/></d_emplist>")
  template=(comment="Employee Address Book"
    name="t_address" xml="<d_emplist><d_emplist_row
      __pbband=~"detail~"/></d_emplist>"))
```

Header and Detail sections

An XML template has a Header section and a Detail section, separated graphically by a line across the tree view.

The items in the Header section are generated only once when the DataWindow is exported to XML, unless the DataWindow is a group DataWindow. For group DataWindow objects, you can choose to generate the contents of the header section iteratively for each group. For more information, see “Generating group headers” on page 400.

The Detail section contains the row data, and is generated iteratively for each row in the DataWindow object.



The Detail Start element

A line across the Export/Import Template view separates the Header section from the Detail section. The first element after this line, `d_dept_list_row` in the previous screenshot, is called the Detail Start element.

There can be only one Detail Start element, and it must be inside the document's root element. By default, the first child of the root element is the Detail Start element. It usually wraps a whole row, separating columns across rows. When the DataWindow is exported to XML, this element and all children and/or siblings after it are generated iteratively for each row. Any elements in the root element above the separator line are generated only once, unless the DataWindow is a group DataWindow and the Iterate Group Headers check box has been selected.

The Detail Start element can be a nested (or multiply-nested) child of an element from the Header section, permitting a nested detail.

Moving the separator

You can change the location of the separator line by selecting the element that you want as the Detail Start element and selecting Starts Detail from its pop-up menu. The separator line is redrawn above the new Detail Start element. When you export the data, the Detail Start element and the children and siblings after it are generated iteratively for each row.

If no Detail Start element is specified (that is, if the Starts Detail option has been deselected), the template has only a Header section. When you export the data, only one iteration of row data is generated.

Header section

The Header section can contain the items listed in Table 14-2. Only the root element is required:

Table 14-2: Items permitted in the Header section of an XML document

Item	Details
XML declaration	This must be the first item in the tree view if it exists. See “XML declaration” on page 391.
Document type declaration	If there is an XML declaration, the document type declaration must appear after the XML declaration and any optional processing instructions and comments, and before the root element. Otherwise, this must be the first item in the tree view. See “Document type declaration” on page 392.
Comments	See “Comments” on page 397.
Processing instructions	See “Processing instructions” on page 398.
Root element (start tag)	See “Root element” on page 393.
Group header elements	See “Generating group headers” on page 400.
Child elements	Child elements in the Header section cannot be iterative except in the case of group DataWindows.

Detail section in root element

The root element displays in the Header section, but the entire content of the Detail section is contained in the root element.

Detail section

The Detail section, which holds the row data, can contain the items listed in Table 14-3.

Table 14-3: Items permitted in the Detail section of an XML document

Item	Details
Detail Start element	See “The Detail Start element” on page 388.
Child or sibling elements to the Detail Start element	To add a sibling to the Detail Start element, add a child to its parent (the root element by default).
Control references	These references are in text format and can include references to column, text, computed field, and report controls. See “Controls” on page 394. Nested report controls can only be referenced as child elements. See “Composite and nested reports” on page 395.

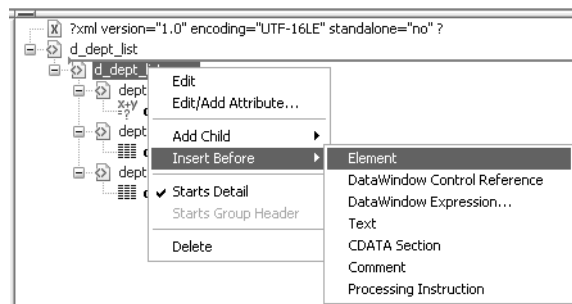
Item	Details
DataWindow expressions	See “DataWindow expressions” on page 394.
Literal text	Literal text does not correspond to a control in the DataWindow object.
Comments	See “Comments” on page 397.
Processing instructions	See “Processing instructions” on page 398.
CDATA sections	See “CDATA sections” on page 397.
Attributes	You can assign attributes to all element types. See “Attributes” on page 395.

Editing XML templates

Using templates for data import

If you use a template created for data export, DataWindow expressions, text, comments, and processing instructions are ignored when data is imported. If you are creating a template specifically for import, do not add any of these items. You need only map column names to element and attribute names.

Every item in the Export/Import Template view has a pop-up menu from which you can perform actions appropriate to that item, such as editing or deleting the item, adding or editing attributes, adding child elements or other items, and inserting elements, processing instructions, CDATA sections, and so forth, before the current item.



If an element has no attributes, you can edit its tag in the Export/Import Template view by selecting it and left-clicking the tag or pressing F2. Literal text nodes can be edited in the same way. You can delete items (and their children) by pressing the Delete key.

The examples in this section show the delimiters used in the XML document. When you edit the template in dialog boxes opened from the Export/Import Template view for XML, you do not need to type these delimiters in text boxes.

The rest of this section describes some of the items in the template. For more information, see the XML specification at <http://www.w3.org/TR/REC-xml>.

XML declaration

The XML declaration specifies the version of XML being used. You may need to change this value for a future version of XML. It can also contain an encoding declaration and a standalone document declaration. From the pop-up menu, you can edit the declaration, and, if the document is well-formed, delete it. If you have deleted the XML declaration, you can insert one from the Insert Before item on the pop-up menu for the next item in the template.

Encoding declaration

The encoding declaration specifies the character-set encoding used in the document, such as UTF-16 or ISO-10646-UCS-4.

If there is no encoding declaration, the value defaults to UTF-16LE encoding in ASCII environments. In DBCS environments, the default is the default system encoding on the computer where the XML document is generated. This ensures that the document displays correctly as a plain text file. However, since the DBCS data is serialized to Unicode, XML documents that use UTF-16LE, UTF-16 Big Endian, or UTF-16 Little Endian can all be parsed or generated correctly on DBCS systems.

Several other encodings are available, including ASCII, UCS4 Big Endian, UCS4 Little Endian, EBCDIC code pages IBM037 and IBM1140, ISO Latin-1, and Latin 1 Windows (code page 1252). You can select these values from a drop-down list box in the XML Declaration dialog box.

Standalone document declaration

The standalone document declaration specifies whether the document contains no external markup that needs to be processed and can therefore stand alone (Yes), or that there are, or might be, external markup declarations in the document (No). The value in the default template is No, and if there is no standalone document declaration, the value is assumed to be No.

Example This is an XML declaration that specifies XML version 1.0, UTF-16LE encoding, and that the document can stand alone:

```
<?xml version="1.0" encoding="UTF-16LE"
standalone="yes"?>
```

Document type declaration

The document type declaration contains or points to markup declarations that provide a grammar for a class of documents. This grammar is known as a document type definition, or DTD. The document type declaration defines constraints on the sequence and nesting of tags, attribute values, names and formats of external references, and so forth. You can edit the document type declaration to change its name, but the name must always be the same as the name of the root element. Changing the name in either the document type declaration or the root element automatically changes the name in the other.

Adding DTDs You can add an identifier pointing to an external DTD subset, and you can add an internal DTD subset. If you supply both external and internal subsets, entity and attribute-list declarations in the internal subset take precedence over those in the external subset.

Public identifiers An external identifier can include a public identifier that an XML processor can use to generate an alternative URI. If an alternative URI cannot be generated, the URI provided in the system identifier is used. External identifiers without a public identifier are preceded by the keyword SYSTEM. External identifiers with a public identifier are preceded by the keyword PUBLIC.

Exporting metadata

If you specify a system or public identifier and/or an internal subset in the Document Type Declaration dialog box, a DTD cannot be generated when the data is exported to XML. A MetadataType of XMLDTD! is ignored. For more information about the properties that control the export of metadata, see “Exporting metadata” on page 404.

Examples These are examples of valid document type declarations.

An external system identifier:

```
<!DOCTYPE d_dept_listing SYSTEM "d_dept_listing.dtd">
```


An external system identifier with a public identifier:

```
<!DOCTYPE d_test PUBLIC "-//MyOrg//DTD Test//EN"
"http://www.mysite.org/mypath/mytest.dtd">
```

An external system identifier with an internal DTD. The internal DTD is enclosed in square brackets:

```
<!DOCTYPE d_orders
SYSTEM "http://www.acme.com/dtds/basic.dtd" [
<!ELEMENT Order (Date, CustID, OrderID, Items*)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT CustID (#PCDATA)>
<!ELEMENT OrderID (#PCDATA)>
<!ELEMENT Items (ItemID, Quantity)>
<!ELEMENT ItemID (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
]>
```

Root element

You can change the name of the root element, add attributes and children, and insert comments, instructions, and, if they do not already exist, XML and/or document type declarations before it.

Changing the name of the root element changes the name of its start and end tags. You can change the name using the Edit menu item, or in the Element Attributes dialog box. Changing the name of the document type declaration, if it exists, also changes the name of the root element, and vice versa. The root element name is always the same as the document type declaration name.

You can add the following kinds of children to the root element:

- Elements
- Text
- Control references
- DataWindow expressions (including column references)
- CDATA sections
- Comments
- Processing instructions

Controls

Adding a DataWindow control reference opens a dialog box containing a list of the columns, computed fields, report controls, and text controls in the document.

Control references can also be added to empty attribute values or element contents using drag-and-drop from the Control List view. Column references can also be added using drag-and-drop from the Column Specifications view.

Drag-and-drop cannot replace

You cannot drag-and-drop an item on top of another item to replace it. For example, if you want to replace one control reference with another control reference, or with a DataWindow expression, you first need to delete the control reference you want to replace.

DataWindow expressions

Adding a DataWindow expression opens the Modify Expression dialog box. This enables you to create references to columns from the data source of the DataWindow object. It also enables the calling of global functions. One use of this feature is to return a fragment of XML to embed, providing another level of dynamic XML generation.

Using Date and DateTime with strings

If you use a control reference or a DataWindow expression that does not include a string to represent Date and DateTime columns in a template, the XML output conforms to ISO 8601 date and time formats. For example, consider a date that displays as 12/27/2004 in the DataWindow object, using the display format mm/dd/yyyy. If the export template does not use an expression that includes a string, the date is exported to XML as 2004-12-27.

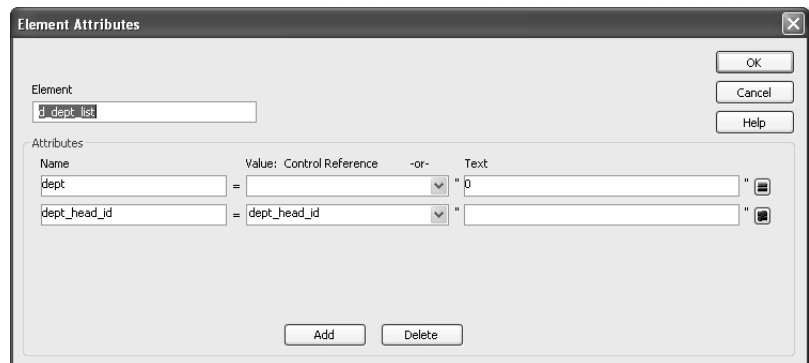
However, if the export template uses an expression that combines a column with a Date or DateTime datatype with a string, the entire expression is exported as a string and the regional settings in the Windows registry are used to format the date and time.

Using the previous example, if the short date format in the registry is MM/dd/yy, and the DataWindow expression is: "Start Date is " + start_date, the XML output is Start Date is 12/27/04.

Attributes

Controls or expressions can also be referenced for element attribute values. Select Edit/Add Attribute from the pop-up menu for elements to edit an existing attribute or add a new one.

For each attribute specified, you can select a control reference from the drop-down list or enter a literal text value. A literal text value takes precedence over a control reference. You can also use the expression button to the right of the Text box to enter an expression.



The expression button and entry operates similarly to DataWindow object properties in the Properties view. The button shows a green equals sign if an expression has been entered, and a red not-equals sign if not. A control reference or text value specified in addition to the expression is treated as a default value. In the template, this combination is stored with the control reference or text value, followed by a tab, preceding the expression. For example:

```
attribute_name=~"text_val~tdw_expression~"
```

Composite and nested reports

Report controls can be referenced in the Detail section of export templates as children of an element.

Nested reports supported for XML export only

Import does not support nested reports. If you attempt to import data in any format, including XML, CSV, DBF, and TXT, that contains a nested report, the nested report is not imported and the import may fail with errors.

Composite reports

For composite reports that use the Composite presentation style, the default template has elements that reference each of its nested reports.

If a composite DataWindow contains two reports that have columns with identical names, you must use the procedure that follows if you want to generate an XML document with a DTD or schema. If you do not follow the procedure, you will receive a parsing error such as “Element ‘*identical_column_name*’ has already been declared.”

- 1 Create a template in the first report and select this template in the Use Template list on the Data Export property page.
- 2 Create a template in the second report.
- 3 If any element name is used in the template in the first report, change it to another name in the template in the second report.
- 4 Select the template for the second report in the Use Template list.
- 5 Generate the XML document.

These steps are necessary because you cannot use a given element name more than once in a valid DTD or schema.

Nested reports

For report controls added to the detail band of a base report that is related to the inserted report with retrieval arguments or criteria, the report control is available to the export template in two ways:

- Select an element in the template or add a new element, then select Add Child>DataWindow Control Reference. Any report controls inserted in the detail band are available for selection in the dialog box that displays.
- Drag a report control from the Control List view and drop it on an existing empty element.

When you export XML using a template that has a reference to a report control, the export template assigned to the nested report with the Use Template property is used, if it exists, to expand the XML for the nested report. If no template is specified for the nested report, the default template is used.

The relationship between the nested report and the base report, for example a Master/Detail relationship, is reflected in the exported XML.

CDATA sections

You can export the name of a column in a CDATA section using the syntax `<![CDATA[columnname]]>`. You can export the value of a column using the syntax `<![CDATA[~t columnname]]>`. The `~t` is used to introduce a DataWindow expression, in the same way that it is used in the Modify method. You can also use an expression such as `~t columnname*columnname` to export a computed value to the XML.

You can import a value into a column using the syntax `<![CDATA[columnname]]>`. Note that this syntax in a template has different results for import and export: it imports the column value but exports the column name.

You *cannot* import an XML file that has a `~t` expression in a CDATA section.

Everything else inside a CDATA section is ignored by the parser. If text contains characters such as less than or greater than signs (`<` or `>`) or ampersands (`&`) that are significant to the parser, it should be defined as a CDATA section. A CDATA section starts with `<![CDATA[` and ends with `]]>`. CDATA sections cannot be nested, and there can be no white space characters inside the `]]>` delimiter—for example, you cannot put a space between the two square brackets.

Example

```
<![CDATA[
    do not parse me
]]>
```

This syntax in an export template exports the value of the column `emp_salary`:

```
<![CDATA[~t emp_salary]]>
```

This syntax in an import template imports the value of the column `emp_salary`:

```
<![CDATA[emp_salary]]>
```

Comments

Comments can appear anywhere in a document outside other markup. They can also appear within the document type declaration in specific locations defined by the XML specification.

Comments begin with `<!--` and end with `-->`. You cannot use the string `--` (a double hyphen) in a comment, and parameter entity references are not recognized in comments.

Example

```
<!-- this is a comment -->
```

Processing instructions

Processing instructions (PIs) enable you to provide information to the application that uses the processed XML. Processing instructions are enclosed in `<? and ?>` delimiters and must have a name, called the target, followed by optional data that is processed by the application that uses the XML. Each application that uses the XML must process the targets that it recognizes and ignore any other targets.

The XML declaration at the beginning of an XML document is an example of a processing instruction. You cannot use the string `xml` as the name of any other processing instruction target.

Example

In this example, `usething` is the name of the target, and `thing=this.thing` is the data to be processed by the receiving application:

```
<?usething thing=this.thing?>
```

Exporting to XML

You can export the data in a `DataWindow` or `DataStore` object to XML using any of the techniques used for exporting to other formats such as PSR or HTML:

- Using the `SaveAs` method:

```
ds_1.SaveAs("C:\TEMP\Temp.xml", Xml, true)
```

- Using the `Describe` method:

```
ls_xmlstring = dw_1.Describe(DataWindow.Data.XML)
```

- Using the `Save Rows As` menu item in the `DataWindow` painter.

With the `Preview` view open, select `File>Save Rows As`, select `XML` from the `Files of Type` drop-down list, provide a file name, and click `Save`. You can use this in the development environment to preview the XML that will be generated at runtime.

When you export data, `DataWindow Designer` uses an export template to specify the content of the generated XML.

Default export format

If you have not created or assigned an export template, DataWindow Designer uses a default export format. This is the same format used when you create a new default export template. See “Creating templates” on page 385.

OLE DataWindow objects cannot be exported using a template. You must use the default format.

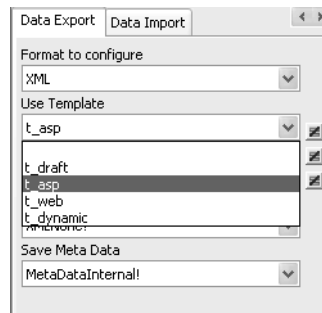
Setting properties on the Data Export page

The Data Export page in the Properties view lets you set properties for exporting data to XML.

In addition to the properties that you can set on this page, DataWindow Designer provides two properties that you can use to let the user of an application select an export template at runtime. See “Selecting templates at runtime” on page 408.

The Use Template property

The names of all templates that you create and save for the current DataWindow object display in the Use Template drop-down list.



The template you select from the list is used to conform the XML to the specifications defined in the named template. Selecting a template from the list box sets the DataWindow object’s `Export.XML.UseTemplate` property. You can also modify the value of the `UseTemplate` property dynamically in a script. For example, an XML publishing engine would change templates dynamically to create different presentations of the same data.

Template used when saving

When you open a DataWindow, the Export/Import Template view displays the template specified in the DataWindow's Use Template property. (If the view is not visible in the current layout, select View>Export/Import Template>XML from the menu bar.) If the property has not been set, the first saved template displays or, if there are no saved templates, the default structured template displays as a basis for editing.

When the DataWindow is saved as XML, DataWindow Designer uses the template specified in the Use Template property. If the property has not been set, DataWindow Designer uses the default template.

When you are working on a template, you might want to see the result of your changes. The template specified in the Use Template property might not be the template currently displayed in the Export/Import Template view, so you should check the value of the Use template property to be sure you get the results you expect.

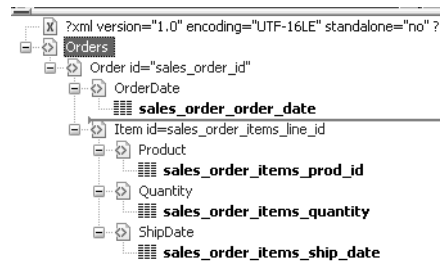
❖ **To save to XML using the current template:**

- 1 Right-click in the Export/Import template view and select Save or Save As from the pop-up menu to save the current template.
- 2 On the Data Export page in the properties view, select the current template from the Use Template drop-down list.
- 3 Select File>Save Rows As, select XML from the Files of Type drop-down list, enter a file name, and click Save.

Generating group headers

To generate the contents of the header section iteratively for each group in a group DataWindow, check the Iterate Header for Groups check box, or set the Export.XML.HeadGroups DataWindow property. This property is on by default.

For example, consider a group DataWindow object that includes the columns `sales_order_id` and `sales_order_order_date`. The following screenshot shows the template for this DataWindow object:



The root element in the Header section of the template, `Orders`, has a child element, `Order`. `Order` has an `id` attribute whose value is a control reference to the column `sales_order_id`. `Order` also has a child element, `OrderDate`, that contains a column reference to the `sales_order_order_date` column. These elements make up the header section that will be iterated for each group.

The Detail Start element, `Item`, has an `id` attribute whose value is a control reference to the column `sales_order_items_line_id`. It also has three child elements that contain column references to the line items for product ID, quantity, and ship date.

When the DataWindow is exported with the Iterate Header for Groups property on, the order ID and date iterate for each group. The following XML output shows the first three iterations of the group header:

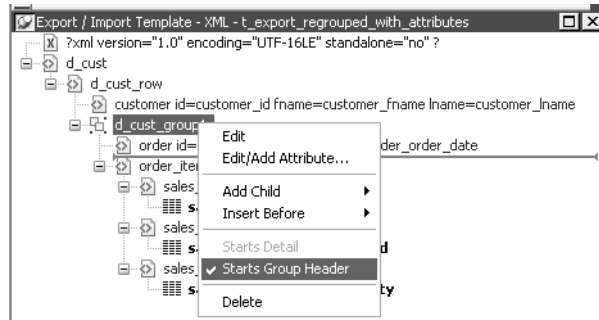
```
<?xml version="1.0" encoding="UTF-16LE"
standalone="no"?>
<Orders>
  <Order id="2001">
    <OrderDate>2002-03-14</OrderDate>
    <Item id="1">
      <Product>300</Product>
      <Quantity>12</Quantity>
      <ShipDate>2005-09-15</ShipDate>
    </Item>
    <Item id="2">
      <Product>301</Product>
      <Quantity>12</Quantity>
      <ShipDate>2005-09-14</ShipDate>
    </Item>
    <Item id="3">
      <Product>302</Product>
      <Quantity>12</Quantity>
```

```

        <ShipDate>2005-09-14</ShipDate>
    </Item>
</Order>
<Order id="2002">
    <OrderDate>2002-03-18</OrderDate>
    <Item id="2">
        <Product>401</Product>
        <Qty>24</Qty>
        <ShipDate>2002-09-18</ShipDate>
    </Item>
    <Item id="1">
        <Product>400</Product>
        <Qty>24</Qty>
        <ShipDate>2002-09-18</ShipDate>
    </Item>
</Order>
<Order id="2003">
    <OrderDate>2002-03-21</OrderDate>
    <Item id="3">
        <Product>400</Product>
        <Qty>12</Qty>
        <ShipDate>2002-09-23</ShipDate>
    </Item>
    ...

```

For DataWindow objects with more than one group, when you generate a new default template, each group after the first is identified with a special icon and a check on the pop-up menu next to the Starts Group Header item.



When the Iterate Header for Groups check box is selected, each XML fragment in the header section between a Group Header element and the next Group Header element or Detail Start element is iterated.

In the template shown in the previous illustration, sales are grouped by customer ID, then by order ID. The customer group header has attributes for the customer's ID and first and last names. The order group header has attributes for the order ID and date. The following illustration shows the DataWindow in the Design view:

Customer ID	First Name	Last Name	Sales Order ID	Order Date	Line #	Product ID	Quantity
Header ↑							
customer_id	customer_fnamecustomer_lname						
1: Header group customer_id ↑							
			sales_order_id	sales_order_order_date			
2: Header group sales_order_id ↑							
			sales_order_items/sales_order_items_prod/sales_order_items_quantit				
Detail ↑							
2: Trailer group sales_order_id ↑							
1: Trailer group customer_id ↑							
Summary ↑							
Footer ↑							

The following XML output shows the first iteration of the customer group header and the first and second iterations of the order group header:

```
<?xml version="1.0" encoding="UTF-16LE" standalone="no"?>
<d_customer>
  <customer id="101" fname="Michaels" lname="Devlin">
    <order id="2001" date="1996-03-14">
      <order_item>
        <sales_order_items_line_id>1</sales_order_items_line_id>
        <sales_order_items_prod_id>300</sales_order_items_prod_id>
        <sales_order_items_quantity>12</sales_order_items_quantity>
      </order_item>
      <order_item>
        <sales_order_items_line_id>2</sales_order_items_line_id>
        <sales_order_items_prod_id>301</sales_order_items_prod_id>
        <sales_order_items_quantity>12</sales_order_items_quantity>
      </order_item>
      <order_item>
        <sales_order_items_line_id>3</sales_order_items_line_id>
        <sales_order_items_prod_id>302</sales_order_items_prod_id>
        <sales_order_items_quantity>12</sales_order_items_quantity>
      </order_item>
    </order>
  <order id="2005" date="1996-03-24">
    <order_item>
      <sales_order_items_line_id>1</sales_order_items_line_id>
      <sales_order_items_prod_id>700</sales_order_items_prod_id>
      <sales_order_items_quantity>12</sales_order_items_quantity>
    </order_item>
  </order>
</d_customer>
```

Formatting the exported XML

By default, the XML is exported without formatting. If you want to view or verify the exported XML in a text editor, check the Include Whitespace check box or set the `Export.XML.IncludeWhitespace` property in code. Turning this property on causes the export process to insert tabs, carriage returns, and linefeed characters into the XML so that it is easier to read. Most of the examples in this chapter were exported with this property turned on.

Do not import formatted XML

You should not try to import XML formatted with white space characters, because the white space between data element tags is considered to be part of the element.

Exporting metadata

You can specify that metadata in the form of a DTD or schema should be exported when you save the `DataWindow` object. You can choose to save the metadata with the XML or in a separate file.

If you export metadata as a schema, you can associate it with a namespace. See “Associating a namespace with an exported schema” on page 406.

To specify how metadata should be saved, select a value from the Meta Data Type drop-down list or set the `Export.XML.MetaDataType` property. The possible values are:

- `XMLNone!`—No metadata is generated
- `XMLSchema!`—An XML schema is generated
- `XMLDTD!`—A DTD is generated

The metadata is saved into the exported XML itself or into an associated file, depending on the setting in the Save Meta Data drop-down list or the `Export.XML.SaveMetaData` property. The possible values are:

- `MetaDataInternal!`—The metadata is saved into the generated XML document or string. To save metadata using the `.Data.XML` expression syntax, you must use this value.
- `MetaDataExternal!`—The metadata is saved as an external file with the same name as the XML document but with the extension `.xsd` (for a schema) or `.dtd` (for a DTD). A reference to the name of the metadata file is included in the output XML document.

Example: internal metadata

For example, if you select XMLDTD! and MetaDataInternal!, the header and first row of the exported XML would look like this for a simple grid DataWindow for the contact table in the EAS Demo DB. The Include Whitespace property has also been selected and the file name is *dtdinternal.xml*:

```
<?xml version="1.0" encoding="UTF-16LE"
standalone="yes"?>
<!DOCTYPE dtdinternal [

```

Example: external metadata

If you select MetaDataExternal! instead, the generated XML in *dtdexternal.xml* looks like this:

```
<?xml version="1.0" encoding="UTF-16LE"?>
<!DOCTYPE dtdexternal SYSTEM "dtdexternal.dtd">
<dtdexternal>
  <dtdexternal_row>
    <id>1</id>
    <last_name>Hildebrand</last_name>
    <first_name>Jane</first_name>
```

```
<title>ma</title>
<street>1280 Washington St.</street>
<city>Emeryville</city>
<state>MI</state>
<zip>94608</zip>
<phone>5105551309</phone>
<fax>5105554209</fax>
</dtdexternal_row>
```

The DTD is in *dtdexternal.dtd*:

```
<?xml version="1.0" encoding="UTF-16LE"?><!ELEMENT
dtdexternal (dtdexternal_row*)>
<!ELEMENT dtdexternal_row (id, last_name, first_name,
title, street, city, state, zip, phone, fax)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT last_name (#PCDATA)>
<!ELEMENT first_name (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT street (#PCDATA)>
<!ELEMENT city (#PCDATA)>
<!ELEMENT state (#PCDATA)>
<!ELEMENT zip (#PCDATA)>
<!ELEMENT phone (#PCDATA)>
<!ELEMENT fax (#PCDATA)>
```

Associating a namespace with an exported schema

If you export metadata in the form of a schema, you can associate a namespace with the schema. To do so, right-click the root element in the Export/Import template view and select Schema Options from the pop-up menu. In the dialog box, specify the namespace prefix and URI.

When the Meta Data Type property is XMLSchema! and the Save Meta Data property is MetaDataInternal!, so that the XML schema is generated inline, you can specify a name for the root element. If the root element name is specified, it appears in the generated XML.

In the following example, the root element name is Contacts, the namespace prefix is po, and the URI is *http://www.example.com/PO1*.

The example shows the header and the first row of the generated XML:

```
<?xml version="1.0" encoding="UTF-16LE"
standalone="no"?>
<Contacts>
  <xs:schema xmlns:po="http://www.example.com/PO1"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.example.com/PO1"
    elementFormDefault="qualified"
    attributeFormDefault="unqualified">
    <xs:element name="d_contact_list">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="d_contact_list_row"
            maxOccurs="unbounded" minOccurs="0"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="d_contact_list_row">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="id"/>
          <xs:element ref="last_name"/>
          <xs:element ref="first_name"/>
          <xs:element ref="city"/>
          <xs:element ref="state"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="id" type="xs:int"/>
    <xs:element name="last_name" type="xs:string"/>
    <xs:element name="first_name" type="xs:string"/>
    <xs:element name="city" type="xs:string"/>
    <xs:element name="state" type="xs:string"/>
  </xs:schema>
  <po:d_contact_list xmlns:po=
    "http://www.example.com/PO1" xmlns:xsi=
    "http://www.w3.org/2001/XMLSchema-instance">
    <po:d_contact_list_row>
      <po:id>1</po:id>
      <po:last_name>Hildebrand</po:last_name>
      <po:first_name>Jane</po:first_name>
      <po:city>Emeryville</po:city>
      <po:state>MI</po:state>
    </po:d_contact_list_row>
```

By default, the generated XML is not associated with a namespace.

Selecting templates at runtime

Two `DataWindow` properties, `Export.XML.TemplateCount` and `Export.XML.Template[].Name`, enable you to provide a list of templates from which the user of the application can select at runtime.

The `TemplateCount` property gets the number of templates associated with a `DataWindow` object. You can use this number as the upper limit in a FOR loop that populates a drop-down list with the template names. The FOR loop uses the `Template[].Name` property.

Before generating the XML, set the export template using the text in the drop-down list box.

Importing XML

You can select XML as a file type in the dialog box that displays when you select `Rows>Import` in the `DataWindow` painter. (The `Preview` view must be open to enable the `Rows>Import` menu item.)

You can also import data from an XML document or string using the `ImportFile`, `ImportString`, or `ImportClipboard` methods. These methods have a parameter that enables you to specify the type of data to be imported.

Data can be imported with or without a template. To import data without a template, the data must correspond to the `DataWindow` column definition. The text content of the XML elements must match the column order, column type, and validation requirements of the `DataWindow` columns.

Composite and Graph `DataWindow` objects

Composite and Graph `DataWindow` objects cannot be imported using a template. You must use the default format.

Importing with a template

If the XML document or string from which you want to import data does not correspond to the DataWindow column definition, or if you want to import attribute values, you must use a template.

If a schema is associated with the XML to be imported, you must create a template that reflects the schema.

For complex, nested XML with row data in an iterative structure, you may need to design a structure that uses several linked DataWindow definitions to import the data. Each DataWindow must define the structure of a block of iterative data with respect to the root element. Importing the data into the DataWindow objects would require multiple import passes using different import templates.

Defining import templates

The XML import template can be defined in the Export/Import Template view for XML. If you are defining a template for use only as an import template, do not include DataWindow expressions, text, comments, and processing instructions. These items are ignored when data is imported.

Only mappings from DataWindow columns to XML elements and attributes that follow the Starts Detail marker in the template are used for import. Element and attribute contents in the header section are also ignored. If the Starts Detail marker does not exist, all element and attribute to column mappings within the template are used for import. For more information about the Starts Detail marker, see “The Detail Start element” on page 388.

Matching template structure to XML

An XML import template must map the XML element and attribute names in the XML document to DataWindow column names, and it must reflect the nesting of elements and attributes in the XML.

The order of elements and attributes with column reference content in the template does not have to match the order of columns within the DataWindow, because import values are located by name match and nesting depth within the XML. However, the order of elements and attributes in the template must match the order in which elements and attributes occur in the XML. Each element or attribute that has column reference content in the template must occur in each row in the XML document or string. The required elements and attributes in the XML can be empty.

If an element or attribute does not occur in the XML document, the DataWindow import column remains empty.

The data for the DataWindow is held in the columns of the data table. Some data columns, such as those used for computed fields, may not have an associated control. To import data into a column that has no control reference, add a child DataWindow expression that contains the column name.

Remove tab characters

When you select a column name in the DataWindow expression dialog box, tab characters are added before and after the name. You should remove these characters before saving the expression.

Importing data with group headers

For XML import using a template, element and attribute contents in the header section are ignored. However, if the Starts Detail marker does not exist, all element and attribute to column mappings within the template are used for import. This has the following implications for DataWindow objects with group headers:

- If data is imported to a Group DataWindow using a template that has a Starts Detail marker, the group header data is not imported because import starts importing from the Starts Detail location.
 - If the Group DataWindow has one group and the import template has no Starts Detail marker, all the data is imported successfully.
-

Nested groups cannot be imported

If the Group DataWindow has nested groups, the data cannot be imported successfully even if the Starts Detail marker in the import template is turned off.

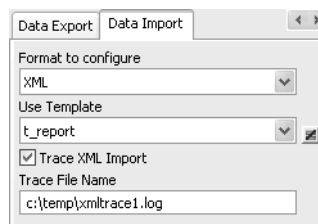
Restrictions

DataWindow columns cannot be referenced twice for import. A second column reference to a DataWindow column within an XML import template is ignored.

An XML element or attribute name whose content references a DataWindow column for import must be unique within the level of nesting. It cannot occur twice in the template at the same nesting level.

Setting the import template

The names of all templates for the current DataWindow object display in the Use Template drop-down list on the Data Import page in the Properties view.



Using export templates for import

If you have already defined an export template for a DataWindow object, you can use it as an import template, but only the mapping of column names to element attribute names is used for import. All other information in the template is ignored.

The template you select in the list box is used to conform the XML imported to the specifications defined in the named template. Selecting a template from the list sets the DataWindow object's Import.XML.UseTemplate property. You can also modify the value of the Import.XML.UseTemplate property dynamically in code.

The Data Import page also contains a check box that enables you to create a trace log of the import. See “Tracing import” on page 416.

Example

This example uses a DataWindow object that includes the columns emp_id, emp_fname, emp_lname, and dept_id. The template used in this example includes only these columns. Any other columns in the DataWindow remain empty when you import using this template.

To illustrate how template import works, create a new template that has one element in the header section, called before_detail_marker. This element contains a column reference to the emp_id column.

The Detail Start element, employee, has an attribute, dept_id, whose value is a control reference to the column dept_id. It also has three children:

- The emp_id element contains a column reference to the emp_id column.
- The emp_fname element contains static text.

- The name element has two children, emp_fname and emp_lname, that contain column references to those columns.



The template exports and imports the dept_id DataWindow column using the attribute of the employee element. It exports and imports the emp_id, emp_fname, and emp_lname columns using the column references in the elements. The following shows the beginning of the XML exported using this template:

```
<?xml version="1.0" encoding="UTF-16LE"
standalone="no"?>

<employee_list>
  <before_detail_marker>102</before_detail_marker>
  <employee dept_id="100">
    <emp_id>102</emp_id>
    <emp_fname>static text content</emp_fname>
    <name>
      <emp_fname>Fran</emp_fname>
      <emp_lname>Whitney</emp_lname>
    </name>
  </employee>
  <employee dept_id="100">
    <emp_id>105</emp_id>
    <emp_fname>static text content</emp_fname>
    <name>
      <emp_fname>Matthew</emp_fname>
      <emp_lname>Cobb</emp_lname>
    </name>
  </employee>
  ...

```

The exported XML can be reimported into the DataWindow columns `dept_id`, `emp_id`, `emp_fname`, and `emp_lname`. Before importing, you must set the import template on the Data Import page in the Properties view or in code using the DataWindow object's `Import.XML.UseTemplate` property.

The following items are exported, but ignored on import:

- The `before_detail_marker` element is ignored because it is in the header section.
- The first occurrence of the element tag name `emp_fname` is ignored because it does not contain a mapping to a DataWindow column name.

If you change the nesting of the `emp_fname` and `emp_lname` elements inside the name element, the import fails because the order of the elements and the nesting in the XML and the template must match.

Default data import

When there is no import template assigned to a DataWindow object with the Use Template property, DataWindow Designer attempts to import the data using the default mechanism described in this section.

Elements that contain text

The text between the start and end tags for each element can be imported if the XML document data corresponds to the DataWindow column definition. For example, this is the case if the XML was exported from DataWindow Designer using the default XML export template.

The text content of the XML elements must match the column order, column type, and validation requirements of the DataWindow columns. (The same restriction applies when you import data from a text file with the `ImportFile` method).

All element text contents are imported in order of occurrence. Any possible nesting is disregarded. The import process ignores tag names of the elements, attributes, and any other content of the XML document.

Empty elements

Empty elements (elements that have no content between the start and end tags) are imported as empty values into the DataWindow column. If the element text contains only white space, carriage returns, and new line or tab characters, the element is treated as an empty element.

Any attributes of empty elements are ignored.

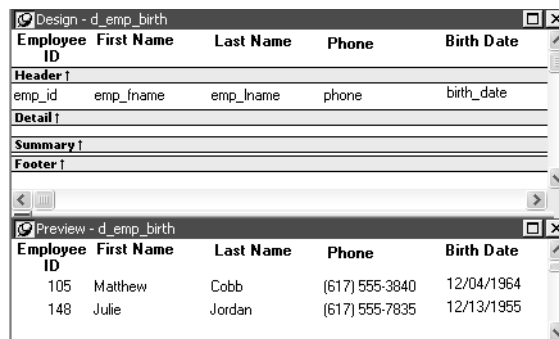
Elements with non-text content

If the element has no text content, but does contain comments, processing instructions, or any other content, it is not regarded as an empty element and is skipped for import.

Example with no empty elements

The three XML documents that follow all show the same result when you select Rows>Import in the DataWindow painter if ImportFile is called with or without default arguments for start and end column, start and end row, and DataWindow start column.

The DataWindow object has five columns: emp_id, emp_fname, emp_lname, phone, and birth_date.



Example 1

This example contains two rows, each with five elements that match the column order, type, and validation requirements for the DataWindow object.

```
<?xml version="1.0"?>
<d_emp_birth_listing>
  <d_emp_birth_row>
    <element_1>105</element_1>
    <element_2>Matthew</element_2>
    <element_3>Cobb</element_3>
    <element_4>6175553840</element_4>
    <element_5>04/12/1960</element_5>
  </d_emp_birth_row>
  <d_emp_birth_row>
    <element_1>148</element_1>
    <element_2>Julie</element_2>
    <element_3>Jordan</element_3>
    <element_4>6175557835</element_4>
    <element_5>11/12/1951</element_5>
  </d_emp_birth_row>
</d_emp_birth_listing>
```

Example 2

In this example, the elements are not contained in rows, but they still match the DataWindow object.

```
<?xml version="1.0"?>
<root_element>
  <element_1>105</element_1>
  <element_2>Matthew</element_2>
  <element_3>Cobb</element_3>
  <element_4>6175553840</element_4>
  <element_5>04/12/1960</element_5>
  <element_6>148</element_6>
  <element_7>Julie</element_7>
  <element_8>Jordan</element_8>
  <element_9>6175557835</element_9>
  <element_10>11/12/1951</element_10>
</root_element>
```

Example 3

The comments and processing instructions in this example are not imported. The nesting of the <first> and <last> elements within the <Name> element is ignored.

```
<?xml version="1.0"?>
<root_element>
  <!-- some comment -->
  <row_element><?process me="no"?>105<name Title="Mr">
    <first>Matthew</first>
    <last>Cobb</last>
  </name>
  <!-- another comment -->
  <phone>6175553840</phone>
  <birthdate>04/12/1960</birthdate>
</row_element>
  <row_element>148<name Title="Ms">
    <first>Julie</first>
    <last>Jordan</last>
  </name>
  <phone>6175557835</phone>
  <birthdate>11/12/1951</birthdate>
</row_element>
</root_element>
```

Result

All three XML documents produce this result:

emp_id	emp_fname	emp_lname	phone	birth_date
105	Matthew	Cobb	6175553840	04/12/1960
148	Julie	Jordan	6175557835	11/12/1951

Example with empty elements

Example 4

This example uses the same DataWindow object, but there are two empty elements in the XML document. The first has no content, and the second has an attribute but no content. Both are imported as empty elements.

```
<?xml version="1.0"?>
<root_element>
<!-- some comment -->
<row_element>
<?process me="no"?>105<name Title="Mr">
<first>Matthew</first>
<!-- another comment -->
<last>Cobb</last>
</name>
<empty></empty>
<birthdate>04/12/1960</birthdate>
</row_element>
<row_element>148<name Title="Ms">
<empty attribute1 = "blue"></empty>
<last>Jordan</last>
</name>
<phone>6175557835</phone>
<birthdate>11/12/1951</birthdate>
</row_element>
</root_element>
```

Result

The XML document produces this result:

emp_id	emp_fname	emp_lname	phone	birth_date
105	Matthew	Cobb		04/12/1960
148		Jordan	6175557835	11/12/1951

Tracing import

When you import data from XML with or without a template, you can create a trace log to verify that the import process worked correctly. The trace log shows whether a template was used and if so which template, and it shows which elements and rows were imported.

To create a trace log, select the Trace XML Import check box on the Data Import page in the Properties view and specify the name and location of the log file in the Trace File Name box. If you do not specify a name for the trace file, DataWindow Designer generates a trace file with the name *pbxmtrc.log* in the current directory.

You can also use the `Import.XML.Trace` and `Import.XML.TraceFile` `DataWindow` object properties.

If you use `ImportClipboard` or `ImportString` to import the data, you must specify XML! as the *importtype* argument. For example:

```
importstring(XML!, ls_xmlstring)
```

If you omit the *importtype* argument, the trace file is not created. You do not need to specify the *importtype* argument if you use `ImportFile`.

Example: default import

The following trace log shows a default import of the department table in the EAS Demo database:

```
/*-----*/
/*                09/10/2005 18:26                */
/*-----*/
CREATING SAX PARSER.
NO XML IMPORT TEMPLATE SET - STARTING XML DEFAULT
IMPORT.
DATAWINDOW ROWSIZE USED FOR IMPORT: 3

ELEMENT: dept_id: 100
ELEMENT: dept_name: R & D
ELEMENT: dept_head_id: 501
--- ROW
ELEMENT: dept_id: 200
ELEMENT: dept_name: Sales
ELEMENT: dept_head_id: 902
--- ROW
ELEMENT: dept_id: 300
ELEMENT: dept_name: Finance
ELEMENT: dept_head_id: 1293
--- ROW
ELEMENT: dept_id: 400
ELEMENT: dept_name: Marketing
ELEMENT: dept_head_id: 1576
--- ROW
ELEMENT: dept_id: 500
ELEMENT: dept_name: Shipping
ELEMENT: dept_head_id: 703
--- ROW
```

Example: template import

The following trace log shows a template import of the department table. The template used is named t_1. Notice that the DataWindow column dept_id is referenced twice, as both an attribute and a column. The second occurrence is ignored for the template import, as described in “Restrictions” on page 410. The Detail Start element has an implicit attribute named __pbband which is also ignored.

```

/*-----*/
/*          09/10/2005 18:25          */
/*-----*/
CREATING SAX PARSER.
USING XML IMPORT TEMPLATE: t_1

XML NAMES MAPPING TO DATAWINDOW IMPORT COLUMNS:
ATTRIBUTE: /d_dept/d_dept_row NAME: '__pbband'
>>> RESERVED TEMPLATE NAME - ITEM WILL BE IGNORED
ATTRIBUTE: /d_dept/d_dept_row/dept_id_xml_name NAME: 'dept_id'
DATAWINDOW COLUMN: 1, NAME: 'dept_id'
ELEMENT: /d_dept/d_dept_row/dept_id_xml_name
>>> DUPLICATE DATAWINDOW COLUMN REFERENCE: 1, NAME: 'dept_id' - ITEM WILL
BE IGNORED
ELEMENT: /d_dept/d_dept_row/dept_head_id
DATAWINDOW COLUMN: 3, NAME: 'dept_head_id'
ELEMENT: /d_dept/d_dept_row/dept_name
DATAWINDOW COLUMN: 2, NAME: 'dept_name'

ATTRIBUTE: /d_dept/d_dept_row/dept_id_xml_name NAME: 'dept_id': 100
ELEMENT: /d_dept/d_dept_row/dept_head_id: 501
ELEMENT: /d_dept/d_dept_row/dept_name: R & D
--- ROW
ATTRIBUTE: /d_dept/d_dept_row/dept_id_xml_name NAME: 'dept_id': 200
ELEMENT: /d_dept/d_dept_row/dept_head_id: 902
ELEMENT: /d_dept/d_dept_row/dept_name: Sales
--- ROW
ATTRIBUTE: /d_dept/d_dept_row/dept_id_xml_name NAME: 'dept_id': 300
ELEMENT: /d_dept/d_dept_row/dept_head_id: 1293
ELEMENT: /d_dept/d_dept_row/dept_name: Finance
--- ROW
ATTRIBUTE: /d_dept/d_dept_row/dept_id_xml_name NAME: 'dept_id': 400
ELEMENT: /d_dept/d_dept_row/dept_head_id: 1576
ELEMENT: /d_dept/d_dept_row/dept_name: Marketing
--- ROW
ATTRIBUTE: /d_dept/d_dept_row/dept_id_xml_name NAME: 'dept_id': 500
ELEMENT: /d_dept/d_dept_row/dept_head_id: 703
ELEMENT: /d_dept/d_dept_row/dept_name: Shipping
--- ROW

```

About this chapter

This chapter describes how to build and use graphs in DataWindow Designer.

Contents

Topic	Page
About graphs	419
Using graphs in DataWindow objects	425
Using the Graph presentation style	439
Defining a graph's properties	440

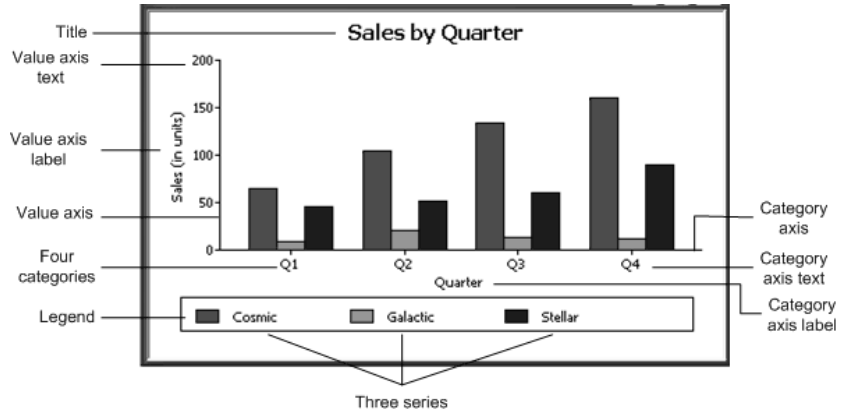
About graphs

Often the best way to display information is graphically. Instead of showing users a series of rows and columns of data, you can present information as a graph in a DataWindow object. For example, in a sales application, you might want to present summary information in a column graph.

DataWindow Designer provides many types of graphs and allows you to customize your graphs in many ways. Probably most of your use of graphs will be in a DataWindow object—the source of the data for your graphs will be the database.

Parts of a graph

Here is a column graph created in DataWindow Designer that contains most major parts of a graph. It shows quarterly sales of three products: Stellar, Cosmic, and Galactic printers:



How data is represented

Graphs display data points. To define graphs, you need to know how the data is represented. DataWindow Designer organizes data into three components.

Table 15-1: Components of a graph

Component	Meaning
Series	A set of data points Each set of related data points makes up one series. In the preceding graph, there is a series for Stellar sales, another series for Cosmic sales, and another series for Galactic sales. Each series in a graph is distinguished by color, pattern, or symbol.
Categories	The major divisions of the data Series data are divided into categories, which are often non-numeric. In the preceding graph, the series are divided into four categories: Q1, Q2, Q3, and Q4. Categories represent values of the independent variable(s).
Values	The values for the data points (dependent variables).

Organization of a graph

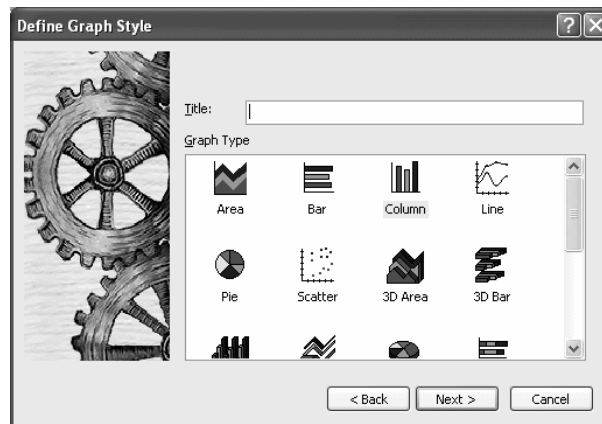
Table 15-2 lists the parts of a typical graph.

Table 15-2: Organization of a graph

Part of graph	What it is
Title	An optional title for the graph. The title appears at the top of the graph.
Value axis	The axis of the graph along which the values of the dependent variable(s) are plotted. In a column graph, as shown in the preceding graph, the Value axis corresponds to the y axis in an XY presentation. In other types of graphs, such as a bar graph, the Value axis can be along the x dimension.
Category axis	The axis along which are plotted the major divisions of the data, representing the independent variable(s). In the preceding graph, the Category axis corresponds to the x axis. It plots four categories: Q1, Q2, Q3, and Q4. These form the major divisions of data in the graph.
Series	A set of data points. There are three series in the preceding graph: Stellar, Cosmic, and Galactic. In bar and column charts, each series is represented by bars or columns of one color or pattern.
Series axis	The axis along which the series are plotted in three-dimensional (3D) graphs.
Legend	An optional listing of the series. The preceding graph contains a legend that shows how each series is represented in the graph.

Types of graphs

DataWindow Designer provides many types of graphs for you to choose from. You choose the type on the Define Graph Style page in the DataWindow wizard or in the General page in the Properties view for the graph.



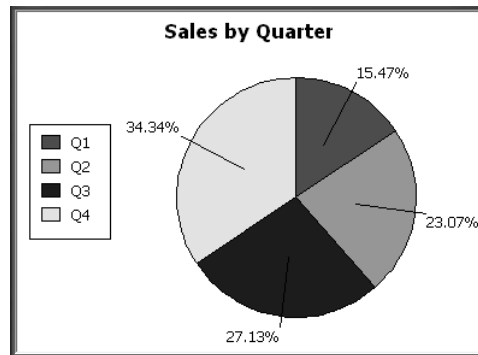
Area, bar, column, and line graphs

Area, bar, column, and line graphs are conceptually very similar. They differ only in how they physically represent the data values—whether they use areas, bars, columns, or lines to represent the values. All other properties are the same. Typically you use area and line graphs to display continuous data and use bar and column graphs to display noncontinuous data.

The only difference between a bar graph and a column graph is the orientation: in column graphs, values are plotted along the y axis and categories are plotted along the x axis. In bar graphs, values are plotted along the x axis and categories are plotted along the y axis.

Pie graphs

Pie graphs typically show one series of data points with each data point shown as a percentage of a whole. The following pie graph shows the sales for Stellar printers for each quarter. You can easily see the relative values in each quarter. (DataWindow Designer automatically calculates the percentages of each slice of the pie.)



You can have pie graphs with more than one series if you want; the series are shown in concentric circles. Multiseries pie graphs can be useful in comparing series of data.

Scatter graphs

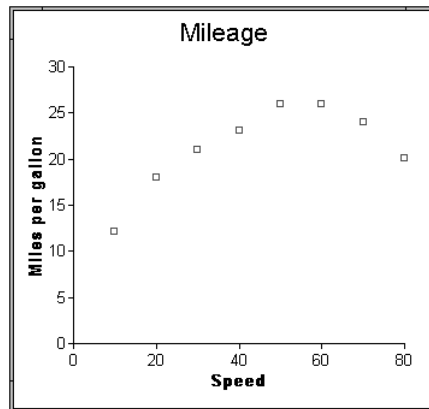
Scatter graphs show xy data points. Typically you use scatter graphs to show the relationship between two sets of numeric values. Non-numeric values, such as string and DateTime datatypes, do not display correctly.

Scatter graphs do not use categories. Instead, numeric values are plotted along both axes—as opposed to other graphs, which have values along one axis and categories along the other axis.

For example, the following data shows the effect of speed on the mileage of a sedan:

Speed	Mileage
10	12
20	18
30	21
40	23
50	26
60	26
70	24
80	20

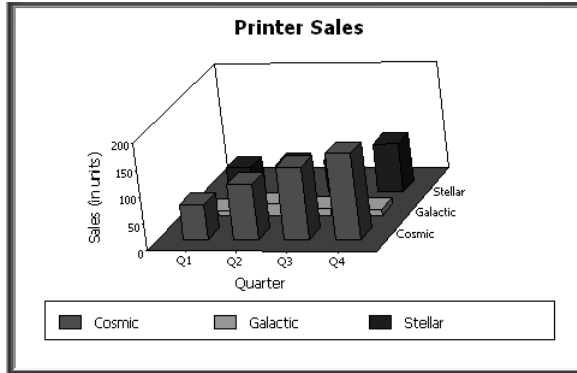
Here is the data in a scatter graph:



You can have multiple series of data in a scatter graph. You might want to plot mileage versus speed for several makes of cars in the same graph.

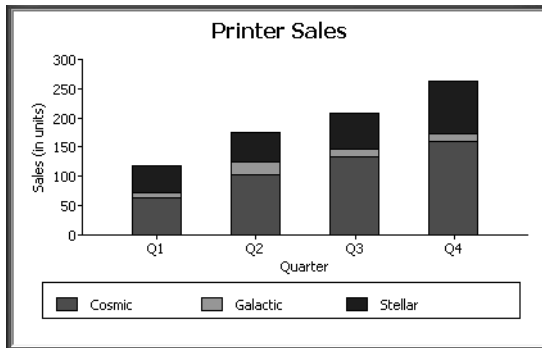
Three-dimensional graphs

You can also create 3-dimensional (3D) graphs of area, bar, column, line, and pie graphs. In 3D graphs (except for 3D pie graphs), series are plotted along a third axis (the Series axis) instead of along the Category axis. You can specify the perspective to use to show the third dimension:



Stacked graphs

In bar and column graphs, you can choose to stack the bars and columns. In stacked graphs, each category is represented as one bar or column instead of as separate bars or columns for each series:



Using graphs in DataWindow objects

Graphs in DataWindow objects are dynamic

Graphs in DataWindow objects are tied directly to the data that is in the DataWindow object. As the data changes, the graph is automatically updated to reflect the new values.

Two techniques

You can use graphs in DataWindow objects in two ways:

- By including a graph as a control in a DataWindow object

The graph enhances the display of information in a DataWindow object, such as a tabular or freeform DataWindow object. This technique is described in "Placing a graph in a DataWindow object" next.

- By using the Graph presentation style

The entire DataWindow object is a graph. The underlying data is not visible. This technique is described in "Using the Graph presentation style" on page 439.

Placing a graph in a DataWindow object

❖ To place a graph in a DataWindow object:

- 1 Open or create the DataWindow object that will contain the graph.
- 2 Select Insert>Control>Graph from the menu bar.
- 3 Click where you want the graph.

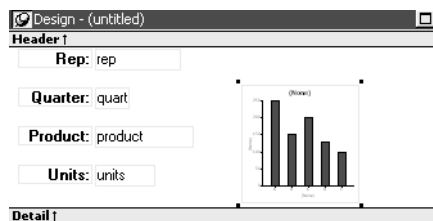
DataWindow Designer displays the Graph Data dialog box:



- 4 Specify which columns contain the data and the type of graph you want, and click OK.

For more information, see “Associating data with a graph” on page 428.

The Design view now contains a representation of the graph:



- 5 Specify the graph's properties in the Properties view.

Using the graph's Properties view

A graph has a Properties view in which you can specify the data as well as the other properties of the graph.

- ❖ **To display the graph's Properties view:**
 - Select Properties from the graph's pop-up menu.

The Properties view for a graph has seven property pages in which you specify information about the graph. Table 15-3 lists the property pages and describes what each property page specifies.

Table 15-3: Property pages for graphs

Property page	What it specifies
Axis	Labels, scale, information about major and minor divisions for the category axes.
Data	Where to get the graph's data.
General	Various general graph properties, including border, graph colors, whether to size the graph to the full screen display, suppression in newspaper columns. Graph type, title, legend location. For 3D graphs, perspective, rotation, and elevation. For bar graphs, overlap, spacing and depth of bars.
Pointer	The pointer to use when the mouse is positioned over the graph.
Position	The x,y location of the upper left corner of the graph, its width and height, sliding options, the layer in which the graph is to be positioned. Whether the graph can be resized and moved at runtime.

Property page	What it specifies
Text	Text properties for text controls that display on the graph, including title, axis text, axis label, and legend. Text properties include font, font style, font size, alignment, rotation, color, display expression, display format.
Other	Descriptions and label for use by assistive technology tools.

Changing a graph's position and size

When you first place a graph in a DataWindow object, it is in the foreground—it sits above the bands in the DataWindow object. Unless you change this setting, the graph displays in front of any retrieved data.

The initial graph is also moveable and resizable, so users have complete flexibility as to the size and location of a graph at runtime. You can change these properties.

❖ To specify a graph's position and size:

- 1 Select Properties from the graph's pop-up menu and then select the Position page or the General page in the Properties view.
- 2 Select the settings for the following options on the Position property page:

Table 15-4: Settings on the Position property page for graphs

Setting	Meaning
Layer	<i>Background</i> — The graph displays behind other elements in the DataWindow object. <i>Band</i> — The graph displays in one particular band. If you choose this setting, you should resize the band to fit the graph. Often you will want to place a graph in the Footer band. As users scroll through rows in the DataWindow object, the graph remains at the bottom of the screen as part of the footer. <i>Foreground</i> — (Default) The graph displays above all other elements in the DataWindow object. Typically, if you choose this setting, you also make the graph movable so it will not obscure data while users display the DataWindow object.
Moveable	The graph can be moved in the Preview view and at runtime.
Resizable	The graph can be resized in the Preview view and at runtime.
Slide Left, Slide Up	The graph slides to the left or up to remove extra white space. For more information, see “Sliding controls to remove blank space in a DataWindow object” on page 219.

Setting	Meaning
X, Y	The location of the upper-left corner of the graph.
Width, Height	The width and height of the graph.

- 3 Select the settings for the following options on the General property page:

Table 15-5: Size and position settings on the General property page

Setting	Meaning
Size To Display	The graph fills the DataWindow object and resizes when users resize the DataWindow object. This setting is used with the Graph presentation style.
HideSnaked	Do not repeat graph after the first column in a DataWindow object using newspaper-style columns.

Associating data with a graph

When using a graph in a DataWindow object, you associate axes of the graph with columns in the DataWindow object.

The only way to get data into a graph in a DataWindow object is through columns in the DataWindow object. You cannot add, modify, or delete data in the graph except by adding, modifying, or deleting data in the DataWindow object.

You can graph data from any columns retrieved into the DataWindow object. The columns do not have to be displayed.

About the examples

The process of specifying data for a graph is illustrated below using the Printer table in the EAS Demo DB.

❖ **To specify data for a graph:**

- 1 If you are creating a new graph, the Graph Data dialog box displays. Otherwise, select Properties from the graph's pop-up menu and select the Data tab in the Properties view.
- 2 Fill in the boxes as described in the sections that follow, and click OK.

Specifying which rows to include in a graph

The Rows drop-down list allows you to specify which rows of data are graphed at any one time:

Table 15-6: Specifying which rows to include in a graph

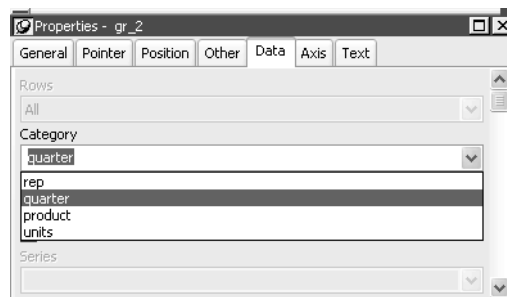
Setting	Meaning
All	Graphs the data from all the rows that have been retrieved but not filtered or deleted (that is, the rows in the primary buffer of the DataWindow object)
Page	Graphs only the data from the rows that are currently displayed on the page
Group n	Graphs only the data in the specified group (in a grouped DataWindow object)

If you select Group

If you are graphing data in the current group in a grouped DataWindow object and have several groups displayed at the same time, you should localize the graph in a group-related band in the Design view. This makes clear which group the graph represents. Usually, the group header band is the most appropriate band.

Specifying the categories

Specify the column or expression whose values determine the categories. In the Graph Data page in the Graph dialog box and on the Data page in the Properties view, you can select a column name from a drop-down list. In the Data tab, you can also type an expression:



There is an entry along the Category axis for each different value of the column or expression you specify.

Using display values of data

If you are graphing columns that use code tables, when data is stored with a data value but displayed to users with more meaningful display values, by default the graph uses the column's data values. To have the graph use a column's display values, use the `LookupDisplay` DataWindow expression function when specifying Category or Series. `LookupDisplay` returns a string that matches the display value for a column:

```
LookupDisplay ( column )
```

For more about code tables, see “Defining a code table” on page 251. For more about `LookupDisplay`, see the *DataWindow Object Reference*.

Specifying the values

DataWindow Designer populates the Value drop-down list. The list includes the names of all the retrieved columns as well as the following aggregate functions:

- Count for all non-numeric columns
- Sum for all numeric columns

Select an item from the drop-down list or type an expression (in the Properties view). For example, if you want to graph the sum of units sold, you can specify:

```
sum(units for graph)
```

To graph 110 percent of the sum of units sold, you can specify:

```
sum(units*1.1 for graph)
```

Specifying the series

Graphs can have one or more series.

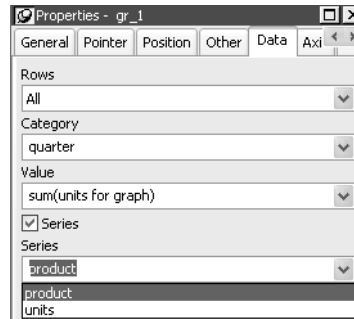
Single-series graphs

If you want only one series (that is, if you want to graph all retrieved rows as one series of values), leave the Series box empty.

Multiple-series graphs

If you want to graph more than one series, select the Series check box and specify the column that will provide the series values.

You can select column names from the drop-down list:



There is a set of data points for each different value of the column you specify here. For example, if you specify in the Series box a column that has 10 values, then your graph will have 10 series: one set of data points for each different value of the column.

Using expressions

You can also specify expressions for Series (on the Data page of the Properties view). For example, you could specify the following for Series:

```
Units / 1000
```

In this case, if a table had unit values of 10,000, 20,000, and 30,000, the graph would show series values of 10, 20, and 30.

Specifying multiple entries

You can specify more than one of the retrieved columns to serve as series. Separate multiple entries by commas.

You must specify the same number of entries in the Value box as you do in the Series box. The first value in the Value box corresponds to the first series identified in the Series box, the second value corresponds to the second series, and so on. The example about graphing actual and projected sales in “Examples” on page 432 illustrates this technique.

Examples

This section shows how to specify the data for several different graphs of the data in the Printer table in the EAS Demo DB. The table records quarterly unit sales of three printers by three sales representatives.

Table 15-7: The Printer table in the EAS Demo DB

Rep	Quarter	Product	Units
Simpson	Q1	Stellar	12
Jones	Q1	Stellar	18
Perez	Q1	Stellar	15
Simpson	Q1	Cosmic	33
Jones	Q1	Cosmic	5
Perez	Q1	Cosmic	26
Simpson	Q1	Galactic	6
Jones	Q1	Galactic	2
Perez	Q1	Galactic	1
...
Simpson	Q4	Stellar	30
Jones	Q4	Stellar	24
Perez	Q4	Stellar	36
Simpson	Q4	Cosmic	60
Jones	Q4	Cosmic	52
Perez	Q4	Cosmic	48
Simpson	Q4	Galactic	3
Jones	Q4	Galactic	3
Perez	Q4	Galactic	6

Graphing total sales

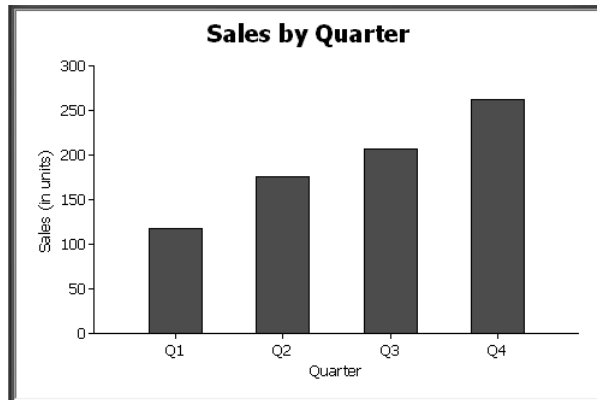
To graph total sales of printers in each quarter, retrieve all the columns into a DataWindow object and create a graph with the following settings on the Data page in the Properties view:

- Set Rows to All
- Set Category to quarter
- Set Value to sum(units for graph)

Leave the Series check box and text box empty.

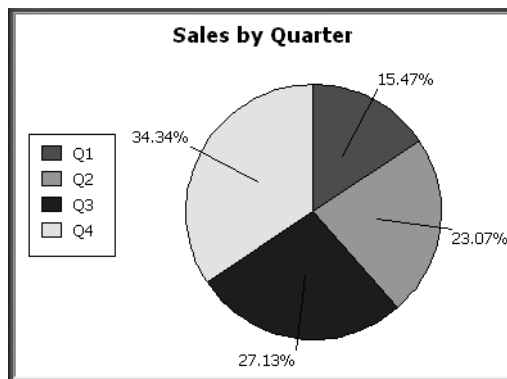
The Quarter column serves as the category. Because the Quarter column has four values (Q1, Q2, Q3, and Q4), there will be four categories along the Category axis. You want only one series (total sales in each quarter), so you can leave the Series box empty, or type a string literal to identify the series in a legend. Setting Value to `sum(units for graph)` graphs total sales in each quarter.

Here is the resulting column graph. DataWindow Designer automatically generates the category text based on the data in the table:



In the preceding graph, there is one set of data points (one series) across four quarters (the category values).

The following is a pie graph, which has exactly the same properties as the preceding column graph except for the type, which is 3D Pie:



In pie graphs, categories are shown in the legend.

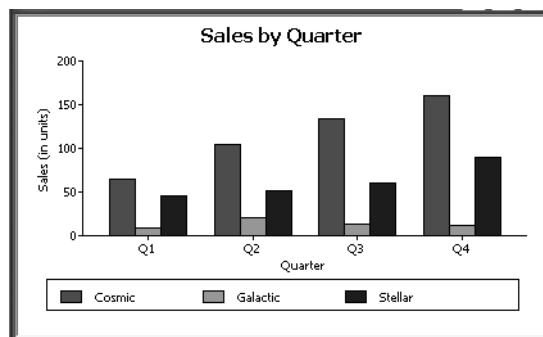
Graphing unit sales of each printer

To graph total quarterly sales of each printer, retrieve all the columns into a DataWindow object and create a graph with the following settings on the Data page in the Properties view:

- Set Rows to All
- Set Category to quarter
- Set Value to `sum(units for graph)`
- Select the Series check box
- Set Series to product

You want a different series for each printer, so the column Product serves as the series. Because the Product column has three values (Cosmic, Galactic, and Stellar), there will be three series in the graph. As in the first example, you want a value for each quarter, so the Quarter column serves as the category, and you want to graph total sales in each quarter, so the Value box is specified as `sum(units for graph)`.

Here is the resulting graph. DataWindow Designer automatically generates the category and series labels based on the data in the table. The series labels display in the graph's legend:

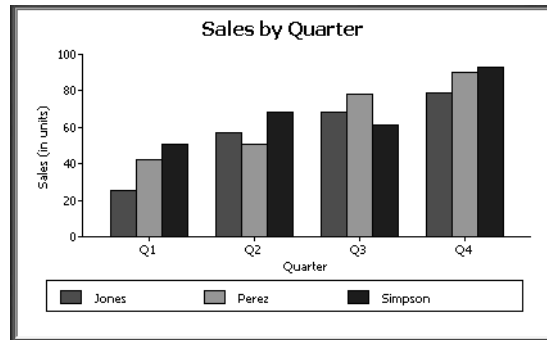


Graphing unit sales by representative

To graph quarterly sales made by each representative, create a graph with the following settings on the Data page in the Properties view:

- Set Rows to All
- Set Category to quarter
- Set Value to `sum(units for graph)`
- Select the Series check box
- Set Series to rep

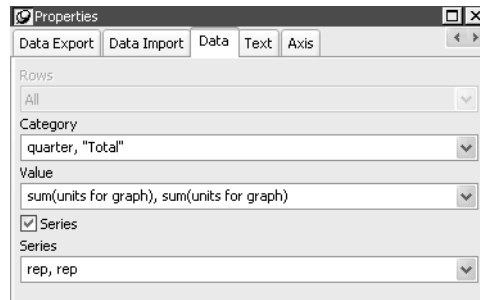
Here is the resulting graph:



Graphing unit sales by representative and total sales

To graph quarterly sales made by each representative, plus total sales for each printer, create a graph with the following settings on the Data page in the Properties view:

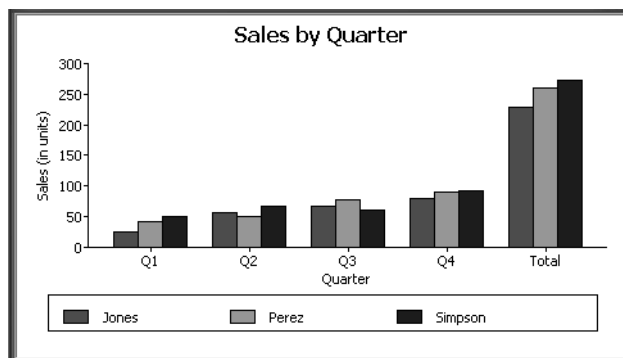
- Set Rows to All
- Set Category to quarter, "Total"
- Set Value to `sum(units for graph), sum(units for graph)`
- Select the Series check box
- Set Series to `rep, rep`



Here you have two types of categories: the first is Quarter, which shows quarterly sales, as in the previous graph. You also want a category for total sales. There is no corresponding column in the DataWindow object, so you can simply type the literal "Total" to identify the category. You separate multiple entries with a comma.

For each of these category types, you want to graph the sum of units sold for each representative, so the Value and Series values are repeated.

Here is the resulting graph:



Notice that DataWindow Designer uses the literal “Total” supplied in the Category box in the Graph Data window as a value in the Category axis.

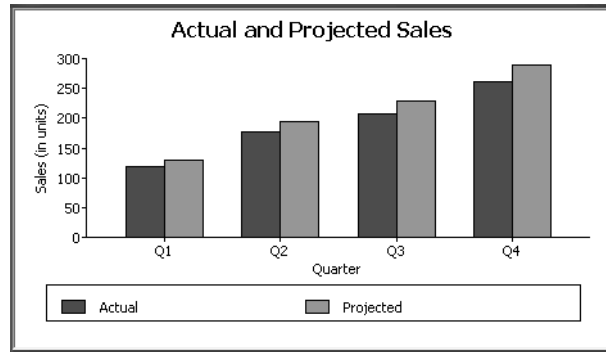
Graphing actual and projected sales

To graph total quarterly sales of all printers and projected sales for next year, create a graph with the following settings on the Data page in the Properties view (you assume that sales will increase by 10% next year):

- Set Rows to All
- Set Category to quarter
- Set Value to `sum(units for graph), sum(units*1.1 for graph)`
- Select the Series check box
- Set Series to 'Actual', 'Projected'

You are using labels to identify two series, Actual and Projected. Note the single quotation marks around the literals. For Values, you enter the expressions that correspond to Actual and Projected sales. For Actual, you use the same expression as in the examples above, `sum(units for graph)`. For Projected sales, you multiply each unit sale by 1.1 to get the 10 percent increase. Therefore, the second expression is `sum(units*1.1 for graph)`.

Here is the resulting graph. DataWindow Designer uses the literals you typed for the series as the series labels in the legend:



Using overlays

It is often useful to call special attention to one of the series in a graph, particularly in a bar or column graph. You can do that by defining the series as an overlay. An overlay series is graphed as a line on top of the other series in the graph. To define a series as an overlay, define it as follows:

- If specifying a column name to identify the series, specify this for the series:

`"@overlay~t" + ColumnName`

- If using a label to identify the series, specify this for the series:

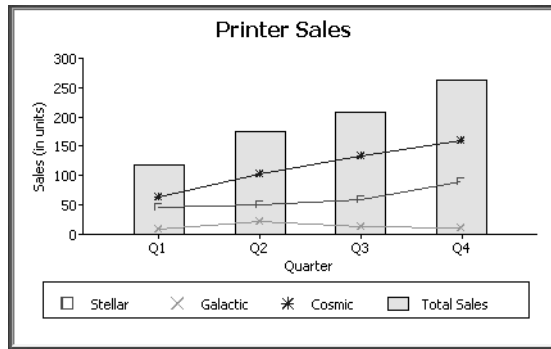
`"@overlay~tSeriesLabel "`

Examples

To graph sales in each quarter and overlay the sales of each individual printer, specify the graph's data as in “Graphing unit sales of each printer” on page 434, but use the following expression in the Series box:

`"Total Sales", "@overlay~t" + product`

Here is the resulting graph:



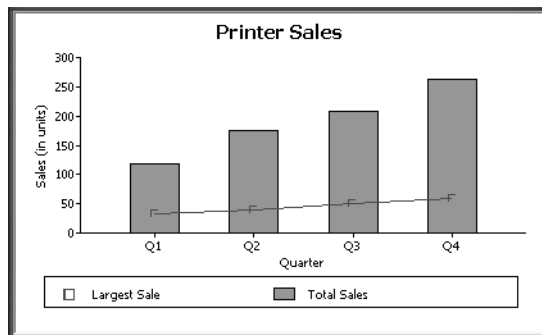
To graph unit sales of printers by quarter and overlay the largest sale made in each quarter, change the Value expression to this:

```
sum(units for graph), max(units for graph)
```

Change the Series expression to this:

```
"Total Sales", "@overlay~tLargets Sale"
```

Here is the resulting graph:



Using the Graph presentation style

Instead of embedding a graph in a DataWindow object, you can use the Graph presentation style to create a DataWindow object that is only a graph—the underlying data is not displayed.

One advantage of the Graph presentation style is that the graph resizes automatically if users resize the DataWindow control associated with the graph DataWindow object at runtime.

❖ **To use the Graph presentation style:**

- 1 Select File>New from the menu bar.

The New dialog box displays.

- 2 Select the DataWindow tab, select the Graph presentation style, and click OK.

- 3 Specify the data source for the DataWindow object. If you want data to be retrieved into the Preview view automatically, select the Retrieve on Preview check box.

For more information, see Chapter 4, “Defining DataWindow Objects.”

- 4 Enter the definitions for the series, categories, and values, as described in “Associating data with a graph” on page 428, and click Next.

Note that when using the Graph presentation style, the graph always graphs all rows; you cannot specify page or group.

- 5 Enter a title for the graph, select a graph type, and click Next.

- 6 Review your specifications and click Finish.

A model of the graph displays in the Design view.

- 7 Specify the properties of the graph, as described in “Defining a graph's properties” next.

- 8 Save the DataWindow object in a library.

- 9 Associate the graph DataWindow object with a DataWindow control on a form.

At runtime, the graph fills the entire control and resizes when the control is resized.

Defining a graph's properties

This section describes properties of a graph. To define the properties of a graph, you use the graph's Properties view. For general information about the property pages, see “Using the graph's Properties view” on page 426.

Using the General property page in the graph's Properties view

You name a graph and define its basic properties on the General property page in the graph's Properties view.

❖ **To specify the basic properties of a graph:**

- Select Properties from the graph's pop-up menu and then select the General page in the Properties view.

About the model graph in the Design view

As you modify a graph's properties, DataWindow Designer updates the model graph shown in the Design view so that you can get an idea of the graph's basic layout:

- DataWindow Designer uses the graph title and axis labels you specify.
- DataWindow Designer uses sample data (not data from your DataWindow object) to illustrate series, categories, and values.

In Preview view, DataWindow Designer displays the graph with data.

Naming a graph

You can modify graphs at runtime. To reference a graph in code, you use its name.

❖ **To name a graph:**

- On the General properties page for the graph, assign a meaningful name to the graph in the Name box.

Defining a graph's title

The title displays at the top of the graph.

❖ **To specify a graph's title:**

- On the General properties page for the graph, enter a title in the Title box.

Multiline titles

You can force a new line in a title by embedding ~n.

For information about specifying properties for the title text, see “Specifying text properties for titles, labels, axes, and legends” on page 442.

- Specifying the type of graph You can change the graph type at any time in the development environment. (To change the type at runtime, modify a graph's GraphType property.)
- ❖ **To specify the graph type:**
 - On the General properties page for the graph, select a graph type from the Graph Type drop-down list.
- Using legends A legend provides a key to your graph's series.
- ❖ **To include a legend for a series in a graph:**
 - On the General properties page for the graph, specify where you want the legend to appear by selecting a value in the Legend drop-down list.

For information on specifying text properties for the legend, see “Specifying text properties for titles, labels, axes, and legends” on page 442.
- Specifying a border For a graph control in a DataWindow object, you can specify the border that DataWindow Designer places around a graph.
- ❖ **To specify a border for a graph:**
 - On the General properties page for the graph, select the type of border to use from the Border drop-down list.
- Specifying point of view in 3D graphs If you are defining a 3D graph, you can specify the point of view that DataWindow Designer uses when displaying the graph.
- ❖ **To specify a 3D graph's point of view:**
 - 1 On the General properties page for the graph, adjust the point of view along the three dimensions of the graph:
 - To change the perspective, move the Perspective slider.
 - To rotate the graph, move the Rotation slider.
 - To change the elevation, move the Elevation slider.
 - 2 Define the depth of the graph (the percent the depth is of the width of the graph) by using the Depth slider.

Sorting data for series and categories

You can specify how to sort the data for series and categories. By default, the data is sorted in ascending order.

- ❖ **To specify how to sort the data for series and categories in a graph:**
 - 1 Select Properties from the graph's pop-up menu and then select the Axis page in the Properties view.
 - 2 Select the axis for which you want to specify sorting.
 - 3 Scroll to Sort, the last option on the Axis page, and select Ascending (order), Descending (order), or Unsorted.

Specifying text properties for titles, labels, axes, and legends

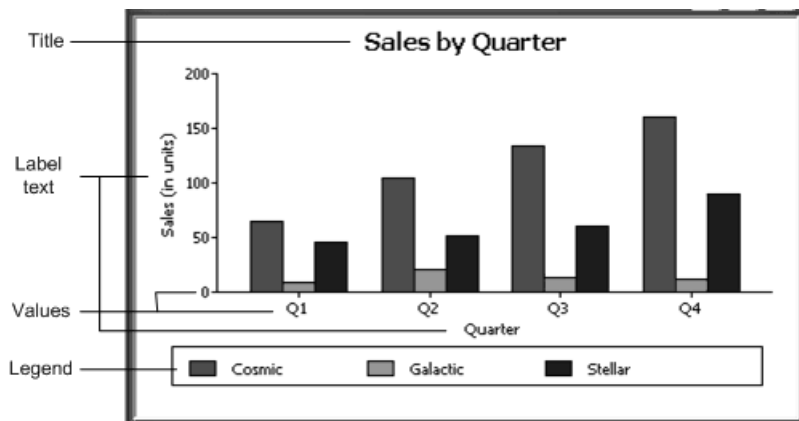
A graph can have four text elements:

Title

Labels for the axes

Text that shows the values along the axes

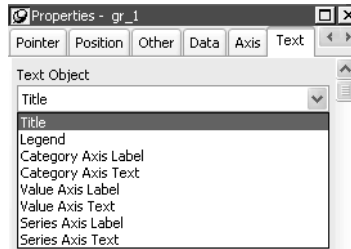
Legend



You can specify properties for each text element.

❖ **To specify text properties for the title, labels, axis values, and legend of a graph:**

- 1 Select Properties from the graph's pop-up menu and then select the Text page in the Properties view.
- 2 Select a text element from the list in the Text Object drop-down list:



- 3 Specify the font and its characteristics.

Using Auto Size

With Auto Size in effect, DataWindow Designer resizes the text appropriately whenever the graph is resized. With Auto Size disabled, you specify the font size of a text element explicitly.

❖ **To have DataWindow Designer automatically size a text element in a graph:**

- 1 On the Text properties page for the graph, select a text element from the list in the Text Object drop-down list.
- 2 Select the Autosize check box (this is the default).

❖ **To specify a font size for a text element in a graph:**

- 1 On the Text properties page for the graph, select a text element from the list in the Text Object drop-down list.
- 2 Clear the Autosize check box.
- 3 Select the Font size in the Size drop-down list.

Rotating text

For all the text elements, you can specify the number of degrees by which you want to rotate the text.

❖ **To specify rotation for a text element in a graph:**

- 1 On the Text properties page for the graph, select a text element from the list in the Text Object drop-down list.

- 2 Specify the rotation you want in the Escapement box using tenths of a degree (450 means 45 degrees).

Changes you make here are shown in the model graph in the Design view and in the Preview view.

Using display formats

❖ **To use a display format for a text element in a graph:**

- 1 On the Text properties page for the graph, select a text element from the list in the Text Object drop-down list.
- 2 Type a display format in the Format box or choose one from the pop-up menu. To display the pop-up menu, click the button to the right of the Format box.

Modifying display expressions

You can specify an expression for the text that is used for each graph element. The expression is evaluated at execution time.

❖ **To specify an expression for a text element in a graph:**

- 1 On the Text properties page for the graph, select a text element from the list in the Text Object drop-down list.
- 2 Click the button next to the Display Expression box.
The Modify Expression dialog box displays.
- 3 Specify the expression.

You can paste functions, column names, and operators. Included with column names in the Columns box are statistics about the columns, such as counts and sums.

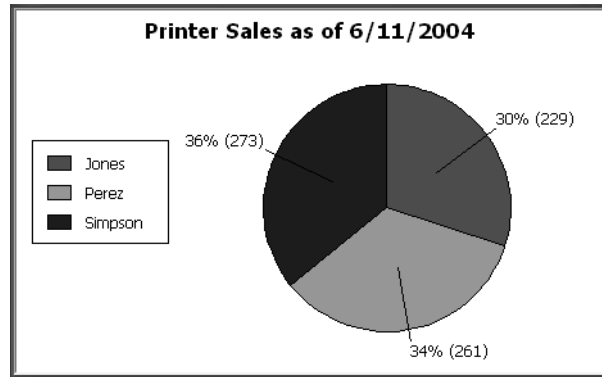
- 4 Click OK to return to the graph's Properties view.

Example

By default, when you generate a pie graph, DataWindow Designer puts the title at the top and labels each slice of the pie with the percentage each slice represents of the whole. Percentages are accurate to two decimal places.

The following graph has been enhanced as follows:

- The current date displays in the title
- The percentages are rounded to integers
- The raw data for each slice is shown in addition to the percentages



To accomplish this, the display expressions were modified for the title and pie graph labels:

Element	Original expression	Modified expression
Title	title	title + " as of " + date(today())
Pie graph labels	if(seriescount > 1, series, string(percentofseries, "0.00%"))	if(seriescount > 1, series, string(percentofseries, "0%") + " (" + value + ")")

Specifying overlap and spacing

With bar and column charts, you can specify the properties in Table 15-8.

Table 15-8: Overlap and spacing properties for bar and column charts

Property	Meaning
Overlap	The percentage by which bars or columns overlap each other. The default is 0 percent, meaning no overlap.
Spacing	The amount of space to leave between bars or columns. The default is 100 percent, which leaves a space equal to the width of a bar or column.

- ❖ **To specify overlap and spacing for the bars or columns in a graph:**
 - 1 Select Properties from the graph's pop-up menu and then select the Graph tab.
 - 2 Specify a percentage for Overlap (% of width) and Spacing (% of width).

Specifying axis properties

Graphs have two or three axes. You specify the axes' properties in the Axis page in the graph's Properties view.

❖ **To specify properties for an axis of a graph:**

- 1 Select Properties from the graph's pop-up menu and then select the Axis page in the Properties view.
- 2 Select the Category, the Value, or the Series axis from the Axis drop-down list.

If you are not working with a 3D graph, the Series Axis options are disabled.

- 3 Specify the properties as described next.

Specifying text properties

You can specify the characteristics of the text that displays for each axis. Table 15-9 shows the two kinds of text associated with an axis.

Table 15-9: Text types associated with each axis of a graph

Type of text	Meaning
Text	Text that identifies the values for an axis.
Label	Text that describes the axis. You specify the label text in a painter. You can use ~n to embed a new line within a label.

For information on specifying properties for the text, see “Specifying text properties for titles, labels, axes, and legends” on page 442.

Specifying datatypes

The data graphed along the Value, Category, and Series axes has an assigned datatype. The Series axis always has the datatype String. The Value and Category axes can have the datatypes listed in Table 15-10.

Table 15-10: Datatypes for Value and Category axes

Axis	Possible datatypes
Both axes (for scatter graph)	Number, Date, Time
Value (other graph types)	Number, Date, DateTime, Time
Category (other graph types)	String, Number, Date, DateTime, Time

DataWindow Designer automatically assigns the datatypes based on the datatype of the corresponding column; you do not specify them.

Scaling axes

You can specify the properties listed in Table 15-11 to define the scaling used along numeric axes.

Table 15-11: Properties for scaling on numeric axes

Property	Meaning
Autoscale	If selected (the default), DataWindow Designer automatically assigns a scaling for the numbers along the axis.
Round To, Round To Unit	Specifies how to round the end points of the axis (note that this just rounds the range displayed along the axis; it does not round the data itself). You can specify a number and a unit. The unit is based on the datatype; you can specify Default as the unit to have DataWindow Designer decide for you. For example, if the Value axis is a Date column, you can specify that you want to round the end points of the axis to the nearest five years. In this case, if the largest data value is the year 1993, the axis extends up to 1995, which is 1993 rounded to the next highest five-year interval.
Minimum Value, Maximum Value	The smallest and largest numbers to appear on the axis (disabled if you have selected Autoscale).
Scale Type	Specifies linear or logarithmic scaling (common or natural).
Scale Value	Specifies whether values are displayed as actual values or as a cumulative value, a percentage, or a cumulative percentage.

Using major and minor divisions

You can divide axes into divisions. Each division is identified by a tick mark, which is a short line that intersects an axis. In the Sales by Printer graphs shown in “Examples” on page 432, the graph's Value axis is divided into major divisions of 50 units each. DataWindow Designer divides the axes automatically into major divisions.

❖ To define divisions for an axis of a graph:

- To divide an axis into a specific number of major divisions, type the number of divisions you want in the MajorDivisions box.

Leave the number 0 to have DataWindow Designer automatically create divisions. DataWindow Designer labels each tick mark in major divisions. If you do not want each tick mark labeled, enter a value in the DisplayEveryNLabels box. For example, if you enter 2, DataWindow Designer labels every second tick mark for the major divisions.
- To use minor divisions, which are divisions within each major division, type the appropriate number in the MinorDivisions box. To use no minor divisions, leave the number 0.

When using logarithmic axes

If you want minor divisions, specify 1; otherwise, specify 0.

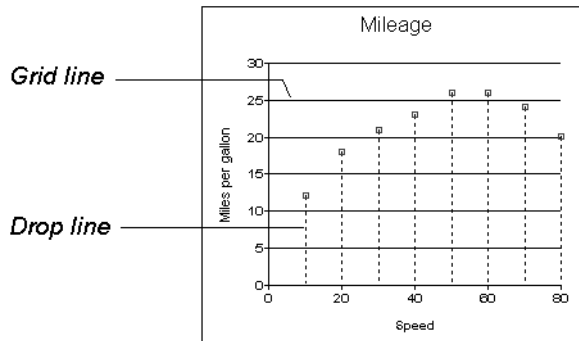
Representing divisions with grid and drop lines

You can specify lines to represent the divisions as described in Table 15-12 and illustrated in Figure 15-1.

Table 15-12: Representing graph divisions with grid and drop lines

Line	Meaning
Grid line	A line that extends from a tick mark across the graph. Grid lines make graphs easier to read.
Drop line	A line that extends vertically from a data point to its axis (not available for all graph types).

Figure 15-1: Grid and drop lines in a graph



Using line styles

You can define line styles for the components of a graph listed in Table 15-13.

Table 15-13: Components of a graph that can have line styles

Component	Meaning
PrimaryLine	The axis itself
SecondaryLine	The axis parallel to and opposite the primary axis
OriginLine	A grid line that represents the value zero
Frame	The frame for the axis in 3D graphs (disabled for 2D graphs)

Specifying a pointer

You can specify a pointer to use when the mouse is over a graph at runtime.

❖ **To specify a pointer for a graph:**

- 1 Select Properties from the graph's pop-up menu and then select the Pointer page in the Properties view.
- 2 Select a stock pointer from the list, or select a *CUR* file containing a pointer.

Appendixes

This part contains information about the rules for naming objects in DataWindow Designer, the extended attribute system tables, and starting DataWindow Designer from a command line.

Identifiers

About this chapter

You use identifiers to name objects. This chapter describes valid identifiers.

Contents

Topic	Page
Rules	451
Reserved words	452

Rules

Identifiers:

- Must start with a letter
- Can have up to 40 characters, but no spaces
- Are case insensitive (PART, Part, and part are identical)
- Can include any combination of letters, numbers, and these special characters:
 - Dash
 - _ Underscore
 - \$ Dollar sign
 - # Number sign
 - % Percent sign

Joining words in multiword names

Since DataWindow Designer does not allow spaces in identifiers, you can use any of the following techniques to join words in an identifier:

- Initial caps (for example, IncomeJanuary)
- Dashes (for example, northeast-sales)
- Underscores (for example, quantity_on_hand)

Examples

Here are some *valid* identifiers:

```
first_quarter_summary
EMPLOYEE_LABELS
EmployeeSalarySummary
Employee_by_#
```

Here are some *invalid* identifiers:

```
2nd-quarter // Does not start with a letter
emp list // Contains a space
Employee'sInfo // Contains an invalid character
```

Reserved words

You cannot use the following reserved words as identifiers, because DataWindow Designer uses them internally:

Table 15-14: Reserved words

alias	event	not	static
and	execute	of	step
autoinstantiate	exit	on	subroutine
call	external	open	super
case	false	or	system
catch	fetch	parent	systemread
choose	finally	post	systemwrite
close	first	prepare	then
commit	for	prior	this
connect	forward	private	throw
constant	from	privateread	throws
continue	function	privatewrite	to
create	global	procedure	trigger
cursor	goto	protected	true
declare	halt	protectedread	try
delete	if	protectedwrite	type
describe	immediate	prototypes	until
descriptor	indirect	public	update
destroy	insert	readonly	updateblob
disconnect	into	ref	using
do	intrinsic	return	variables
dynamic	is	rollback	while
else	last	rpcfunc	with
elseif	library	select	within
end	loop	selectblob	_debug
enumerated	next	shared	

The Extended Attribute System Tables

About this appendix

This appendix describes each column in the extended attribute system tables.

Contents

Topic	Page
About the extended attribute system tables	453
The extended attribute system tables	454
Edit style types for the PBCatEdt table	457

About the extended attribute system tables

DataWindow Designer stores information you provide for a database table (such as the text to use for labels and headings for the columns, validation rules, display formats, and edit styles) in system tables in your database. These system tables are called the extended attribute system tables. The tables contain all the information related to the extended attributes for the tables and columns in the database. The extended attributes are used in DataWindow objects.

The system tables

There are five extended attribute system tables.

Table B-1: List of extended attribute system tables

Table	Contains information about
PBCatTbl	Tables in the database
PBCatCol	Columns in the database
PBCatFmt	Display formats
PBCatVld	Validation rules
PBCatEdt	Edit styles

What to do with the tables

You can open and look at these tables in the Database painter just like other tables. You might want to create a report of the extended attribute information used in your database by building a DataWindow object whose data source is the extended attribute system tables.

Caution

You should not change the values in the extended attribute system tables. DataWindow Designer maintains this information automatically whenever you change information for a table or column in the Database painter.

The extended attribute system tables

This section lists and describes all of the columns in each of the extended attribute system tables.

Table B-2: The PBCatTbl table

Column	Column name	Description
1	pbt_tnam	Table name
2	pbt_tid	Adaptive Server Enterprise Object ID of table (used for Adaptive Server Enterprise only)
3	pbt_ownr	Table owner
4	pbd_fhgt	Data font height, PowerBuilder units
5	pbd_fwgt	Data font stroke weight (400=Normal, 700=Bold)
6	pbd_fitl	Data font Italic (Y=Yes, N=No)
7	pbd_funl	Data font Underline (Y=Yes, N=No)
8	pbd_fchr	Data font character set (0=ANSI, 2=Symbol, 255=OEM)
9	pbd_fptc	Data font pitch and family (see note)
10	pbd_ffce	Data font typeface
11	pbh_fhgt	Headings font height, PowerBuilder units
12	pbh_fwgt	Headings font stroke weight (400=Normal, 700=Bold)
13	pbh_fitl	Headings font Italic (Y=Yes, N=No)
14	pbh_funl	Headings font Underline (Y=Yes, N=No)
15	pbh_fchr	Headings font character set (0=ANSI, 2=Symbol, 255=OEM)
16	pbh_fptc	Headings font pitch and family (see note)

Column	Column name	Description
17	pbh_ffce	Headings font typeface
18	pbl_fhgt	Labels font height, PowerBuilder units
19	pbl_fwgt	Labels font stroke weight (400=Normal, 700=Bold)
20	pbl_fitl	Labels font Italic (Y=Yes, N=No)
21	pbl_funl	Labels font Underline (Y=Yes, N=No)
22	pbl_fchr	Labels font character set (0=ANSI, 2=Symbol, 255=OEM)
23	pbl_fptc	Labels font pitch and family (see note)
24	pbl_ffce	Labels font typeface
25	pbt_cmnt	Table comments

About font pitch and family

Font pitch and family is a number obtained by adding together two constants:

Pitch: 0=Default, 1=Fixed, 2=Variable

Family: 0=No Preference, 16=Roman, 32=Swiss, 48=Modern, 64=Script, 80=Decorative

Table B-3: The PBCatCol table

Column	Column name	Description
1	pbc_tnam	Table name
2	pbc_tid	Adaptive Server Enterprise Object ID of table (used for Adaptive Server Enterprise only)
3	pbc_ownr	Table owner
4	pbc_cnam	Column name
5	pbc_cid	Adaptive Server Enterprise Column ID (used for Adaptive Server Enterprise only)
6	pbc_labl	Label
7	pbc_lpos	Label position (23=Left, 24=Right)
8	pbc_hdr	Heading
9	pbc_hpos	Heading position (23=Left, 24=Right, 25=Center)
10	pbc_jtfy	Justification (23=Left, 24=Right)
11	pbc_mask	Display format name
12	pbc_case	Case (26=Actual, 27=UPPER, 28=lower)
13	pbc_hght	Column height, PowerBuilder units
14	pbc_wdth	Column width, PowerBuilder units
15	pbc_ptrn	Validation rule name
16	pbc_bmap	Bitmap/picture (Y=Yes, N=No)

Column	Column name	Description
17	pbk_init	Initial value
18	pbk_cmnt	Column comments
19	pbk_edit	Edit style name
20	pbk_tag	(Reserved)

Table B-4: The PBCatFmt table

Column	Column name	Description
1	pbf_name	Display format name
2	pbf_frm	Display format
3	pbf_type	Data type to which format applies
4	pbf_cntr	Concurrent-usage flag

Table B-5: The PBCatVld table

Column	Column name	Description
1	pbv_name	Validation rule name
2	pbv_vald	Validation rule
3	pbv_type	Data type to which validation rule applies
4	pbv_cntr	Concurrent-usage flag
5	pbv_msg	Validation error message

Table B-6: The PBCatEdt table

Column	Column name	Description
1	pbe_name	Edit style name
2	pbe_edit	Format string (edit style type dependent; see below)
3	pbe_type	Edit style type (see below)
4	pbe_cntr	Revision counter (increments each time edit style is altered)
5	pbe_seqn	Row sequence number for edit types requiring more than one row in PBCatEdt table
6	pbe_flag	Edit style flag (edit style type dependent; see below)
7	pbe_work	Extra field (edit style type dependent; see below)

Edit style types for the *PBCatEdt* table

Table B-7 shows the edit style types available for the *PBCatEdt* table.

Table B-7: Edit style types for the *PBCatEdt* table

Edit style type	pbe_type value (column 3)
CheckBox	85
RadioButton	86
DropDownListBox	87
DropDownDataWindow	88
Edit	89
Edit Mask	90

CheckBox edit style (code 85)

Table B-8 shows a sample row in the *PBCatEdt* table for a CheckBox edit style. Table B-9 shows the meaning of the values in Table B-8.

Table B-8: Sample row in *PBCatEdt* for a CheckBox edit style

Name	Edit	Type	Cntr	Seqn	Flag	Work
MyEdit	<i>Text</i>	85	1	1	<i>Flag</i>	
MyEdit	<i>OnValue</i>	85	1	2	0	
MyEdit	<i>OffValue</i>	85	1	3	0	
MyEdit	<i>ThirdValue</i>	85	1	4	0	

Table B-9: Values used in CheckBox edit style sample

Value	Meaning
<i>Text</i>	CheckBox text
<i>OnValue</i>	Data value for On state
<i>OffValue</i>	Data value for Off state
<i>ThirdValue</i>	Data value for Third state (this row exists only if 3 State is checked for the edit style—bit 30 of <i>Flag</i> is 1)

Value	Meaning
<i>Flag</i>	<p>32-bit flag. Low-order four hex digits are generic edit type; high-order four are styles within the type. A 1 in any bit indicates the corresponding style is checked. A 0 in any bit indicates the corresponding style is unchecked.</p> <ul style="list-style-type: none"> Bit 31: Left Text Bit 30: 3 State Bit 29: 3D Bit 28: Scale Box Bits 27 – 16 (3 hex digits): Not used (set to 0) Bits 15 – 4 (3 hex digits): Always 0 for CheckBox edit style Bit 3: Always 0 for CheckBox edit style Bit 2: Always 1 for CheckBox edit style Bit 1: Always 0 for CheckBox edit style Bit 0: Always 0 for CheckBox edit style

RadioButton edit style (code 86)

Table B-10 shows a sample row in the PBCatEdt table for a RadioButton edit style. Table B-11 shows the meaning of the values in Table B-10.

Table B-10: Sample row in PBCatEdt for a RadioButton edit style

Name	Edit	Type	Cntr	Seqn	Flag	Work
MyEdit	<i>Columns</i>	86	1	1	<i>Flag</i>	
MyEdit	<i>Display1</i>	86	1	2	0	
MyEdit	<i>Data1</i>	86	1	3	0	
MyEdit	<i>Display2</i>	86	1	4	0	
MyEdit	<i>Data2</i>	86	1	5	0	

Table B-11: Values used in RadioButton edit style sample

Value	Meaning
<i>Columns</i>	Character representation (in decimal) of number of columns (buttons) across.
<i>Display1</i>	Display value for first button.
<i>Data1</i>	Data value for first button.
<i>Display2</i>	Display value for second button.
<i>Data2</i>	Data value for second button. Display and data values are repeated in pairs for each radio button defined in the edit style.

Value	Meaning
<i>Flag</i>	<p>32-bit flag. Low-order four hex digits are generic edit type; high-order four are styles within the type. A 1 in any bit indicates the corresponding style is checked. A 0 in any bit indicates the corresponding style is unchecked.</p> <ul style="list-style-type: none"> Bit 31: Left Text Bit 30: 3D Bit 29: Scale Circles Bit 38: Not used (set to 0) Bits 27 – 16 (3 hex digits): Not used (set to 0) Bits 15 – 4 (3 hex digits): Always 0 for RadioButton edit style Bit 3: Always 1 for RadioButton edit style Bit 2: Always 0 for RadioButton edit style Bit 1: Always 0 for RadioButton edit style Bit 0: Always 0 for RadioButton edit style

DropDownListBox edit style (code 87)

Table B-12 shows a sample row in the PBCatEdt table for a DropDownListBox edit style. Table B-13 shows the meaning of the values in Table B-12.

Table B-12: Sample row in PBCatEdt for a DropDownListBox edit style

Name	Edit	Type	Cntr	Seqn	Flag	Work
MyEdit	<i>Limit</i>	87	1	1	<i>Flag</i>	<i>Key</i>
MyEdit	<i>Display1</i>	87	1	2	0	
MyEdit	<i>Data1</i>	87	1	3	0	
MyEdit	<i>Display2</i>	87	1	4	0	
MyEdit	<i>Data2</i>	87	1	5	0	

Table B-13: Values used in DropDownListBox edit style sample

Value	Meaning
<i>Limit</i>	Character representation (in decimal) of the <i>Limit</i> value.
<i>Key</i>	One-character accelerator key.
<i>Display1</i>	Display value for first entry in code table.
<i>Data1</i>	Data value for first entry in code table.
<i>Display2</i>	Display value for second entry in code table.
<i>Data2</i>	Data value for second entry in code table.
	Display and data values are repeated in pairs for each entry in the code table.

Value	Meaning
<i>Flag</i>	<p>32-bit flag. Low-order four hex digits are generic edit type; high-order four are styles within the type. A 1 in any bit indicates the corresponding style is checked. A 0 in any bit indicates the corresponding style is unchecked.</p> <ul style="list-style-type: none"> Bit 31: Sorted Bit 30: Allow editing Bit 29: Auto HScroll Bit 28: VScroll bar Bit 27: Always show list Bit 26: Always show arrow Bit 25: Uppercase Bit 24: Lowercase (if bits 25 and 24 are both 0, then case is Any) Bit 23: Empty string is NULL Bit 22: Required field Bit 21: Not used (set to 0) Bit 20: Not used (set to 0) Bits 19 – 16 (1 hex digit): Not used (set to 0) Bits 15 – 4 (3 hex digits): Always 0 for DropDownListBox edit style Bit 3: Always 0 for DropDownListBox edit style Bit 2: Always 0 for DropDownListBox edit style Bit 1: Always 1 for DropDownListBox edit style Bit 0: Always 0 for DropDownListBox edit style

DropDownDataWindow edit style (code 88)

Table B-14 shows a sample row in the PBCatEdt table for a DropDownDataWindow edit style. Table B-15 shows the meaning of the values in Table B-14.

Table B-14: Sample row in PBCatEdt for a DropDownDataWindow edit style

Name	Edit	Type	Cntr	Seqn	Flag	Work
MyEdit	<i>DataWin</i>	88	1	1	<i>Flag</i>	<i>Limit</i>
MyEdit	<i>DataCol</i>	88	1	2	0	<i>Key</i>
MyEdit	<i>DisplayCol</i>	88	1	3	0	<i>Width%</i>

Table B-15: Values used in DropDownDataWindow edit style sample

Value	Meaning
<i>DataWin</i>	Name of DataWindow object to use.
<i>DataCol</i>	Data column from DataWindow object.
<i>DisplayCol</i>	Display column from DataWindow object.

Value	Meaning
<i>Limit</i>	Character representation (in decimal) of <i>Limit</i> value.
<i>Key</i>	One-character accelerator key.
<i>Width%</i>	Width of the dropdown part of the DropDownDataWindow in %.
<i>Flag</i>	<p>32-bit flag. Low-order four hex digits are generic edit type; high-order four are styles within the type. A 1 in any bit indicates the corresponding style is checked. A 0 in any bit indicates the corresponding style is unchecked.</p> <ul style="list-style-type: none"> Bit 31: Allow editing Bit 30: Auto HScroll Bit 29: VScroll bar Bit 28: Always show list Bit 27: Uppercase Bit 26: Lowercase (if bits 27 and 26 are both 0, then case is Any) Bit 25: HScroll bar Bit 24: Split horizontal scroll bar Bit 23: Empty string is NULL Bit 22: Required field Bit 21: Always show arrow Bit 20: Not used (set to 0) Bits 19 – 16 (1 hex digit): Not used (set to 0) Bits 15 – 8 (2 hex digits): Always 0 for DropDownDataWindow edit style Bit 7: Always 0 for DropDownDataWindow edit style Bit 6: Always 0 for DropDownDataWindow edit style Bit 5: Always 0 for DropDownDataWindow edit style Bit 4: Always 1 for DropDownDataWindow edit style Bit 3 – 0 (1 hex digit): Always 0 for DropDownDataWindow edit style

Edit edit style (code 89)

Table B-16 shows a sample row in the PBCatEdt table for an Edit edit style. Table B-17 shows the meaning of the values in Table B-16.

About the example

This example shows an Edit edit style using a code table of display and data values. There is a pair of rows in PBCatEdt for each entry in the code table *only if* bit 23 of *Flag* is 1.

For information about code tables in edit styles, see Chapter 8, “Displaying and Validating Data.”

Table B-16: Sample row in PBCatEdt for an Edit edit style

Name	Edit	Type	Cntr	Seqn	Flag	Work
MyEdit	<i>Limit</i>	89	1	1	<i>Flag</i>	<i>Key</i>
MyEdit	<i>Format</i>	89	1	2	0	<i>Focus</i>
MyEdit	<i>Display1</i>	89	1	3	0	
MyEdit	<i>Data1</i>	89	1	4	0	
MyEdit	<i>Display2</i>	89	1	5	0	
MyEdit	<i>Data2</i>	89	1	6	0	

Table B-17: Values used in Edit edit style sample

Value	Meaning
<i>Limit</i>	Character representation (in decimal) of <i>Limit</i> value.
<i>Key</i>	One-character accelerator key.
<i>Format</i>	Display format mask.
<i>Focus</i>	Character "1" if Show Focus Rectangle is checked. NULL otherwise.
<i>Flag</i>	<p>32-bit flag. Low-order four hex digits are generic edit type; high-order four are styles within the type. A 1 in any bit indicates the corresponding style is checked. A 0 in any bit indicates the corresponding style is unchecked.</p> <ul style="list-style-type: none"> Bit 31: Uppercase Bit 30: Lowercase (if Bits 31 and 30 are both 0, then case is Any) Bit 29: Auto selection Bit 28: Password Bit 27: Auto HScroll Bit 26: Auto VScroll Bit 25: HScroll bar Bit 24: VScroll bar Bit 23: Use code table Bit 22: Validate using code table Bit 21: Display only Bit 20: Empty string is NULL Bit 19: Required field Bit 18: Not used (set to 0) Bit 17: Not used (set to 0) Bit 16: Not used (set to 0) Bits 15 – 4 (3 hex digits): Always 0 for Edit edit style Bit 3: Always 0 for Edit edit style Bit 2: Always 0 for Edit edit style Bit 1: Always 0 for Edit edit style Bit 0: Always 1 for Edit edit style

Edit Mask edit style (code 90)

Table B-18 shows a sample row in the PBCatEdt table for an EditMask edit style. Table B-19 shows the meaning of the values in Table B-18.

About the example

This example shows an Edit Mask edit style using a code table of display and data values as part of a spin control. Rows 2 and beyond exist in PBCatEdt only if the edit mask is defined as a spin control (bit 29 of *Flag* is 1). Rows 3 and beyond exist only if the optional code table is populated.

For information about using an edit mask as a spin control, see Chapter 8, “Displaying and Validating Data.”

Table B-18: Sample row in PBCatEdt for an EditMask edit style

Name	Edit	Type	Cntr	Seqn	Flag	Work
MyEdit	<i>Format</i>	90	1	1	<i>Flag</i>	<i>DtFcKy</i>
MyEdit	<i>Range</i>	90	1	2	0	<i>SpinInc</i>
MyEdit	<i>Display1</i>	90	1	3	0	
MyEdit	<i>Data1</i>	90	1	4	0	
MyEdit	<i>Display2</i>	90	1	5	0	
MyEdit	<i>Data2</i>	90	1	6	0	

Table B-19: Values used in EditMask edit style sample

Value	Meaning
<i>Format</i>	Display format mask.
<i>DtFcKy</i>	Concatenated string with 1-character data-type code, 1-character focus-rectangle code (0 or 1), and 1-character accelerator key. Data type codes: Format String = "0" Format Number = "1" Format Date = "2" Format Time = "3" Format DateTime= "4" Examples: "10x" means format is Number type, focus rectangle option is unchecked, accelerator key is "x" "31z" means format is Time type, focus rectangle option is checked, accelerator key is "z"

Value	Meaning
<i>Range</i>	<p>Character representation (in decimal) of spin control range. The min value and max value are tab-delimited.</p> <p>Example:</p> <p>"1[tab]13" means min = 1, max = 13</p>
<i>SpinInc</i>	Character representation (in decimal) of spin increment.
<i>Display1</i>	Display value for first entry in code table.
<i>Data1</i>	Data value for first entry in code table.
<i>Display2</i>	Display value for second entry in code table.
<i>Data2</i>	<p>Data value for second entry in code table.</p> <p>Display and data values are repeated in pairs for each entry in the code table.</p>
<i>Flag</i>	<p>32-bit flag. Low-order four hex digits are generic edit type; high-order four are styles within the type. A 1 in any bit indicates the corresponding style is checked. A 0 in any bit indicates the corresponding style is unchecked.</p> <ul style="list-style-type: none"> Bit 31: Required Bit 30: Autoskip Bit 29: Spin control Bit 28: Read only (code table option) Bit 27: Use code table Bit 26: Not used (set to 0) Bit 25: Not used (set to 0) Bit 24: Not used (set to 0) Bit 23 – 16 (2 hex digits): Not used (set to 0) Bit 15 – 8 (2 hex digits): Always 0 for Edit Mask edit style Bit 7: Always 0 for Edit Mask edit style Bit 6: Always 0 for Edit Mask edit style Bit 5: Always 1 for Edit Mask edit style Bit 4: Always 0 for Edit Mask edit style Bits 3 – 0 (1 hex digit): Always 0 for Edit Mask edit style

Starting DataWindow Designer from the Command Line

About this chapter

This chapter describes how to use command-line arguments to start DataWindow Designer.

Contents

Topic	Page
Starting DataWindow Designer from the command line	465

Starting DataWindow Designer from the command line

You can start DataWindow Designer from a command line (or the Windows Run dialog box) and optionally open one of the following painters or tools:

- | | |
|------------------|-----------------|
| Database painter | Library painter |
| File Editor | Report painter |

To start DataWindow Designer and open a painter or tool, use the following syntax:

```
directory\dw105.exe /P paintername
```

<i>paintername</i>	<p>The name of the painter you want to open. The default is the workspace that displays when you begin a new DataWindow Designer session.</p> <p>The painter name must uniquely identify the painter. You do not have to enter the entire name. For example, you can enter <code>l</code> to open the Library painter and <code>dat</code> to open the Database painter.</p> <p>The painter name is not case sensitive. To open the file editor, you could set <i>paintername</i> to <code>FI</code> or <code>fileeditor</code>.</p>
--------------------	--

Opening an object or creating a new object

You can also add one or more of the following optional switches to the command line to open a specific object or create a new one.

`{/L libraryname} {/O objectname} {/N} {/R} {/RO} {/A arguments}`

All of these switches must follow `/P paintername`, as shown in the examples after the tables.

Table 15-15: DataWindow Designer command-line switches

Switch	Description
<code>/L</code>	Identifies the library that contains the object you want to open
<code>/O</code>	Identifies the DataWindow object that you want to open
<code>/N</code>	Creates a new DataWindow object
<code>/R</code>	Runs the DataWindow object specified with <code>/O</code> and allows designing
<code>/RO</code>	Runs the DataWindow object specified with <code>/O</code> but does not allow designing
<code>/A</code>	Provides retrieval arguments for the DataWindow object specified with <code>/O</code>

Table C-1: DataWindow Designer command-line parameters

Parameter	Description
<i>libraryname</i>	The name of the library that contains the object you want to open.
<i>objectname</i>	The name of the object you want to open.
<i>arguments</i>	For a DataWindow object, retrieval arguments for the specified DataWindow object. Arguments must be in the correct order, separated by semicolons (;). Array argument values must be separated by commas (.). Decimal arrays are not supported.

Examples

The following examples use `dd20` to represent the directory where DataWindow Designer is installed.

Enter this command to start DataWindow Designer and open the Database painter:

```
dd20\dw105.exe /P datab
```

Enter this command to start DataWindow Designer and open the DataWindow object called `d_emp_report` in the library `master.pbl`:

```
dd20\dw105.exe /P dataw /L master.pbl /O d_emp_report
```

Index

Symbols

- + operator 207
- @
 - used in crosstabs 341
 - used in validation rules 259

Numerics

- 24-hour times 237

A

- accelerator keys
 - and CheckBox edit style 244
 - and RadioButton edit style 245
- activity log 59
- Activity Log view 56
 - using 59
- Adaptive Server Anywhere
 - databases, creating and deleting 60
- Adaptive Server Enterprise, temporary tables 74
- ADO DataSet data source 139
- ADO.NET DataSet, as data source 139
- aggregate functions, in graphs 430
- alignment
 - extended attribute 66
 - in DataWindow painter 218
- AND operator, in Quick Select 117
- area graphs
 - about 422
 - making three-dimensional 424
- arguments
 - referencing retrieval 130
 - using retrieval 129
- arrays, in retrieval arguments 130
- asterisks (*)
 - displaying user input as 242

- wildcard character 39
- Auto Size setting, in graphs 443
- autoincrement columns, in DataWindow objects 194
- Autosize Height
 - bands 182
 - with nested reports 330
- average, computing 208
- axes
 - scaling 446
 - specifying line styles 448
 - specifying properties in graphs 446
 - specifying text properties 442
 - using major and minor divisions 447

B

- Background Color drop-down toolbar 154
- background.color property
 - about 297
 - specifying colors 314
- bands
 - in DataWindow painter 152
 - resizing in DataWindow painter 157
- bar graphs
 - about 422
 - making three-dimensional 424
 - specifying overlap and spacing 445
- base reports 317
- BAT files 17
- bind variables, used with nested reports 328
- BMP files
 - adding to DataWindow objects 204
 - naming in resource files 49
 - specifying column as 68
- books, online 29
- boolean expressions
 - in filters 266
 - in validation rules 258, 261
- border property 298

Index

- borders
 - in DataWindow objects 181
 - in graphs 441
 - Borders drop-down toolbar in DataWindow painter 154
 - breaks, in grouped DataWindow objects 270
 - brush.color property
 - about 299
 - specifying colors 314
 - brush.hatch property 300
 - business cards 104
 - buttons
 - adding to DataWindow objects 210
 - adding to toolbars 23
 - custom 25
 - deleting from toolbar 25
 - moving on toolbar 24
- ## C
- caching data in DataWindow objects 160
 - case
 - converting in DataWindow objects 242
 - sensitivity and code tables 253
 - categories, graph
 - basics 420
 - specifying 429
 - Category axis, graph 421
 - CheckBox edit style, defining 244
 - ClearValues function 244
 - clipboard, copying data to 167
 - closing views 14
 - code tables
 - about 251
 - defining 252
 - in Specify Retrieval Criteria dialog box 188
 - modifying at runtime 244
 - processing 253
 - using display values in crosstabs 339
 - using display values in graphs 430
 - using in drop-down lists 243
 - code, recompiling 45
 - color property
 - about 301
 - specifying colors 314
 - colors
 - changing in Database painter 59, 76
 - defining custom 16
 - for DataWindow objects 172
 - in display formats 232
 - in Select painter 123
 - column graphs
 - about 422
 - specifying overlap and spacing 445
 - columns
 - adding to DataWindow objects 201
 - appending to table 68
 - applying display formats to 228
 - applying edit styles to 241
 - defining display formats 228, 230
 - defining edit styles 240, 242
 - defining validation rules 257, 260
 - displaying as a drop-down DataWindow 249
 - displaying as check boxes 244
 - displaying as drop-down lists 243
 - displaying as radio buttons 245
 - displaying in Library painter 38
 - displaying with fixed formats 246
 - foreign key 78
 - formatting in DataWindow objects 227
 - graphing data in 425, 428
 - initial values 260
 - named in DataWindow painter Design view 154
 - presenting in DataWindow objects 238
 - preventing updates in DataWindow objects 191, 194
 - removing display formats 228
 - reordering in grid forms 166
 - resizing in forms 166
 - restricting input 246
 - selecting in Select painter 124
 - sliding to remove blank space 219
 - specifying extended attributes 66
 - specifying for crosstabs 339
 - updatable, in DataWindow objects 191, 194
 - validating input in DataWindow objects 255
 - variable length 182
 - Columns view 56
 - command line, starting from 465
 - comment extended attribute 66

- comments
 - including in SQL statements 93
 - modifying in Library painter 43
 - comments in XML export template 397
 - compiling
 - on import 48
 - regenerating library entries 45
 - Composite presentation style
 - about 317
 - limitations 319
 - using 319
 - composite reports
 - about 317
 - creating 319
 - limitations 319
 - specifying footer position 332
 - starting on new page 331
 - computed columns, including in SQL Select 124
 - computed fields
 - adding to DataWindow objects 205
 - creating from toolbar 27
 - defining 207
 - defining custom buttons for 209
 - in crosstabs 347
 - specifying display formats 230
 - summary statistics 208
 - conditional modification
 - example, gray bar 290
 - example, highlighting rows 292
 - example, rotating controls 291
 - example, size and location 294
 - modifying controls 289
 - continuous data, graphing 422
 - Control List view 157
 - control names in the DataWindow painter 180
 - Controls drop-down toolbar in DataWindow painter 154
 - controls in DataWindow objects
 - adding 201
 - aligning 218
 - copying 216
 - deleting 216
 - displaying boundaries 215
 - equalizing size 219
 - equalizing spacing 218
 - moving 216
 - resizing 217
 - selecting 156
 - count
 - computing 208
 - in graphs 430
 - Create ASA Database utility 60
 - CREATE TABLE statement 77
 - CREATE VIEW statement 83
 - Crosstab Definition dialog box 338
 - crosstabs
 - about 333
 - associating data 338
 - basic properties 345
 - changing column and row labels 346
 - changing definition of 346
 - creating 337
 - defining summary statistics 347
 - dynamic 336
 - functions 349
 - grid lines in 345
 - modifying data 346
 - previewing 344
 - property conditional expressions in 354
 - specifying columns 339
 - specifying multiple columns and rows 343
 - static 336, 353
 - using expressions 340
 - using ranges of values 350
 - CUR files
 - naming in resource files 49
 - selecting mouse pointers 174
 - currency display format 231
 - custom colors 16
 - Customize dialog box 23
- ## D
- data
 - associating with graphs in DataWindow objects 428
 - caching in DataWindow objects 160
 - changing 87, 161
 - copying to clipboard 167
 - formatting in DataWindow objects 227
 - importing 90, 163

- presenting in DataWindow objects 238
 - retrieving and updating 188
 - retrieving in DataWindow objects 159, 160
 - saving in external files 91, 168
 - saving in HTML Table format 169
 - storing in DataWindow objects 185
 - updating, controlling 191
 - validating in DataWindow objects 255
- data entry forms 102
- Data Manipulation view
- opening 86
 - printing 91
 - sorting rows 88, 89
- data source
- ADO DataSet 139
 - defining for DataWindow objects 110
 - External 136
 - modifying 183
 - Query 136
 - Quick Select 112
 - SQL Select 121
 - Stored Procedure 137
- data validation
- in code tables 254
 - with validation rules 255
- data values
- in graphs 430
 - of code tables 251
 - specifying fonts in tables 65
 - using in graphs 430
- database administration
- database access 96
 - executing SQL 92
 - painting SQL 92
 - security 96
- Database painter
- changing colors in 59
 - creating tables 62
 - defining display formats 228
 - defining validation rules 257
 - dragging and dropping 57
 - previewing data 86
 - specifying extended attributes 66
 - tasks 57
 - views 56
- working with edit styles 240
 - workspace 55
- database views
- extended attributes of 67
 - working with 81
- databases
- accessing through Quick Select 112
 - accessing through SQL Select 121
 - ADO DataSets 139
 - changing 55
 - connecting to 109
 - controlling access to 96
 - controlling updates to 191
 - creating and deleting local Adaptive Server
Anywhere 60
 - creating tables 62
 - ensuring referential integrity 75
 - executing SQL statements 95
 - importing data 90, 163
 - limiting retrieved data 265
 - logging work 59
 - retrieving, presenting, and manipulating data 86,
99
 - specifying fonts 65
 - stored procedures 137
 - system tables 74
 - updating 87, 161
 - using as data source in DataWindow object 111
- DataSet, as data source 139
- DataTable, ADO.NET, as data source 139
- datatypes
- in display formats 231
 - in graphs 446
- DataWindow objects
- about 99
 - adding controls 201
 - aligning controls 218
 - and graphs in 425
 - borders in 181
 - buttons, adding 210
 - caching data 160
 - changing margins 164
 - columns, adding 201
 - Composite presentation style 317
 - computed fields, adding 205
 - computed fields, defining 207

- controlling updates in 191
- creating new 110
- custom buttons that add computed fields 209
- data source, modifying 183
- data sources 110
- data, storing in 185
- defaults 171
- display formats 227
- drawing controls, adding 203
- edit styles 238
- escapement 222
- expressions in computed fields 207
- extended attribute information used 178
- filtering rows 265
- generating 142
- Graph presentation style 439
- graphs, adding 213
- grid style 173
- grid, working in 166
- group boxes, adding 203
- Group presentation style 272
- grouping rows 270
- initial values for columns 260
- modifying 144
- multiple column 105
- naming 144
- nesting reports 321
- newspaper columns in 176
- pictures, adding 204
- positioning of controls in 221
- presentation styles 101
- previewing 158
- previewing without retrieving data 159
- printing 165
- prompting for criteria 187
- result sets, modifying 184
- retrieval arguments, modifying 184
- retrieval criteria 187
- retrieving as needed 188
- retrieving data 159, 160
- rotating controls in 222
- saving 144
- setting colors 172
- setting timer 172
- sharing with other developers 186
- sorting rows 268
- suppressing repeating values 269, 270
- tab order 179
- text, adding 202
- TreeView presentation style 357
- units of measure 172
- validation rules 255
- years, how interpreted 236
- DataWindow painter
 - copying controls 216
 - defining validation rules 260
 - deleting controls 216
 - equalizing size 219
 - equalizing spacing 218
 - importing data 163
 - MicroHelp 157
 - modifying data 161
 - moving controls 216
 - nesting reports 317
 - printing data 165
 - resizing bands 157
 - resizing controls 217
 - retrieving data 159
 - saving data 168
 - selecting controls 156
 - sliding controls 219
 - toolbars in 154
 - undoing changes 158
 - using the Properties view 155
 - working in 150
 - working with display formats 230
 - working with edit styles 242
 - zooming 157
- DataWindow wizards, list of 101
- dates
 - display formats for 235
 - displaying in Library painter 38
- dBASE file, using as data source for DataWindow object 111
- DBMS
 - ADO DataSet 139
 - changing 55
 - controlling database access 96
 - CREATE VIEW statement 83
 - defining primary keys 77
 - executing SQL statements 95
 - exporting table syntax 72

Index

- generating SQL statement 85
- specifying an outer join 84
- stored procedures 137
- supported 54
- DDE application, using as data source for DataWindow object 111
- default layouts in views 15
- defaults, for DataWindow objects 171
- defining retrieval arguments 130
- Delete ASA Database utility 61
- Delete Library dialog box 44
- DELETE statements
 - building in Database painter 94
 - specifying WHERE clause 195
- Describe Rows dialog box
 - displaying 162
 - in Data Manipulation view 90
- detail bands
 - in DataWindow painter 153
 - resizable 182
- directories, reporting on 49
- disk space 44
- display expressions in graphs 444
- display formats
 - about 227
 - adding buttons in DataWindow painter 231
 - applying to columns 228
 - assigning from toolbar 27
 - colors in 232
 - data types 231
 - defining 231
 - deleting 263
 - for dates 235
 - for numbers 233
 - for strings 235
 - for times 237
 - in databases 66
 - in DataWindow objects 226
 - maintaining 263
 - masks 231
 - removing 228
 - sections 231
 - setting at runtime 233
 - using in graphs 444
 - working with in Database painter 228
 - working with in DataWindow painter 230
- display values
 - of code tables 251
 - using in crosstabs 339
 - using in graphs 430
- display-only fields in DataWindow objects 242
- DISTINCT keyword 122
- divisions, axis 447
- document type declaration 392
- document type declaration in XML template 392
- dragging and dropping, in Database painter 57
- drawing controls, adding to DataWindow objects 203
- drop lines, graph 448
- DROP VIEW statement 85
- drop-down toolbars 21
- DropDownDataWindow edit style
 - defining 249
 - defining code tables with 253
- DropDownDataWindow Edit Style dialog box 250
- DropDownListBox edit style
 - defining 243
 - defining code tables with 252
- DTD, about 380
- duplicate values, index 80

E

- edges, displaying in DataWindow painter 215
- Edit edit style
 - defining 242
 - defining code tables with 252
- Edit Mask edit style
 - defining 246
 - defining code tables with 253
 - spin controls 248
- Edit Style dialog box 241
- edit styles
 - about 238
 - and selection criteria 116
 - applying to columns 241
 - deleting 263
 - in databases 66
 - in DataWindow objects 226
 - in Specify Retrieval Criteria dialog box 188
 - maintaining 263

- working with in Database painter 240
- working with in DataWindow painter 242
- elevation, in 3D graphs 441
- encoding declaration 391
- Equality Required property 188
- error messages, customizing in validation rules 260
- escapement 222
- executables, running from DataWindow Designer 26
- export template
 - about 383
 - creating and saving 384
 - view 384
- Export/Import Template view
 - about 384
 - icons 384
- expressions
 - in computed fields 207
 - in crosstabs 340
 - in filters 266
 - in graphs 444
 - in validation rules 258, 261
 - specifying graph series with 431
 - specifying graph values with 430
- extended attribute system tables
 - about 73, 143, 453
 - deleting orphan table information 71
 - information used in DataWindow objects 142, 178
 - storing display formats 228
 - storing edit styles 240
 - storing extended attributes 67
 - storing validation rules 257
- Extended Attributes view 56
- extended column attributes
 - about 66
 - how stored 453
 - picture columns 68
 - used for text 178
- External data source
 - modifying result sets 184
 - updating data 192
- external data, importing 90
- external files
 - importing data from 163
 - saving data in 168

- saving table data in 91
- using as data source for DataWindow object 111

F

- file editor 17
- filters
 - in Data Manipulation view 89
 - removing 267
- focus, moving from column to column 179
- font.escapement property 302
- font.height property 303
- font.italic property 304
- font.strikethrough property 305
- font.underline property 306
- font.weight property 306
- fonts
 - in DataWindow object, changing 179
 - specifying for tables 65
- footer bands, in DataWindow painter 154
- Foreground Color drop-down toolbar in DataWindow painter 154
- foreign keys
 - about 75
 - defining 78
 - displaying in Database painter 75
 - joining tables 84, 126
 - opening related tables 76
- format property 307
- Freeform style
 - default wrap height 142
 - detail band in 153
 - of DataWindow objects 102
- Function For Toolbar dialog box 210
- functions, crosstab 349

G

- General keyword, in number display formats 233, 236
- GetFormat function 233
- GetText function, using in validation rules 261
- GetValue function 244
- GIF files, adding to DataWindow objects 204
- graphics, adding to DataWindow objects 204

- graphs
 - about 419
 - adding to DataWindow objects 213
 - adding to reports 213
 - autosizing text 443
 - changing position of 427
 - data types of axes 446
 - default positioning in DataWindow objects 221
 - default positioning in reports 221
 - defining properties 440
 - examples 432
 - expressions in 444
 - in DataWindow objects 425
 - legends in 441
 - major and minor divisions 447
 - multiple series 430
 - names of 440
 - overlays in 437
 - parts of 420
 - rotating text 443
 - scaling axes 446
 - selecting data 428
 - single series 430
 - sorting series and categories 442
 - specifying borders 441
 - specifying categories 429
 - specifying overlap and spacing of bars and columns 445
 - specifying pointers 448
 - specifying properties of axes 446
 - specifying rows 429
 - specifying series 430
 - specifying type 441
 - specifying values 430
 - text properties in 442
 - titles in 440
 - types of 421
 - using display formats 444
 - using Graph presentation style 439
 - GraphType property 441
 - grid lines, graph 448
 - Grid style
 - basic properties 173
 - detail band in 153
 - displaying grid lines 173
 - of DataWindow objects 103
 - reordering columns 166
 - resizing columns 166
 - using split horizontal scrolling 167
 - working in 166
 - grid, aligning controls in DataWindow objects 215
 - group box, adding to DataWindow objects 203
 - GROUP BY criteria 134
 - group headers, in XML 400
 - Group presentation style
 - properties of 274, 361
 - using 272
 - grouping
 - in SQL Select 134
 - restricting 135
 - groups in DataWindow objects
 - graphing 429
 - of rows 270
 - sorting 281
- ## H
- HAVING criteria 135
 - header bands, in DataWindow painter 153
 - header section in XML template 389
 - heading extended attribute 66
 - headings
 - in DataWindow objects 153
 - specifying fonts in tables 65
 - height property 307
 - Help, using 29
 - HTML Table format, saving data in 169
- ## I
- identity columns, in DataWindow objects 194
 - import template
 - about 383
 - creating and saving 384
 - defining 409
 - importing data 90, 163
 - IN operator, in Quick Select 117
 - indexes
 - creating 80

- dropping from tables 79, 80
- properties 80
- initial values, for columns 260
- initialization files
 - about 32
 - changing path 33
 - editing 17
- InkPicture control 214
- INSERT statements, building in Database painter 94
- Interactive SQL view 56

J

- Join dialog box 84
- joins, in Select painter 126
- JPEG files, adding to DataWindow objects 204

K

- key and modified columns, updating rows 195
- key and updatable columns, updating rows 195
- key columns
 - updating rows 195
- key modification, updating rows 197
- keyboard shortcuts
 - customizing 30
 - resetting 31
- keys, database
 - arrows specifying key relationship 113
 - displaying in Database painter 75
 - dropping from tables 79
 - specifying in DataWindow objects 193
 - updating values in DataWindow objects 197
 - using primary and foreign 75
- keywords, display format 232

L

- label extended attribute 66
- Label style
 - detail band in 153
 - of DataWindow objects 103
 - removing blank lines 219

- labels
 - mailing 103, 219
 - specifying fonts in tables 65
- layer attribute of graphs 427
- Layout drop-down toolbar in DataWindow painter 154
- layouts
 - restoring default in views 15
 - saving in views 14
- legends
 - in graphs 421
 - specifying text properties 442
 - using 441
- libraries
 - about 35
 - deleting 44
 - migrating 45
 - optimizing 44
 - PBL files 35
 - regenerating 45
 - reporting on 49
- library entries
 - exporting to text files 46
 - regenerating 45
 - selecting 39
- Library painter
 - about 35, 36
 - columns, displaying 38
 - dates, displaying 38
 - displaying libraries and objects 37
 - moving back, forward, and up levels 42
 - opening 36
 - pop-up menu 38
 - restricting displayed objects 39
 - setting the root 42
 - sorting 37
 - using drag and drop 38
 - views 36
 - what you can do in 35
 - workspace 36
- LIKE operator, in Quick Select 117
- line drawing controls 203
- line graphs
 - about 422
 - making three-dimensional 424
- line styles, graph 448

Index

- List view
 - about 36
 - sorting 37
 - using drag and drop 38
- local Adaptive Server Anywhere databases 60
- log files
 - about 59
 - saving 60
- logging
 - exporting table syntax 72
 - starting 59
 - stopping 60
- logical operators 116
- LookUpDisplay function 430

M

- mailing labels 103
- major divisions, in graphs 447
- managing libraries 35
- masks for display formats 231
- Match button with validation rules 259
- match patterns, validation rules 259
- metafiles, specifying columns as 68
- MicroHelp, in DataWindow painter 157
- military time 237
- minor divisions, in graphs 447
- Modify Result Set Description dialog box 184
- mouse pointers, in DataWindow objects 174
- multiple columns in DataWindow objects 105, 176
- multiple-series graphs 430

N

- Name column, sorting 37
- name tags 104
- names
 - of controls in DataWindow objects 180
 - of DataWindow objects 144
 - of graphs 440
 - of queries 147
- names, of columns in DataWindow painter Design view 154

- naming conventions
 - for DataWindow objects 144
 - for queries 147
- nested reports
 - about 317
 - adding another report 327
 - adding to report (DataWindow) 321
 - adjusting width 325
 - autosize height 330
 - changing 326
 - changing definition of 326
 - displayed in Design view 323
 - how retrieval works 319
 - limitations 319
 - slide options 331
 - specifying criteria 329
 - using retrieval arguments 323, 327
- New dialog box, accessing wizards 7
- New Name dialog box 346
- New Page command 331
- newline characters in text 178
- newspaper columns 176
- NULL values
 - allowing in code tables 252
 - allowing in tables 62
 - altering table definition 68
 - specifying display formats for 233
- numbers, display formats for 233
- N-Up style
 - computed fields in 208
 - detail band in 153
 - of DataWindow objects 105

O

- Object Details view 56
- Object Layout view 56
- objects
 - creating new 8
 - exporting syntax 46
 - exporting to text files 46
 - importing syntax 46
 - opening 8
 - previewing 9
 - regenerating 45

- running 9
 - selecting 39
 - Objects view 56
 - online books 29
 - online Help, using 29
 - Open dialog box 9
 - opening
 - Data Manipulation view 86
 - database views 82
 - Library painter 36
 - Query painter 40, 145
 - Select painter 121
 - operators, in Quick Select criteria 116
 - optimizing libraries 44
 - Options dialog box, in Library painter 39
 - OR operator, in Quick Select 117
 - ORDER BY clause
 - in SELECT statements 133
 - specifying in Quick Select 115
 - outer join, specifying 84
 - oval drawing controls 203
 - overlap, of columns in graphs 445
 - overlays, in graphs 437
- P**
- page, graphing data on 429
 - PainterBar
 - about 21
 - adding custom buttons to 25
 - controlling display of 22
 - painters
 - summary of 10
 - using views 10
 - painting SQL statements 92
 - panes
 - adding 14
 - docking 13
 - floating 13
 - in views 10
 - moving 11
 - removing 14
 - resizing 11
 - title bar, displaying and using 11
 - passwords
 - displaying as asterisks 242
 - fields 242
 - pasting
 - SQL statements in Database painter 93
 - PBCatCol system table 73, 455
 - PBCatEdt system table 73, 456
 - PBCatFmt system table 73, 456
 - PBCatTbl system table 73, 454
 - PBCatVld system table 73, 456
 - PBD files
 - creating 48
 - including resources 48
 - PBLAB105.INI 104
 - PBR files, creating 48
 - PDF, saving data as 168
 - pen.color property
 - about 308
 - specifying colors 314
 - pen.style property 308
 - pen.width property 310
 - percent display format 231
 - performance, and fragmented libraries 44
 - periodic data, in DataWindow objects 105
 - perspective, in 3D graphs 441
 - phone lists, creating 176
 - pictures
 - adding to DataWindow objects 204
 - specifying column as 68
 - pie graphs
 - about 422
 - making three-dimensional 424
 - pixels, as DataWindow object unit of measure 172
 - pixels, as report unit of measure 172
 - placeholders, in validation rules 259
 - point of view, in 3D graphs 441
 - pointer property 310
 - pointers
 - in DataWindow objects 174
 - in graphs 448
 - points, specifying size for tables 65
 - pop-up menus
 - controlling toolbars with 22
 - in Library painter 38
 - using 15
 - position, changing graph's 427

Index

- PowerBar
 - about 21
 - adding custom buttons to 25
 - controlling display of 22
 - displaying available buttons 24
 - using 6
- PowerTips
 - assigning text in custom buttons 27
 - using 6
- presentation styles
 - of DataWindow objects 101
 - using Crosstab 333
 - using Graph 439
 - using Group 272
- preview
 - for crosstabs 344
 - retrieving rows 159
- Preview view
 - in DataWindow painter 158
 - modifying data 161
- primary keys
 - about 75
 - defining 76
 - displaying in Database painter 75
 - identifying updatable rows 193
 - joining tables 84, 126
 - modifying 79
 - opening related tables 76
- Print Preview
 - about 163
 - command 163
- print specifications, reports 174
- printing
 - data, using Print Preview 163
 - DataWindow objects 165
 - in Data Manipulation painter 91
- processing instructions in XML template 398
- properties
 - about 285
 - example, gray bar 290
 - example, highlighting rows 292
 - example, rotating controls 291
 - example, size and location 294
 - modifying controls 289
 - specifying colors 314
 - using expressions 287
- Properties view
 - about 20
 - for graphs 426
 - in DataWindow painter 155
- property conditional expressions 286, 354
- property sheets
 - about 20
 - arrangement of tabs 20
 - buttons 20
 - displaying 20
- property values
 - about 295
 - background.color 297
 - border 298
 - brush.color 299
 - brush.hatch 300
 - color 301
 - font.escapement 302
 - font.height 303, 307
 - font.italic 304
 - font.strikethrough 305
 - font.underline 306
 - font.weight 306
 - format 307
 - pen.color 308
 - pen.style 308
 - pen.width 310
 - pointer 310
 - protect 311
 - specifying colors 314
 - supplying in conditional expressions 295
 - Timer.Interval 311
 - visible 311
 - width 312
 - x 312
 - x1, x2 313
 - y 313
 - y1, y2 314
- Protect property 311
- PSR files, creating 168

Q

queries

- defining 145
- modifying 147
- modifying comments 43
- naming 147
- previewing 146
- running from toolbar 27
- saving 146
- Query data source 136
- Query painter, opening 40, 145
- question marks (?) 39
- Quick Select data source
 - defining 112
 - up and down arrows 113

R

- RadioButton edit style, defining 245
- ranges, cross-tabulating 350
- Rebuild Columns At Runtime check box 353
- rectangle drawing controls 203
- referential integrity, in databases 75
- regenerating objects 45
- Report painter
 - moving controls 216
- reports
 - Composite style 107
 - Crosstab style 108
 - graphs, adding 213
 - Group style 106, 109
 - modifying comments 43
 - n-up 105
 - positioning of controls in 221
 - print specifications 174
 - PSR files 168
 - running from toolbar 27
 - units of measure 172
- resource files, creating 49
- resources
 - execution search path 49
 - naming in resource files 49
- result sets, modifying 184
- retrieval arguments
 - defining 129
 - in nested reports 323
 - modifying in DataWindow objects 184
 - referencing 130
 - specifying in WHERE clause 131
- retrieval criteria
 - in nested reports 329
 - in Quick Select grid 116
 - prompting for in DataWindow objects 187
- Retrieve command 159
- Retrieve on Preview option 159
- Retrieve Only As Needed 189
- retrieving data as needed 160
- RLE files
 - adding to DataWindow objects 204
 - naming in resource files 49
- rotation
 - in 3D graphs 441
 - of text in graphs 443
- rotation, about 222
- Round Maximum To, in graphs 447
- RoundRectangle drawing controls 203
- rows
 - allowing users to select 187
 - displaying information about 90, 162
 - filtering 89, 265
 - graphing 429
 - grouping 270
 - grouping in SQL Select 134
 - modifying in Data Manipulation painter 87
 - modifying in the Preview view 161
 - removing filters 267
 - retrieving as needed 188
 - saving in external files 91
 - sorting 88, 89, 268
 - sorting in SQL Select 133
 - suppressing repeating values 269
- Rows to Disk command 189
- rulers
 - displaying in print preview 164
 - in DataWindow painter 215
- Run/Preview dialog box 9
- runtime libraries (PBD files)
 - creating 48
 - including resources 48

- S**
- Save As dialog box 168
 - Save Rows As dialog box 91
 - saving
 - data in DataWindow objects 185
 - data in external files 168
 - data in HTML Table format 169
 - DataWindow objects 144
 - layouts in views 14
 - queries 146
 - scatter graphs 423
 - search path, of resources in resource files 49
 - Select painter
 - adding tables 123
 - colors in 123
 - defining retrieval arguments 129
 - joining tables 126
 - opening 121
 - saving work as query 121
 - selecting tables 122
 - specifying selection, sorting, and grouping criteria 130
 - specifying what is displayed 123
 - SELECT statements
 - building in Database painter 94
 - displaying 125
 - editing syntactically 125
 - for view, displaying 83
 - limiting data retrieved 265
 - predefined 145
 - saved as queries 145
 - sorting rows 268
 - selecting controls in DataWindow painter 156
 - selection criteria
 - allowing users to specify 120, 187
 - specifying in Quick Select 115
 - specifying in SQL Select 131
 - separator line in XML template 388
 - Series axis, graph 421
 - series, graph
 - as overlays 437
 - basics 420
 - specifying 430
 - SetFormat function 233
 - SetTabOrder function 180
 - setting the root 42
 - SetValue function 244
 - ShareData, in Data view 186
 - shortcuts, keyboard 30
 - Show Edges option 215
 - single-series graphs 430
 - size
 - equalizing in DataWindow painter 219
 - of bands in DataWindow painter 157
 - of controls in DataWindow objects 217
 - Slide drop-down toolbar, in DataWindow painter 154
 - sliding
 - in reports 219
 - used in nested reports 331
 - snaking columns, in DataWindow objects 176
 - sort criteria, specifying in Quick Select 115
 - sorting
 - groups 281
 - in graphs 442
 - in SQL Select 133
 - rows 268
 - sorting Name column in Library painter 37
 - source, exporting to text files 46
 - space, in libraries 44
 - spacing
 - equalizing in DataWindow painter 218
 - of columns in graphs 445
 - Specify Sort Columns dialog box 268
 - Specify Update Properties dialog box 192
 - spin controls
 - defining edit masks as 248
 - SQL Select
 - adding tables 123
 - data source, colors in 123
 - defining retrieval arguments 129
 - joining tables with 126
 - selecting columns 124
 - selecting tables 122
 - specifying selection, sorting, and grouping criteria 130
 - specifying what is displayed 123
 - using as data source 121
 - SQL Statement Type dialog box 93
 - SQL statements
 - building and executing 92
 - displaying 125
 - executing 92, 95

- exporting to another DBMS 72
- for views, displaying 83
- generating through Quick Select 112
- generating through SQL Select 121
- importing from text files 95
- logging 59
- painting 92
- typing 94
- SQLCache variable 328
- stacked graphs 424
- standalone document declaration 391
- starting InfoMaker from command line 465
- Stored Procedure data source 137
- stored procedures
 - modifying result sets in DataWindow objects 184
 - updating data in DataWindow objects 192
 - updating data in forms 192
 - using 137
- string
 - concatenating 207
 - display formats for 235
- style, of DataWindow objects 171
- StyleBar
 - about 21
 - controlling display of 22
 - in DataWindow painter 155
- sum
 - computing 208
 - in graphs 430
- summary bands, in DataWindow painter 154
- summary statistics
 - computing 208
 - in crosstabs 347
- syntax
 - exporting to another DBMS 72
 - of view SELECT statement 83
- system options tab 33
- system tables
 - DBMS 74
 - extended attribute 67, 73, 453

T

- tab order in DataWindow objects 179
- tables
 - altering definition of 68, 69
 - applying display formats to columns 228
 - applying edit styles to columns 241
 - controlling updates to 191
 - creating 62
 - creating indexes 80
 - dropping 70
 - dropping indexes 79, 80
 - exporting syntax to another DBMS 72
 - extended attributes, specifying 66
 - fonts 65
 - joining in Select painter 126
 - opening, related to foreign keys 76
 - opening, related to primary keys 76
 - presenting in Freeform style 102
 - presenting in Grid style 103
 - presenting in Label style 103
 - presenting in N-Up style 105
 - presenting in Tabular style 102
 - printing data 91
 - removing from Database painter view 70
 - saving data in external files 91
 - selecting for SQL Select 112, 121
 - specifying extended attributes 67
 - specifying fonts 65
 - specifying updatable 193
 - temporary 74
 - working with data 86
- tab-separated files, using as data source for DataWindow
 - object 111
- Tabular style
 - detail band in 153
 - of DataWindow objects and reports 102
- temporary tables, ASE 74
- text
 - cutting, copying, and pasting 70
 - editing 17
 - in DataWindow objects 178, 202
 - inserting newline characters 178
 - on toolbar buttons 22
 - rotating in graphs 443

Index

- text files
 - exporting objects to 46
 - importing SQL statements from 95
 - text patterns, matching in validation rules 259
 - text properties
 - in DataWindow objects 179
 - in graphs 442
 - three-dimensional graphs
 - about 424
 - point of view 441
 - Time keyword 238
 - timer, setting in DataWindow objects 172
 - Timer_Interval property 311
 - times, display formats for 237
 - timestamps, used in updating rows 195
 - title bars, displaying in views 11
 - titles
 - of graphs 421, 440
 - specifying text properties 442
 - To-Do List 18
 - Toolbar Item Command dialog box 26
 - toolbars
 - about 21
 - controlling display of 22
 - custom buttons 25
 - customizing 23
 - docking 23
 - drop down 21
 - in DataWindow painter 154
 - moving 23
 - moving buttons 24
 - resetting 25
 - Toolbars dialog box 22
 - tracing XML import 416
 - Trail Footer command 332
 - Tree view
 - about 36
 - expanding and collapsing 37
 - using drag and drop 38
 - TreeView DataWindow
 - adding levels 363
 - creating 359
 - Design view 366
 - icons in the Design view 366
 - properties 367
 - tree node icons 367
 - TreeView presentation style 357
- ## U
- Undo command
 - in Database painter 70
 - in DataWindow painter 158
 - unique indexes
 - creating 80
 - defining for primary key 77
 - unique keys, specifying for DataWindow 193
 - units of measure, specifying for DataWindow objects 172
 - units of measure, specifying for reports 172
 - up and down arrows, in Quick Select 113
 - updatable columns in DataWindow object 194
 - Update function 162
 - UPDATE statements
 - building in Database painter 94
 - specifying WHERE clause 195
 - updates, in DataWindow objects 191
- ## V
- validation rules
 - about 66, 255
 - customizing error messages 260
 - defining in Database painter 257
 - defining in DataWindow painter 260
 - deleting 263
 - maintaining 263
 - Value axis, graph 421
 - values
 - ensuring validity of 75
 - specifying for graphs 430
 - suppressing repeating 269
 - variables
 - in retrieval arguments 131
 - SQLCache 328
 - View Definition dialog box 84
 - View painter, opening 82
 - views
 - adding 14
 - closing 14

- docking 13
- dropping 85
- floating 13
- in Library painter 36
- moving 11
- removing 14
- resizing 11
- restoring layouts 15
- saving layouts 14
- title bar, displaying and using 11
- updating 191
- using in painters 10

Visible property 311

W

WHERE clause

- specified for update and delete 195
- specifying in Quick Select 115
- user modifying at runtime 187

WHERE criteria 131

width property 312

wildcards, in Library painter 39

WMF files

- adding to DataWindow objects 204
- naming in resource files 49

workspace

- in Database painter 55
- in Library painter 36

wrap height, default in freeform reports 142

X

x property 312

x1, x2 property 313

XML

- about 379
- associating a namespace with a schema 406
- Detail Start element 388
- exporting 398
- exporting metadata 404
- header and detail sections 387
- importing 408
- importing group headers 410

- importing with a template 409
- importing without a template 413
- Iterate Header for Groups 400
- Iterate Header for Groups and Detail Start element 388
- parsing 382
- syntax 381
- templates, about 384
- templates, creating 385
- templates, editing 390
- templates, saving 387
- tracing import 416
- valid and well-formed 380

XML declaration in XML export template 391

XML Schema 381

Y

y property 313

y1, y2 property 314

years in DataWindow objects, specified with two digits 235

Z

zero display format 233

Zoom command, in print preview 164

