SYBASE®

DataWindow Designer User’s Guide

DataWindow .NET™

2.5

DOCUMENT ID: DC00043-01-0250-01

LAST REVISED: August 2007

Contents

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 the DataWindow Designer plug-in for Visual Studio 2005 or the DataWindow Designer standalone tool 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 the DataWindow Designer plug-in to create DataWindow objects.

Related documents

For information on using DataWindow objects in Visual Studio 2005, see the DataWindow .NET *Programmer's Guide*.

For a complete list of books and online documentation, see the preface in the DataWindow .NET *Programmer's Guide*.

Other sources of information

Use the Sybase Getting Started CD, the SyBooks CD, and the Sybase Product Manuals Web site to learn more about your product:

- 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.

  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.

  Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.

- 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

DataWindow Designer

This part introduces you to the DataWindow Designer plug-in for Visual Studio 2005 and describes how to work with it. It also describes how to work with databases, tables, views, and extended attributes.

CHAPTER 1

Working with DataWindow Designer

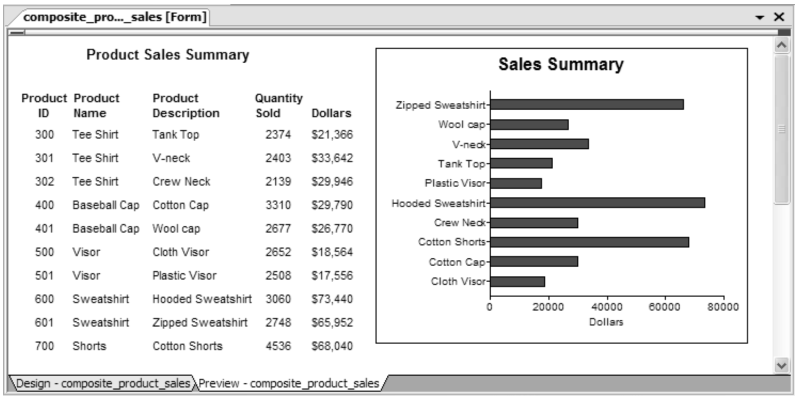
About this chapter

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

Contents

About DataWindow Designer

DataWindow Designer is a plug-in for Visual Studio 2005 that creates DataWindow objects for use in DataWindow .NET 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, shown in the Preview view in the DataWindow painter, retrieves and displays product sales.



Product Sales Summary

Product ID	Product Name	Product Description	Quantity Sold	Dollars
300	Tee Shirt	Tank Top	2374	$21,366
301	Tee Shirt	V-neck	2403	$33,642
302	Tee Shirt	Crew Neck	2139	$29,946
400	Baseball Cap	Cotton Cap	3310	$29,790
401	Baseball Cap	Wool cap	2677	$26,770
500	Visor	Cloth Visor	2652	$18,564
501	Visor	Plastic Visor	2508	$17,556
600	Sweatshirt	Hooded Sweatshirt	3060	$73,440
601	Sweatshirt	Zipped Sweatshirt	2748	$65,952
700	Shorts	Cotton Shorts	4536	$68,040

Standalone DataWindow Designer
DataWindow Designer is also installed as a standalone tool with its own development environment. This book and the other books in the DataWindow .NET 2.5 collection focus on how to use the plug-in. Most tasks are performed in a similar way in the plug-in and the standalone tool, but some procedures are different. If you need help performing tasks in the standalone tool, see the DataWindow .NET 2.0 documentation on the Sybase Product Manuals Web site at http://infocenter.sybase.com/help/index.jsp?topic=/com.sybase.help.dwnet_2.0.dwbug/html/dwbug/title.htm.

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

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

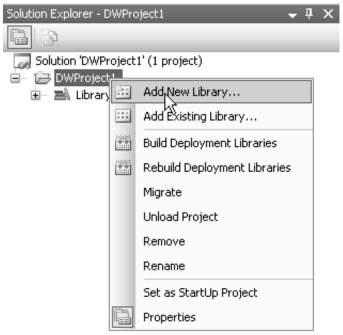
Painters

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 and SQL statements.

DataWindow Designer projects

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 *.dwproj* (in the plug-in) or *.dwp* (in the standalone tool). The two project file types are both text files but their formats are different.The DataWindow objects you create are stored in binary files called libraries. Library files have the extension *.pbl*.

To create a new DataWindow project and a library to hold your DataWindow objects, select File>New>Project in Visual Studio 2005 and select DataWindow Projects from the list of project types. To create a library with a different name, right-click the new project in the Solution Explorer and select Add New Library.



If you have worked with DataWindow Designer before, you can also add existing projects to your solution and existing libraries to your project. You must migrate both projects and libraries to the new version before attempting to open a DataWindow object.

To open a project created in the standalone DataWindow Designer tool in the plug-in, first back up all your PBLs. In Visual Studio, select File>Open>Project. In the Open Project dialog box, navigate to the location of the project built in the standalone tool, select DataWindow Project Files from the Files of type list, and select your project. A message box displays asking you to confirm that you want to migrate a project in the old format to the new format. When you click Yes, a new project with the extension *.dwproj* is created and opened in the Solution Explorer and any PBLs in the project are migrated to the new version.

Working with DataWindow objects

In DataWindow Designer, you can:

- Create new DataWindow objects
- Open existing DataWindow objects
- Preview DataWindow objects

After you create or open a DataWindow object, it displays in the Design view in the DataWindow painter.

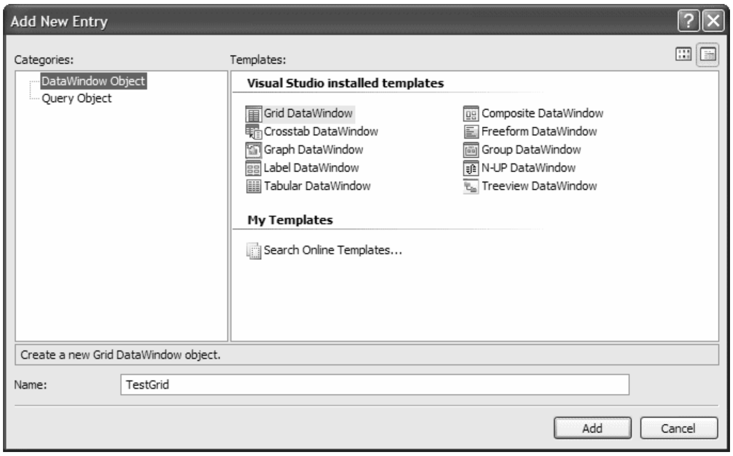
❖ **To create a new DataWindow object:**

1. If the data you want to use to create the DataWindow is in a database, make sure you are connected to it.

   To connect to a database you use a database profile, which you can create in the Database painter. Select View>Database Painter to open the Database painter. For more information about creating profiles, see *Connecting to Your Database*.

2. Right-click the library where you want to save the DataWindow object in the Solution Explorer and select Add New Entry.

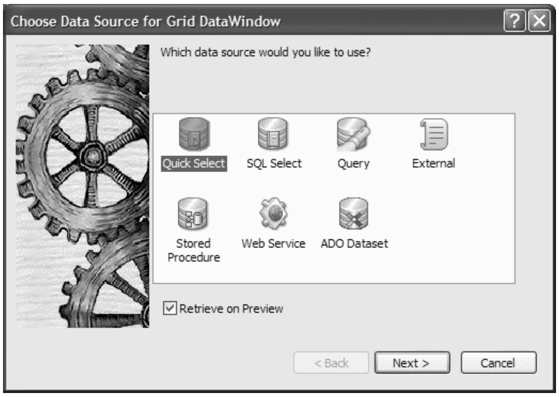
3 In the Add New Entry dialog box, select DataWindow Object from the categories list, select a DataWindow style from the Templates list, provide a name for the DataWindow object, and click Add.



Create a new Grid DataWindow object.

Name: TestGrid

4 In the DataWindow wizard, select a data source.



Which data source would you like to use?

☑ Retrieve on Preview

5 Complete the rest of the wizard to create the DataWindow object.

The remaining steps in the DataWindow wizard depend on the DataWindow style and data source you selected. For more information about choosing a style and data source and using the wizard, see Chapter 3, “Defining DataWindow Objects.”

❖ **To open an existing DataWindow object:**

- Double-click the DataWindow object in the Solution Explorer.

❖ **To preview a DataWindow object:**

- Click the Preview tab in the DataWindow painter.

  The Preview view displays as a tabbed view in the DataWindow painter when you first open the painter. If you do not see the Preview view, select View>DataWindow Painter Layout>Default Layout.

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
Query painter	Graphically define and save SQL SELECT statements for reuse with DataWindow objects
SQL Select painter	Graphically define SQL SELECT statements for DataWindow objects

Using views in painters

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. 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.

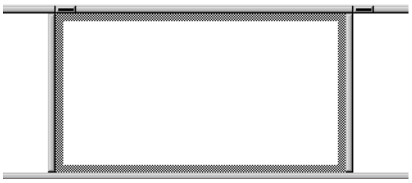
Moving and resizing panes and views

To move a pane, 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, drag the tab for the view you want to move.

❖ **To move a pane:**

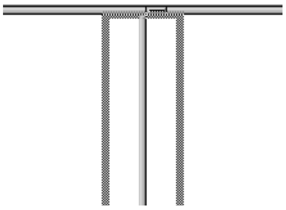
1 Place the pointer anywhere on the title bar of the view at the top of the stack, hold down the left mouse button, and start moving the pane.

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

To move a pane here	Drop the outline here
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 Visual Studio window.

❖ **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.

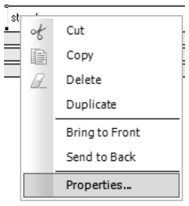
Using pop-up menus

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 window

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.

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



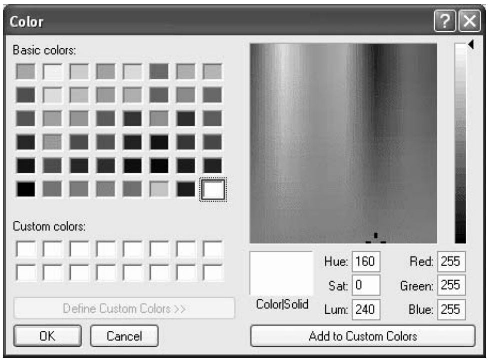
Defining colors

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:



Hue: 160 Sat: 0 Lum: 240 Red: 255 Green: 255 Blue: 255

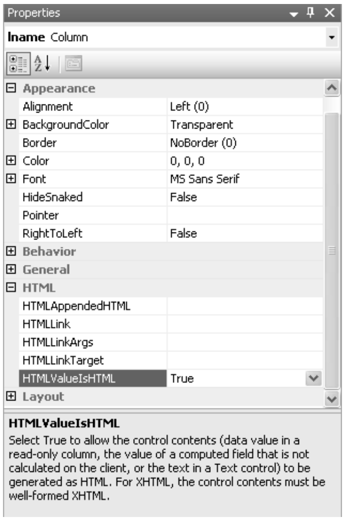
2 Define your custom colors:

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

Area of the Color dialog box	What you do
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

Setting properties

In DataWindow Designer, you set properties for DataWindow and Query objects in the Visual Studio Properties window. For example, for a column in a DataWindow object, you can set several different kinds of properties (in the categories appearance, behavior, general, HTML, and layout) by expanding categories in the Properties window.



Properties

lname Column

⊟ **Appearance**	
Alignment	Left (0)
⊞ BackgroundColor	Transparent
Border	NoBorder (0)
⊞ Color	0, 0, 0
⊞ Font	MS Sans Serif
HideSnaked	False
Pointer	
RightToLeft	False
⊞ **Behavior**	
⊞ **General**	
⊟ **HTML**	
HTMLAppendedHTML	
HTMLLink	
HTMLLinkArgs	
HTMLLinkTarget	
HTMLValueIsHTML	True
⊞ **Layout**	

HTMLValueIsHTML
Select True to allow the control contents (data value in a read-only column, the value of a computed field that is not calculated on the client, or the text in a Text control) to be generated as HTML. For XHTML, the control contents must be well-formed XHTML.

Brief help for the property displays in the Properties window. For more detailed help, see the *DataWindow Object Reference*.

The Properties window 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.

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 1-2.

Table 1-2: Accessing Help

Approach	What it does
From the Windows Start menu, select Program Files>Sybase>DataWindow Designer 2.5>Help Files	Opens the Windows Help.
From the Windows Start menu, select Program Files>Sybase>DataWindow Designer 2.5>Compiled HTML Help File	Opens the compiled HTML Help.
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.
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.

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 xiii.

How DataWindow Designer preferences are 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/2.5*.

Installation-related information is stored in *HKEY_LOCAL_MACHINE/Software/Sybase/DataWindowDesigner/2.5*.

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 *Sybase\DataWindow Designer 2.5* directory.

CHAPTER 2 Managing the Database

About this chapter

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

Contents

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

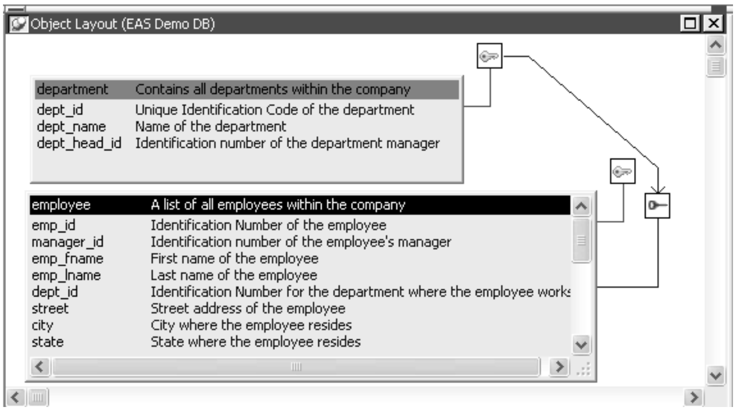
Relational databases use keys to ensure database integrity.

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:



department	Contains all departments within the company
dept_id	Unique Identification Code of the department
dept_name	Name of the department
dept_head_id	Identification number of the department manager

employee	A list of all employees within the company
emp_id	Identification Number of the employee
manager_id	Identification number of the employee's manager
emp_fname	First name of the employee
emp_lname	Last name of the employee
dept_id	Identification Number for the department where the employee works
street	Street address of the employee
city	City where the employee resides
state	State where the employee resides

For more information, see "Working with keys" on page 39.

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 44.

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 46.

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 31. 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 may 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
Triggers
Events
Web services

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 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.

You can drag most items in these folders to the Object Details view to display their properties. You can also drag procedures, functions, triggers, and events to the ISQL view.

Trigger information is listed for Adaptive Server Enterprise and SQL Anywhere tables. A trigger is a special form of stored procedure that is associated with a specific database table. Triggers fire automatically whenever someone inserts, updates or deletes rows of the associated table. Triggers can call procedures and fire other triggers, but they have no parameters and cannot be invoked by a CALL statement. You use triggers when referential integrity and other declarative constraints are insufficient.

Events can be used in a SQL Anywhere database to automate database administration tasks, such as sending a message when disk space is low. Event handlers are activated when a provided trigger condition is met. If any events are defined for a SQL Anywhere connection, they display in the Events folder for the connection in the Objects view.

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

Using the Database painter, you can do the following in any DBMS to which you have been given access by the database administrator:

- Modify local table and column properties
- Retrieve, change, and insert data
- Create new local tables or modify existing tables

Setting the database connection

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.

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.

Changing the database connection

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.

For more about changing the database you are connected to, see *Connecting to Your Database*.

Creating and deleting databases

When you are connected to SQL Anywhere, you can create a new database or delete an existing database using the Database painter.

For all other DBMSs, creating and deleting a database is an administrative task that you cannot do within DataWindow Designer.

Using the Database painter

To open the Database painter, select View>Database Painter.

Views in the Database painter

Table 2-1 lists the views available in the Database painter.

Table 2-1: 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.
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.
Object Details	Displays an object's properties. For some objects, its properties are read-only; for others, properties can be modified.
Object Layout	Displays a graphical representation of tables and their relationships.
Objects	Lists 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, events, triggers, and utilities (the database components listed depend on the database and your user privileges).
Results	Displays data in a grid, table, or freeform format.

Adding a view

If a view that you want to use does not display in the painter, you can add the view. Select View>Database Painter Layout and select the view that you want to add.

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. Position the pointer on the database object's icon and drag it to the appropriate view.

Table 2-2: 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, event trigger	Object Details view
Table or view	Object Layout view
Table or column	Columns view
Procedure or view	ISQL view

Database painter tasks

Table 2-3 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 menu selections from the menu bar or from pop-up menus.

Table 2-3: 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. You can use the Import and Export Profiles menu selections to copy profiles. For more information, see the section on importing and exporting database profiles in *Connecting to Your Database*.
Connect to a database	Highlight a database profile and then select Connect from the pop-up menu.
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.
Build, execute or explain SQL	Use the ISQL view to build SQL statements. Use the Paste Special>SQL pop-up menu 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.

To	Do this
Define or modify extended attributes	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 2-4: 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.
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 save the table, 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.

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 57.

Creating and deleting a SQL 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 SQL Anywhere database from within DataWindow Designer.

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

❖ **To create a local SQL 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 SQL 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 29.

   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 Syntax from the menu bar or select Pending Syntax 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. Select Save Table from the 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 31.

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 30.

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 2-5: 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.

Field	What you enter
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

Select this tab	To modify this property
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.

Select this tab	To modify these extended attributes
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 7, “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.

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. Select Cut Column or Copy Column from the pop-up menu.

❖ **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. Select Paste Column from the pop-up menu.

❖ **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. Select Paste Column from the pop-up menu.

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:

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

Copying and pending SQL changes

- When you are viewing pending SQL changes, you can copy pending changes to the clipboard.

❖ **To copy the SQL syntax to the clipboard:**

- In the Pending Syntax view, select Select All and then Copy from the pop-up menu.

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 Definition from the pop-up menu.

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 the syntax of an existing table to a log:**

1. Select the table in the Objects or Object Layout view.
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 25.

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_lname. 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 2-6 lists the extended attribute system tables. For more information, see Appendix B, “The Extended Attribute System Tables.”

Table 2-6: 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 temporary tables

You can create and edit temporary tables in the Database painter, SQL Select painter, or DataWindow painter when you use the ASE or SYC native driver to connect to an Adaptive Server database, or the SNC native driver to connect to a Microsoft SQL Server 2005 database. Temporary tables persist for the duration of a database connection, residing in a special database called “tempdb”.

Creating temporary tables

You add a temporary table to the tempdb database by right-clicking the Temporary Tables icon in the Objects view and selecting New. The table is designated as a temporary table by assigning a name that starts with the # character. When you save the table, the Create New Temporary Table dialog box displays. The # character is added automatically.

If there is no Temporary Tables icon in the Objects view, right-click the Tables icon and select New. Assign a table name prefaced with the # character.

For SNC, use # for a local temporary table or ## for a global temporary table. Temporary tables must start with the # character. Local temporary tables are visible only in the user’s current connection and are deleted when the user disconnects. Global temporary tables are visible to any user connected to the instance of SQL Server, and they are deleted when all users referencing the table disconnect.

Working with temporary tables

After you create a temporary table, you can create indexes and a primary key for the table from the pop-up menu for the table in the Object Layout view. If you define a unique index or primary key, you can perform insert, update, and delete operations in DataWindow objects.

Selecting Edit Data from the pop-up menu of a temporary table retrieves data that you store in that table. You can also select Drop Table, Add to Layout, Export Syntax, and properties from the pop-up menu in the Objects view.

Working with keys

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

Why you should use keys

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.

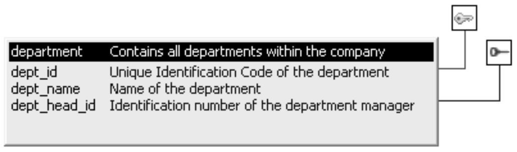
Viewing keys

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)



department	Contains all departments within the company
dept_id	Unique Identification Code of the department
dept_name	Name of the department
dept_head_id	Identification number of the department manager

If you cannot see the lines
If the color of your window background makes it difficult 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 24.

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 Database 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 select New>Foreign Key from the pop-up menu.
- 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 select Properties from the 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 SQL Anywhere databases
In SQL 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 select New>Index from the pop-up menu.
   - Select Object>Insert>Index from the main 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 SQL 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 select Properties from the pop-up menu.
   - Select Properties from the Object 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 SQL Select painter. For more information about the SQL Select painter, see "Selecting a data source" on page 75.

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. Select 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 SQL Select painter:

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 48.

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 75.

6. When you have completed the view, click the OK 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 SQL Select painter or want to look at the definition of an existing view.

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

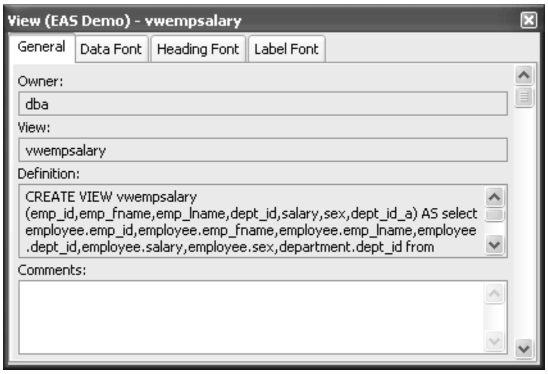
- Select the Syntax tab in SQL Select painter.

  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:



Owner: dba

View: vwempsalary

Definition:

```
CREATE VIEW vwempsalary
(emp_id,emp_fname,emp_lname,dept_id,salary,sex,dept_id_a) AS select
employee.emp_id,employee.emp_fname,employee.emp_lname,employee
.dept_id,employee.salary,employee.sex,department.dept_id from
```

Comments:

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 SQL Select 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:

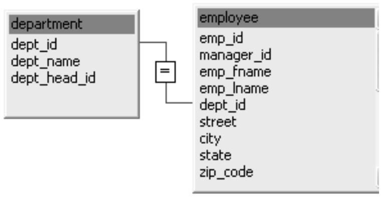
- If there is a primary/foreign key relationship between the tables, DataWindow Designer automatically joins them.
- If there are no keys, DataWindow Designer tries to join tables based on common column names and types.

❖ **To join tables:**

1 Click the Join button.

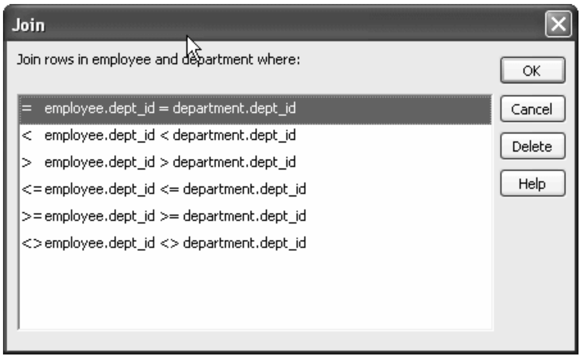
2 Click the columns on which you want to join the tables.

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:



Join rows in employee and department where:

```
=  employee.dept_id = department.dept_id
<  employee.dept_id < department.dept_id
>  employee.dept_id > department.dept_id
<= employee.dept_id <= department.dept_id
>= employee.dept_id >= department.dept_id
<> employee.dept_id <> department.dept_id
```

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 92.

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. 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.

Exporting view syntax

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

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

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

   For more information about the log, see "Logging your work" on page 25.

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

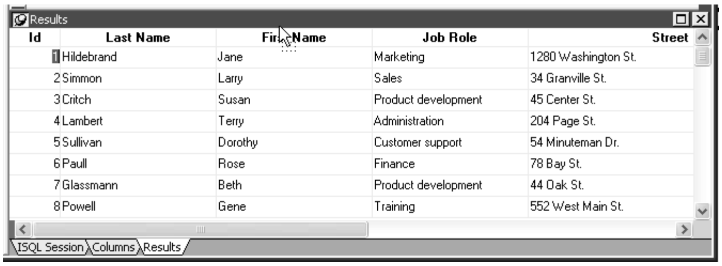
❖ **To retrieve data:**

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 or the pop-up menu 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, select Insert Row from the pop-up menu and enter data in the new row.
- To delete a row, select Delete Row from the pop-up menu.

  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 Select Save from the pop-up menu to apply changes to the database.

Sorting rows

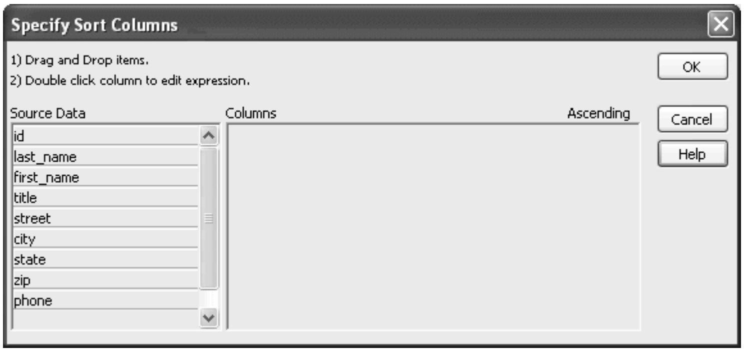
You can sort the data, but any sort criteria you define are for testing only and are not saved with the table or passed to the DataWindow painter.

❖ **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`.

5 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.

6 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.

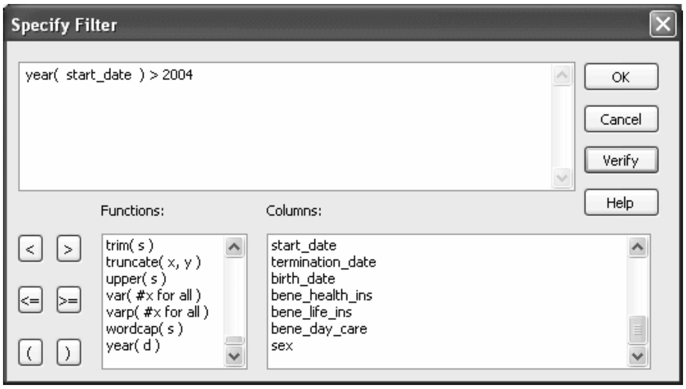
The filters you define are for testing only and are not saved with the table or passed to the DataWindow painter.

❖ **To filter the rows:**

1 Select Rows>Filter from the menu bar.

 The Specify Filter dialog box displays.

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



```
year(  start_date  ) > 2004
```

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

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

Saving data

You can save the displayed data in an external file.

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

1 Select Save Rows As from the pop-up menu.

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, PDF, 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 133.

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 create 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 Database 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. Select Paste Special>SQL from the pop-up menu, then the statement type (Select, Insert, Update, or Delete).

   The Select Table(s) dialog box displays.

2. 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 SQL Select painter, see "Selecting a data source" on page 75.

3. Do one of the following:

   - For a SELECT statement, define the statement exactly as in the SQL 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 46.

   - 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.

4. When you have finished creating the SQL statement, click OK.

   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. This includes 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. However, you might want to use a SET statement to change a default setting in the development environment, such as SET NOCOUNT ON or SET ANSI_WARNINGS OFF. You can enable SET commands in the ISQL view for database interfaces that support them by adding the following line to the [Database] section in your *DW.INI* file:

```
EnableSet=1
```

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 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:**

- Select Execute... from the pop-up menu.

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 50.

Customizing the editor

The Interactive SQL view 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. You can set Script and Font properties in the Script and Editor Font tag pages.

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.

PART 2

DataWindows

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

C H A P T E R 3

Defining DataWindow Objects

About this chapter

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

Contents

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 Web service).

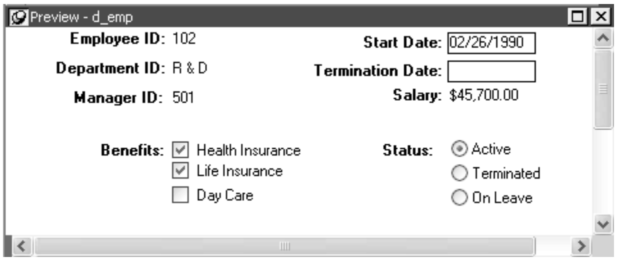
DataWindow objects have knowledge about the data they are retrieving. You can specify display formats, presentation styles, and other data properties so that users can make the most meaningful use of the data.

DataWindow object examples

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

Edit styles

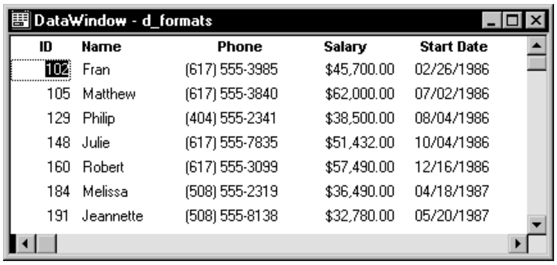
If a column can take only a small number of values, you can have the data appear as radio buttons in a DataWindow object so that users know what their choices are.



Employee ID: 102
Department ID: R & D
Manager ID: 501
Start Date: 02/26/1990
Termination Date:
Salary: $45,700.00
Benefits: [✓ Health Insurance] [✓ Life Insurance] Day Care
Status: [✓ Active] Terminated On Leave

Display formats

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



ID	Name	Phone	Salary	Start Date
102	Fran	(617) 555-3985	$45,700.00	02/26/1986
105	Matthew	(617) 555-3840	$62,000.00	07/02/1986
129	Philip	(404) 555-2341	$38,500.00	08/04/1986
148	Julie	(617) 555-7835	$51,432.00	10/04/1986
160	Robert	(617) 555-3099	$57,490.00	12/16/1986
184	Melissa	(508) 555-2319	$36,490.00	04/18/1987
191	Jeannette	(508) 555-8138	$32,780.00	05/20/1987

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 5, “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.

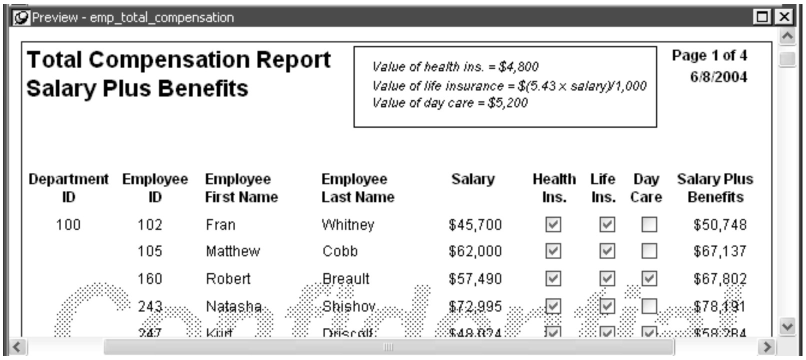
When you create a DataWindow object, you can choose from the presentation styles listed in the following table.

Table 3-1: 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:



Total Compensation Report
Salary Plus Benefits

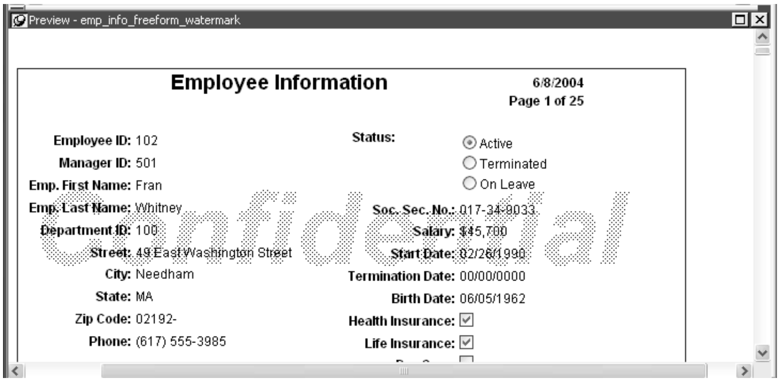
Value of health ins. = $4,800
Value of life insurance = $(5.43 x salary)/1,000
Value of day care = $5,200

Page 1 of 4
6/8/2004

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	✓	✓	☐	$50,748
	105	Matthew	Cobb	$62,000	✓	✓	☐	$67,137
	160	Robert	Breault	$57,490	✓	✓	✓	$67,802
	243	Natasha	Shishov	$72,995	✓	✓	☐	$78,191
	247	Kurt	Driscoll	$48,024	✓	✓	✓	$58,284

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.



Employee Information

6/8/2004
Page 1 of 25

Employee ID:	102	**Status:**	[✓ Active] Terminated On Leave
Manager ID:	501		
Emp. First Name:	Fran		
Emp. Last Name:	Whitney	**Soc. Sec. No.:**	017-34-8033
Department ID:	100	**Salary:**	$45,700
Street:	49 East Washington Street	**Start Date:**	02/26/1990
City:	Needham	**Termination Date:**	00/00/0000
State:	MA	**Birth Date:**	06/05/1962
Zip Code:	02192-	**Health Insurance:**	✓
Phone:	(617) 555-3985	**Life Insurance:**	✓

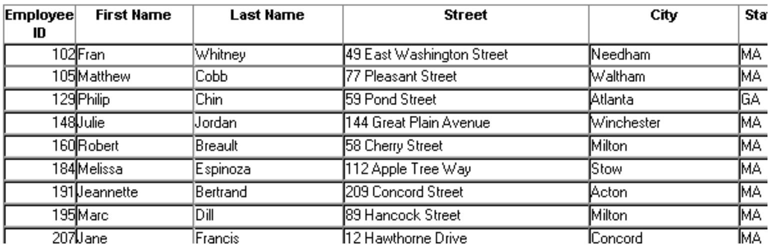
Using the Grid style

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

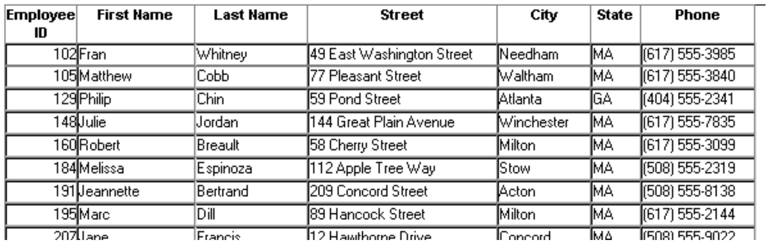
This grid report shows employee information. Several of the columns have a large amount of extra white space:



Employee ID	First Name	Last Name	Street	City	Sta
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	Jeannette	Bertrand	209 Concord Street	Acton	MA
195	Marc	Dill	89 Hancock Street	Milton	MA
207	Jane	Francis	12 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:

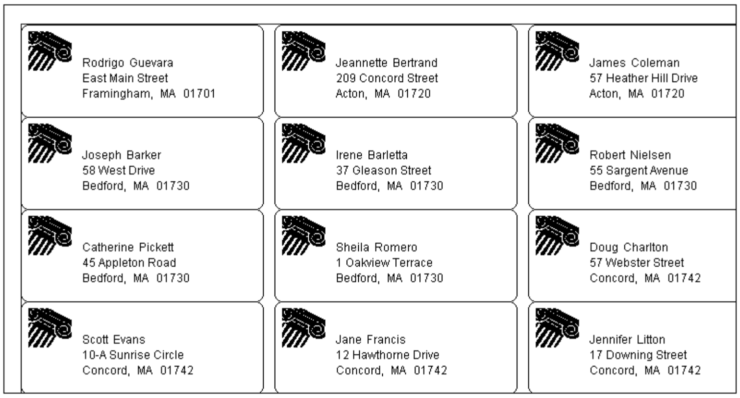


Employee ID	First Name	Last Name	Street	City	State	Phone
102	Fran	Whitney	49 East Washington Street	Needham	MA	(617) 555-3985
105	Matthew	Cobb	77 Pleasant Street	Waltham	MA	(617) 555-3840
129	Philip	Chin	59 Pond Street	Atlanta	GA	(404) 555-2341
148	Julie	Jordan	144 Great Plain Avenue	Winchester	MA	(617) 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	Jeannette	Bertrand	209 Concord Street	Acton	MA	(508) 555-8138
195	Marc	Dill	89 Hancock Street	Milton	MA	(617) 555-2144
207	Jane	Francis	12 Hawthorne Drive	Concord	MA	(508) 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

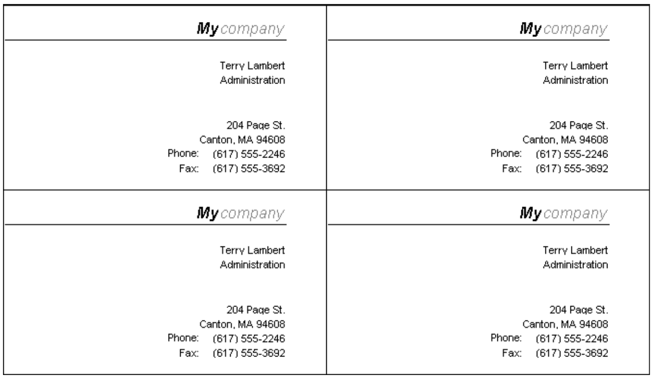
Doug Charlton
57 Webster Street
Concord, MA 01742

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



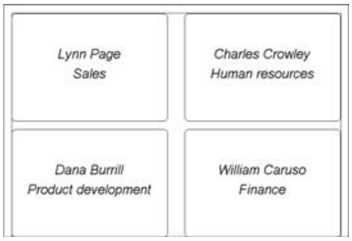
My company
Terry Lambert
Administration
204 Page St.
Canton, MA 94608
Phone: (617) 555-2246
Fax: (617) 555-3692

My company
Terry Lambert
Administration
204 Page St.
Canton, MA 94608
Phone: (617) 555-2246
Fax: (617) 555-3692

My company
Terry Lambert
Administration
204 Page St.
Canton, MA 94608
Phone: (617) 555-2246
Fax: (617) 555-3692

My company
Terry Lambert
Administration
204 Page St.
Canton, MA 94608
Phone: (617) 555-2246
Fax: (617) 555-3692

Name tags



Lynn Page
Sales

Charles Crowley
Human resources

Dana Burrill
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 *pblab110.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.

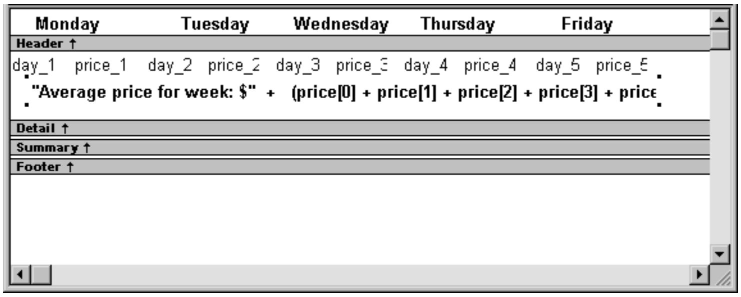
After you select a data source, you are asked how many rows to display across the page.

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
Detail ↑
Summary ↑
Footer ↑

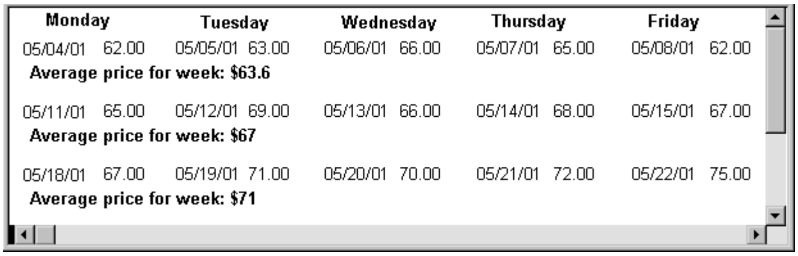
"Average price for week: $" + (price[0] + price[1] + price[2] + price[3] + price

---

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 4, "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									

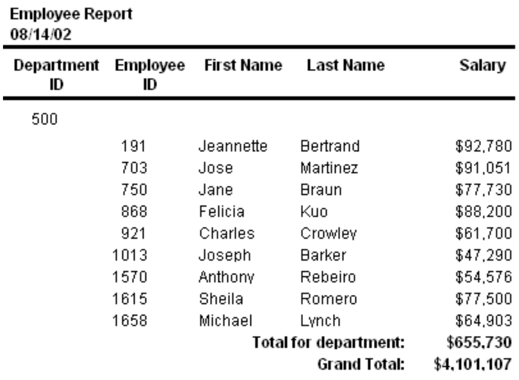
Another way to get multiple-column DataWindow objects
In an n-up DataWindow object, the data is displayed across and then down. If you want your data to go down the page and then across in multiple columns, as in a phone list, you should create a standard tabular DataWindow object, then specify newspaper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

For more information on newspaper columns, see Chapter 4, “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

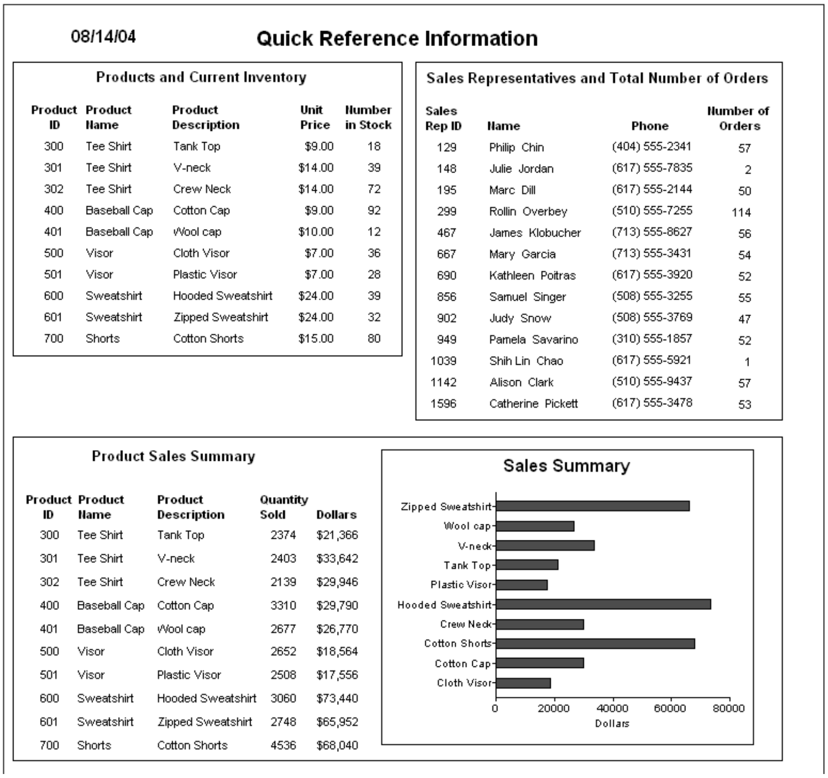
Department ID	Employee ID	First Name	Last Name	Salary
500				
	191	Jeannette	Bertrand	$92,780
	703	Jose	Martinez	$91,051
	750	Jane	Braun	$77,730
	868	Felicia	Kuo	$88,200
	921	Charles	Crowley	$61,700
	1013	Joseph	Barker	$47,290
	1570	Anthony	Rebeiro	$54,576
	1615	Sheila	Romero	$77,500
	1658	Michael	Lynch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

For more about the Group presentation style, see Chapter 8, "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:



08/14/04 **Quick Reference Information**

Products and Current Inventory

Product ID	Product Name	Product Description	Unit Price	Number in Stock
300	Tee Shirt	Tank Top	$9.00	18
301	Tee Shirt	V-neck	$14.00	39
302	Tee Shirt	Crew Neck	$14.00	72
400	Baseball Cap	Cotton Cap	$9.00	92
401	Baseball Cap	Wool cap	$10.00	12
500	Visor	Cloth Visor	$7.00	36
501	Visor	Plastic Visor	$7.00	28
600	Sweatshirt	Hooded Sweatshirt	$24.00	39
601	Sweatshirt	Zipped Sweatshirt	$24.00	32
700	Shorts	Cotton Shorts	$15.00	80

Sales Representatives and Total Number of Orders

Sales Rep ID	Name	Phone	Number of Orders
129	Philip Chin	(404) 555-2341	57
148	Julie Jordan	(617) 555-7835	2
195	Marc Dill	(617) 555-2144	50
299	Rollin Overbey	(510) 555-7255	114
467	James Klobucher	(713) 555-8627	56
667	Mary Garcia	(713) 555-3431	54
690	Kathleen Poitras	(617) 555-3920	52
856	Samuel Singer	(508) 555-3255	55
902	Judy Snow	(508) 555-3769	47
949	Pamela Savarino	(310) 555-1857	52
1039	Shih Lin Chao	(617) 555-5921	1
1142	Alison Clark	(510) 555-9437	57
1596	Catherine Pickett	(617) 555-3478	53

Product Sales Summary

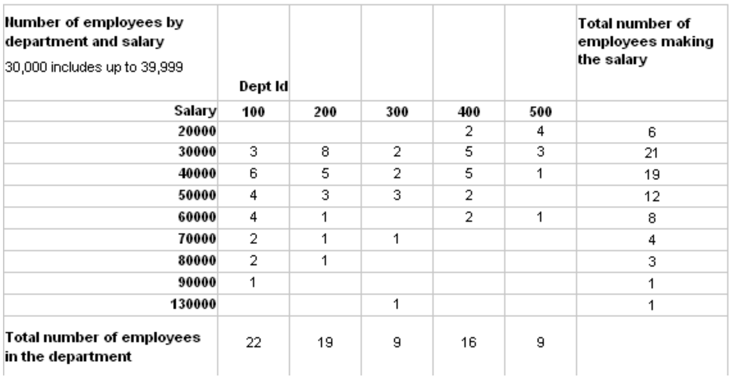
Product ID	Product Name	Product Description	Quantity Sold	Dollars
300	Tee Shirt	Tank Top	2374	$21,366
301	Tee Shirt	V-neck	2403	$33,642
302	Tee Shirt	Crew Neck	2139	$29,946
400	Baseball Cap	Cotton Cap	3310	$29,790
401	Baseball Cap	Wool cap	2677	$26,770
500	Visor	Cloth Visor	2652	$18,564
501	Visor	Plastic Visor	2508	$17,556
600	Sweatshirt	Hooded Sweatshirt	3060	$73,440
601	Sweatshirt	Zipped Sweatshirt	2748	$65,952
700	Shorts	Cotton Shorts	4536	$68,040

For more about the Composite presentation style, see Chapter 10, "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 71. 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:



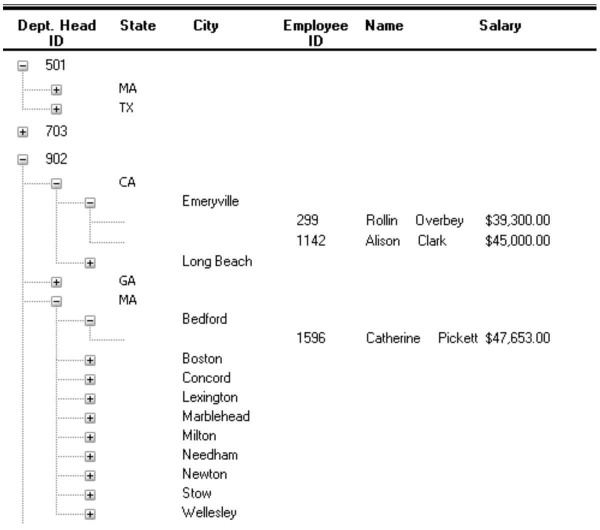
Number of employees by department and salary 30,000 includes up to 39,999	**Dept Id**					**Total number of employees making the salary**
Salary	**100**	**200**	**300**	**400**	**500**	
20000				2	4	6
30000	3	8	2	5	3	21
40000	6	5	2	5	1	19
50000	4	3	3	2		12
60000	4	1		2	1	8
70000	2	1	1			4
80000	2	1				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 14, "Working with Graphs," and Chapter 11, "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  State   City          Employee  Name                Salary
    ID                               ID
⊟ 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 12, "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 In the Solution Explorer, right-click the library where you want to save the DataWindow object and select Add New Entry.

2 In the Add New Entry dialog box, select DataWindow Object from the categories list.

3 Select a DataWindow style from the Templates list.

   The presentation style determines how the data is displayed. See “Choosing a presentation style” on page 65.

4 Provide a name for the DataWindow object, and click Add.

   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.”

   The appropriate DataWindow object wizard for the style you selected starts.

5 If you want data to be retrieved in the Preview view when the DataWindow object opens, select the Retrieve on Preview check box.

6 Define the data source.

   See “Selecting a data source” on page 75.

7 Choose options for the DataWindow object and click Next.

   See “Choosing DataWindow object-wide options” on page 109.

8 Review your specifications and click Finish.

   The DataWindow object displays in the Design view.

9 Save the DataWindow object.

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 a SQL Anywhere data source (meaning a database file) or an XML data source (meaning an XML file). *Connecting to Your Database* uses the term data source in this second sense.

If the data is in the database

If the data for the DataWindow object will be retrieved from a database, choose one of the data sources from Table 3-2.

Table 3-2: Data source choices for data from a database

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 *or* 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

Web Service data source Select the Web Service data source if you want to populate the DataWindow object with data you obtain from a Web service.

For more information, see “Using a Web service data source” on page 107.

External data source 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.

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 78.
4. (Optional) Sort the rows before you retrieve data.

   For more information, see "Specifying sorting criteria" on page 79.
5. (Optional) Select what data to retrieve.

   For more information, see "Specifying selection criteria" on page 80.
6. Click the OK button in the 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 4, "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.

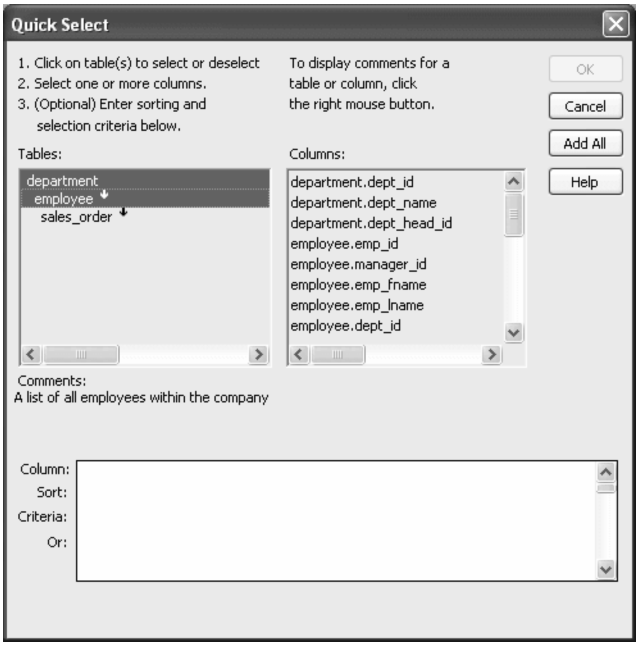
Which tables and views display?

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. SQL 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. You can select any of these related tables if you want to include columns from them in the DataWindow object.



1. Click on table(s) to select or deselect
2. Select one or more columns.
3. (Optional) Enter sorting and selection criteria below.

To display comments for a table or column, click the right mouse button.

Tables:
department
employee
sales_order

Columns:
department.dept_id
department.dept_name
department.dept_head_id
employee.emp_id
employee.manager_id
employee.emp_fname
employee.emp_lname
employee.dept_id

Comments:
A list of all employees within the company

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.

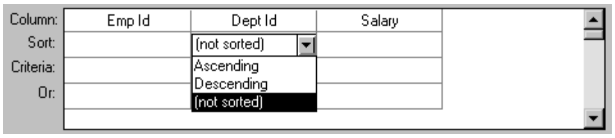
Specifying sorting criteria

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 3-3: 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

Operator	Meaning
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 wildcard asterisk (*) used in many applications, 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

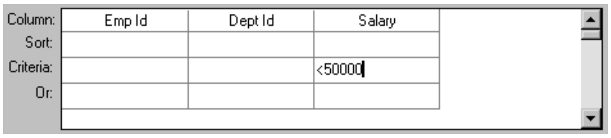
This technique is particularly handy when you want to retrieve a range of values in a column. See example 6 below.

SQL expression examples

The first six examples in this section all refer to a grid that contains three columns from the employee table: emp_id, dept_id, 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.



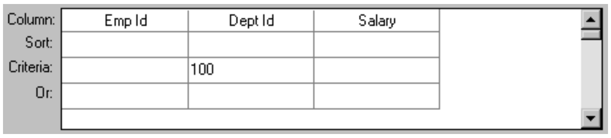
Column:	Emp Id	Dept Id	Salary
Sort:			
Criteria:			<50000
Or:			

The SELECT statement that DataWindow Designer creates is:

```
SELECT employee.emp_id,
   employee.dept_id,
   employee.salary
FROM employee
WHERE employee.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.



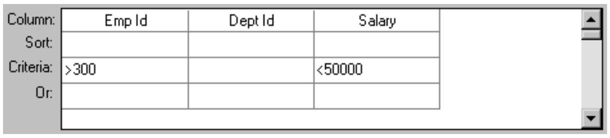
Column:	Emp Id	Dept Id	Salary
Sort:			
Criteria:		100	
Or:			

The SELECT statement that DataWindow Designer creates is:

```
SELECT employee.emp_id,
   employee.dept_id,
   employee.salary
FROM employee
WHERE employee.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 employee.emp_id,
   employee.dept_id,
   employee.salary
FROM employee
WHERE (employee.emp_id >'300') AND
   employee.salary <'50000'
```

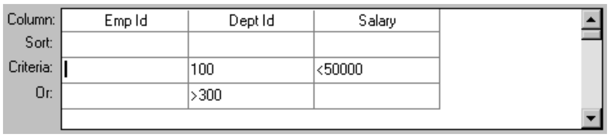
Example 4

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



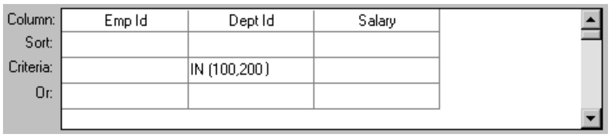
Column:	Emp Id	Dept Id	Salary
Sort:			
Criteria:		100	<50000
Or:		>300	

The SELECT statement that DataWindow Designer creates is:

```
SELECT employee.emp_id,
   employee.dept_id,
   employee.salary
FROM employee
WHERE (employee.dept_id = '100') AND
   (emplyee.salary < '50000')OR
   (employee.dept_id > '300')
```

Example 5

The expression `IN(100,200)` in the Criteria row in the DeptId column in the grid retrieves information for employees who are in department 100 *or* 200 *or* 500.



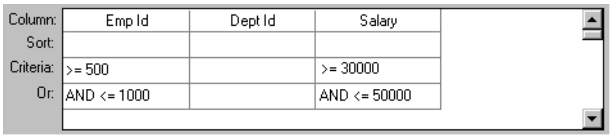
Column:	Emp Id	Dept Id	Salary
Sort:			
Criteria:		IN (100,200)	
Or:			

The SELECT statement that DataWindow Designer creates is:

```
SELECT employee.emp_id,
   employee.dept_id,
   employee.salary
FROM employee
WHERE employee.dept_id IN ('100,200')
```

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.



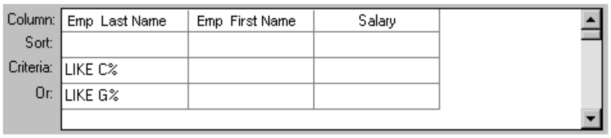
Column:	Emp Id	Dept Id	Salary
Sort:			
Criteria:	>= 500		>= 30000
Or:	AND <= 1000		AND <= 50000

The SELECT statement that DataWindow Designer creates is:

```
SELECT employee.emp_id,
   employee.dept_id,
   employee.salary
FROM employee
WHERE (((employee.emp_id >='500') AND
   (employee.salary >='30000') AND
   (employee.emp_id <='1000') AND
   (employee.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 employee.emp_last_name,
   employee.emp_first_name,
   employee.salary
FROM employee
WHERE (((employee.emp_last_name LIKE 'C%'))OR
   ((employee.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 4, “Enhancing DataWindow Objects.”

- You can place the DataWindow object in query mode using the Modify method.

For more information, see the *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 SQL 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 SQL Select painter, you can save the current SELECT statement as a query by selecting File>Save As from the menu bar. Doing so allows you to easily use this data specification again in other DataWindows.

For more information about queries, see "Defining queries" on page 112.

❖ **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 88.

4 Join the tables if you have selected more than one.

   For more information, see "Joining tables" on page 91.

5 Select retrieval arguments if appropriate.

   For more information, see "Using retrieval arguments" on page 94.

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 96.

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 OK.

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 SQL Select painter. What tables and views display in the dialog box depends on the DBMS. For some DBMSs, 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. SQL Anywhere does not restrict the display, so all tables and views display, whether or not you have authorization.

❖ **To select the tables and views:**

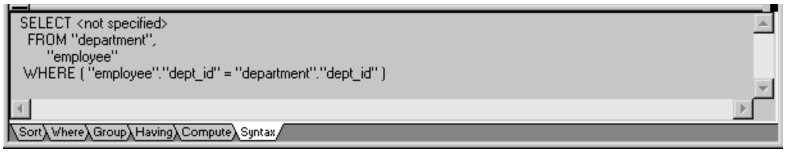
- Do one of the following:
  - Click the name of each table or view you want to open.

    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.



```
SELECT <not specified>
  FROM "department",
      "employee"
 WHERE ( "employee"."dept_id" = "department"."dept_id" )
```

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 SQL Select painter

The colors used by the SQL 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 24.

Adding and removing tables and views

You can add tables and views to your Table Layout view at any time.

Table 3-4: Adding tables and views in the SQL 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 SQL Select painter, DataWindow Designer joins columns based on their key relationship.

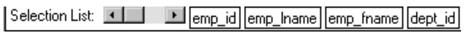
For information about joins, see “Joining tables” on page 91.

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 SQL 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.



❖ **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 a SQL 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 DataWindow 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 SQL 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 4, “Enhancing DataWindow Objects.”

Displaying the underlying SQL statement

As you specify the data for the DataWindow object in the SQL 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 SQL 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 SQL Select painter.
- Click OK 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 SQL 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.

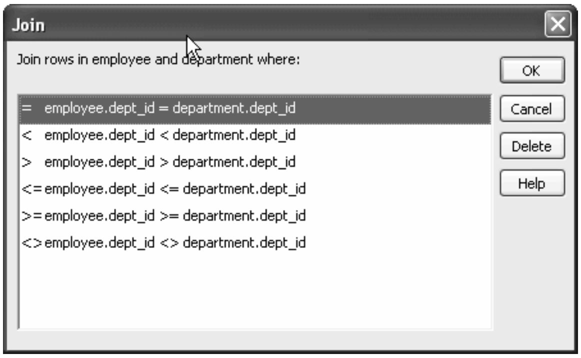
❖ **To join tables:**

1 Click the Join button in the toolbar.

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:



Join rows in employee and department where:

```
=   employee.dept_id = department.dept_id
<   employee.dept_id < department.dept_id
>   employee.dept_id > department.dept_id
<=  employee.dept_id <= department.dept_id
>=  employee.dept_id >= department.dept_id
<>  employee.dept_id <> department.dept_id
```

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 SQL 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

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.

Adding retrieval arguments

If you decide later that you need arguments, you can return to the SQL 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 4, “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 SQL 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 to retrieve 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.

```
int x[3]
// Now populate the array with values
// such as x[1] = sle_dept.Text, and so on,
// then retrieve the data, as follows.
dw1.Retrieve(x)
```

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 3-5: 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
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 SQL 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 8, "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).

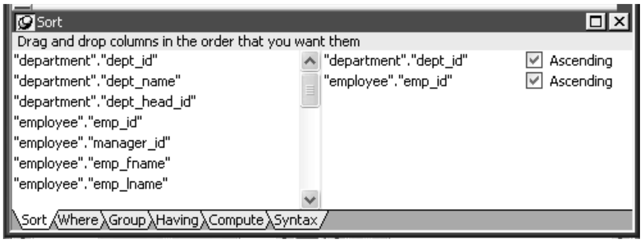
The columns you selected display in the order of selection. You might need to scroll to see your selections.

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:



"department"."dept_id"
"department"."dept_name"
"department"."dept_head_id"
"employee"."emp_id"
"employee"."manager_id"
"employee"."emp_fname"
"employee"."emp_lname"

"department"."dept_id"	✓ Ascending
"employee"."emp_id"	✓ Ascending

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.

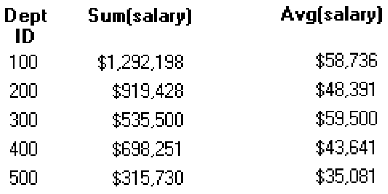
Defining GROUP BY criteria

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.

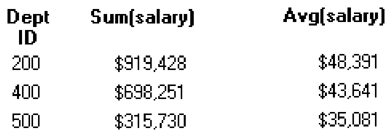
5. Click the Return button to return to the DataWindow painter.

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 97.

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, and 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 112.

Using External

If the data for the DataWindow object does not come from a database (either through a native Sybase database interface or through a standard database interface), 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. The number datatype is equivalent to the DataWindow Designer double datatype.

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 code, 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 or 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 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 4, “Enhancing DataWindow Objects.”

For information about using a stored procedure to update the database, see “Using stored procedures to update the database” on page 164.

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 4, “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 101.

Datatype mappings

Table 3-6 lists .NET datatypes and the DataWindow datatypes to which they map when you use a DataTable as a data source.

Table 3-6: 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*.

Using a Web service data source

You can use a Web service as the data source for a DataWindow object with any presentation style.

Using the DataWindow wizard

When you select Web Service as the data source and click Next, the DataWindow wizard opens a page that prompts you to select a WSDL file. The file you select should be in a publicly accessible location for all members of the development team. You can enter the URL to a WSDL, ASMX, or XML file, or you can browse a mapped drive for these types of files.

The Choose WSDL File page of the DataWindow wizard also lets you name the assembly file that the wizard will create. The assembly file serves as an interface between the DataWindow and the Web service. If you do not name the assembly file, the wizard will select a name for you based on the name of the WSDL file entry.

The next step to access a Web service data source is to select a service described in the WSDL, and then one of its public methods. You must then select a parameter for the DataWindow to use as the result set for the method.

A DataWindow typically obtains its data from an array of structures. Because a Web service method can pass an array of structures in one of its arguments rather than in a return value, the wizard prompts you to select one of the method's arguments or its return value as the designated result set for the method. If you want data for a single row and column only, you can select a parameter that has a simple datatype. You can also select a parameter that is an array of simple datatypes rather than an array of structures.

You complete the wizard as you would when using any other type of data source for your DataWindow. After you complete the wizard, the DataWindow displays in the DataWindow painter. However, there is no equivalent to the SQL painter for a DataWindow with a Web service data source. For this type of DataWindow, you cannot select Design>Data Source from the DataWindow painter menu to change selected columns or modify the DataWindow syntax.

Runtime requirements on a deployment computer
To run the Web service DataWindow application from a deployment computer, the assembly file that you generate with the wizard must be copied along with the application executable and required runtime DLLs for Web service applications. For information on the required DLLs and the Runtime Packager tool that you can use to deploy them, see "Deploying DataWindow .NET Applications" in the *Programmer's Guide*.

For information on rebuilding an assembly generated by the DataWindow wizard, see “Regenerating an assembly” on page 169.

Datatype mappings

Table 3-7 lists .NET datatypes and the DataWindow datatypes to which they map when you use a .NET Web service as a data source. Arrays are also supported for these datatypes except for System.Byte.

Table 3-7: Datatype mapping for .NET datatypes

.NET datatype	DataWindow datatype
System.Boolean	long (Handled as a boolean at runtime.)
System.Byte	ulong
System.DateTime	datetime (Minimum and maximum dates for .NET can be outside the range of dates supported by DataWindow .NET. DataWindow .NET does not support dates prior to the year 1000 or after the year 3000.)
System.Decimal	decimal
System.Double	number
System.Int16	long
System.Int32	long
System.Int64	decimal
System.SByte	long
System.Single	real
System.String	string(40)
System.UInt16	ulong
System.UInt32	ulong
System.UInt64	decimal

The DataWindow can also use a Web service data source that has structures for parameters, as long as the structures are composed of the simple datatypes that can be mapped to DataWindow datatypes. An array of structures can be mapped to *n* rows with *x* columns where *n* is the size of the array and *x* is the number of members in the structure. Nested structures are not supported.

Using parameters by reference

A parameter passed by reference is a bidirectional [IN,OUT] parameter by definition. The Web Service DataWindow wizard lets you select a Web service method [OUT] or [IN,OUT] parameter, instead of the method return value, to pass a result set to a DataWindow object. However, the parameter you select cannot be used for both a return value and a retrieval argument by the same DataWindow object.

Database-related functions and events

In the Web Service DataWindow, some database or transaction-related functions and events are not supported and meaningless because the Web Service DataWindow has no direct relation to the database. The following functions cannot be used with the Web Service DataWindow: GetSqlSelect, SetSqlSelect, and SetTransaction.

The DbError event is also not supported for the Web Service DataWindow. Instead, you can use the WSError error event to handle errors during retrieve, insert, or update operations.

Using the Web Service Connection object

Some Web services support or require a user ID and password, and other session-related properties like firewall settings. The WebServiceConnection object can provide this information for your DataWindow connections.

You use an instance of the WebServiceConnection object to connect to a Web service by calling the SetWSConnection method.

The following C# code instantiates a WebServiceConnection object with user-related and authentication information, then sets the object as the connection object for a Web service data source:

```
private void button_Click(object sender, EventArgs e)
{
   WebServiceConnection w = new WebServiceConnection();
   w.UserName =  "johndoe";
   w.Password = "mypassword";
   w.EndPoint = "myendpoint";
   w.AuthenticationMode = "basic";
   w.UseWindowsIntegratedAuthentication = true;
   dw1.SetWSConnection(w);
}
```

For more information about updating the database with a Web service DataWindow, see “Using a Web service to update the database” on page 166.

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 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 2, “Managing the Database,” and Appendix B, “The Extended Attribute System Tables.”

Modifying an existing DataWindow object

❖ **To modify an existing DataWindow object:**

- Double-click the DataWindow object in the Solution Explorer.

  The DataWindow object opens in the DataWindow painter. To learn how you can modify an existing DataWindow object, see Chapter 4, “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 new query:**

1. In the Solution Explorer, right-click the library where you want to save the query object and select Add New Entry.
2. In the Add New Entry dialog box, select Query from the categories list.
3. Provide a name for the query object and click Add.

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."

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 85.

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 2, "Managing the Database."

3 When you have finished previewing the query, click the Close button in the PainterBar to return to the Query painter.

Modifying a query

❖ **To modify a query:**

- Double-click the query in the Solutions Explorer.

CHAPTER 4

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

Related topics

Other ways to enhance DataWindow objects are covered in later chapters:

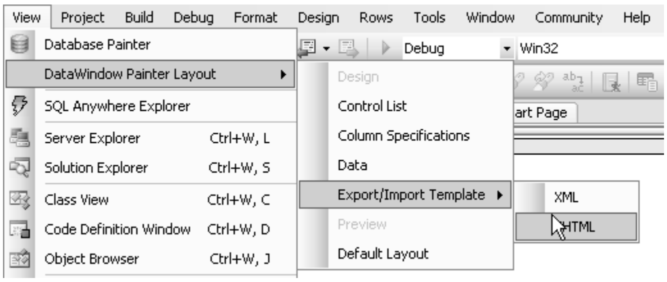
Chapter	Explains how to
Chapter 6, “Working with Controls in DataWindow Objects”	Add controls to a DataWindow object and reorganize, position, and rotate them
Chapter 5, “Controlling Updates in DataWindow objects”	Control update capabilities
Chapter 7, “Displaying and Validating Data”	Specify display formats, edit styles, and validation rules for column data
Chapter 8, “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 9, “Highlighting Information in DataWindow Objects”	Highlight data by using conditional expressions to modify the properties of controls in DataWindow objects
Chapter 10, “Using Nested Reports”	Place reports inside DataWindow objects
Chapter 14, “Working with Graphs”	Use graphs to visually present information retrieved in a DataWindow object
Chapter 11, “Working with Crosstabs”	Use crosstabs to present analyses of data retrieved in a DataWindow object

Chapter	Explains how to
Chapter 12, “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 default layout in the DataWindow painter shows the Design view and Preview view. To display the other views, select View>DataWindow Painter Layout and select the views you want.



You can rearrange the views using the handle in the top-left corner. For more information, see “Using views in painters” on page 8.

Design view

The Design view 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.

Preview view

The Preview view shows the DataWindow object with data as it will appear at runtime. If the Print Preview toggle (File>Print Preview) is selected, you see the DataWindow object as it would appear when printed with an optional blue outline that shows where the page margins are located.

Export/Import Template view for XML

The Export/Import Template view for XML 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 13, “Exporting and Importing XML Data.”

Export Template view for XHTML

The Export Template view for XHTML 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 *Programmer's Guide*.

Control List view

The Control List view lists all controls in the DataWindow object. Selecting controls in this view selects them in the Design view and the Visual Studio Properties window. You can also sort controls by Control Name, Type, or Tag.

Data view

The Data view 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 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).

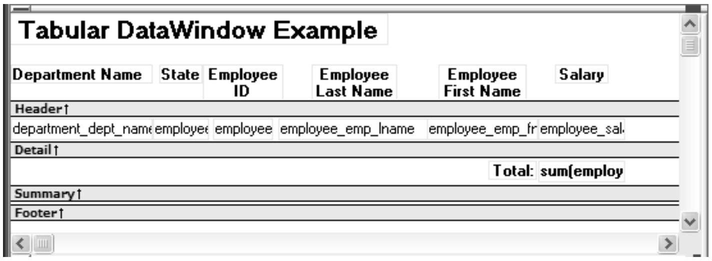
Understanding the DataWindow painter Design view

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.


Tabular DataWindow Example

Department Name	State	Employee ID	Employee Last Name	Employee First Name	Salary
department_dept_nam	employee	employee	employee_emp_lname	employee_emp_fr	employee_sal
				Total:	sum(employ

Header↑
Detail↑
Summary↑
Footer↑

Table 4-1: 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 175

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:

(Height of the DataWindow object *– Height of headers and footers) / 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.

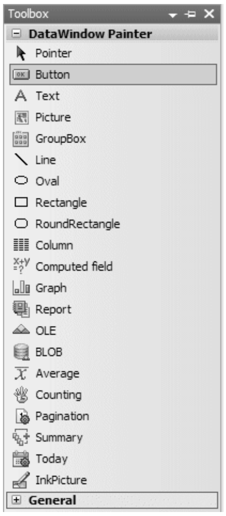
The summary and footer bands

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

Adding controls to a DataWindow object

One of the ways you can enhance a DataWindow object is to add controls, such as columns, drawing objects, buttons, and computed fields. To add a control, select it in the Visual Studio toolbox and click where you want to place it in the DataWindow painter Design view.



For more information about adding controls, see Chapter 6, "Working with Controls in DataWindow Objects."

Using the DataWindow painter toolbar

The DataWindow painter toolbar includes buttons for common formatting and database operations. It has five drop-down toolbars, which are indicated by a small black triangle on the right part of a button. Table 4-2 lists the drop-down toolbars that are available.

Table 4-2: Drop-down toolbars in the DataWindow painter

Toolbar	Used to
Alignment	Aligns selected controls.
Spacing	Spaces selected controls evenly either horizontally or vertically
Sizing	Makes selected controls the same height, width, or both
Border	Specifies the border style of selected controls
Slide	Specify sliding for controls.

Using the Properties window in the DataWindow painter

Each part of the DataWindow object (such as text, columns, computed fields, bands, graphs, even the DataWindow object itself) has a set of properties appropriate to the part. The properties display in the Properties window.

You can use the Properties window to modify the parts of the DataWindow object.

❖ **To use the Properties window 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 window displays. The window displays the properties of the currently selected control(s), the band, or the DataWindow object itself. The contents of the Properties window change as different controls are selected (made current).

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 in a DataWindow object in the Control List view:**

1. Select View>DataWindow Painter Layout>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.

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>DataWindow Painter Layout>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 128.

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 toolbar 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 clear 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 toolbar.
  - 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 155.

- **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 155.

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.

Changing input language

You (and your users) can add or modify data in a DataWindow object in multiple input languages. If you use multiple input languages, you can display a Language bar on your desktop to change the current input language. In a DataWindow object, the input language in effect the first time a column gets focus becomes the default input language for that column. If you subsequently change the input language when that column has focus, the new input language becomes the default for that column. This behavior does not apply to columns that have the RightToLeft property set.

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 5, “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 next.

❖ **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 toolbar 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 DataWindow painter. You can then click the Update Database button in the toolbar 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 13, "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.

Using the IntelliMouse pointing device
Using the IntelliMouse pointing device, users can scroll a DataWindow object by rotating the wheel. Users can also zoom a DataWindow object larger or smaller by holding down the Ctrl key while rotating the wheel.

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.

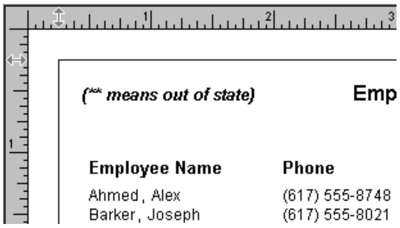
Changing margins

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, set the PrintPreviewOutline property to False in the Print Specifications category in the Properties window.



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 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.

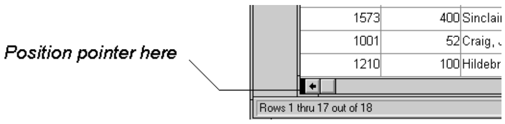
DataWindow Designer selects the column and displays a line representing the column border.

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 scroll bar.



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 13, “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 Ghostscript. You can save all types of reports to PDF.

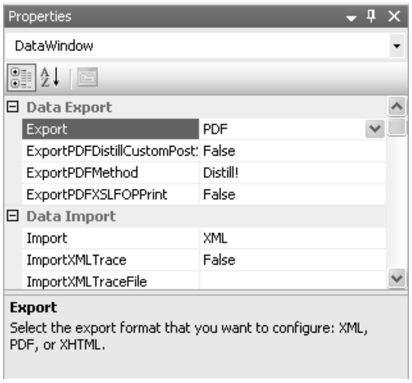
DataWindow Designer uses a PostScript printer driver specifically designed for distilling purposes to configure the PDF output. You can choose to use a different PostScript printer driver if you want to customize your PostScript settings for generating PDF.

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 Expand the Data Export category in the Properties window for the DataWindow object and select PDF from the Export drop-down list.



Data Export	
Export	PDF
ExportPDFDistillCustomPost:	False
ExportPDFMethod	Distill!
ExportPDFXSLFOPPrint	False
Data Import	
Import	XML
ImportXMLTrace	False
ImportXMLTraceFile	

Export
Select the export format that you want to configure: XML, PDF, or XHTML.

2 Select Distill! from the ExportPDFMethod drop-down list and True from the ExportPDFDistillCustomPostScript list.

3 Expand the Print Specifications category and specify the name of the printer whose settings you want to use in the PrintPrinterName field.

4 Save the DataWindow object, then, with the Preview view selected, 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 Ghostscript, you must download and install Ghostscript files on your system as described in the chapter on deploying applications in the *Programmer's Guide*. You also need to install PostScript driver files. If you have installed a PostScript printer on your computer, these driver files are already installed. If you have never installed a PostScript printer, use the Printers and Faxes option in the Windows control panel to install a generic PostScript printer. If the *Pscript5.dll* has never been installed, you may be prompted to insert the Windows install CD.

Other related files are installed in *Sybase\DataWindow .NET 2.5\drivers*.

Saving as PDF fails at runtime on Windows 2003 Server. This is caused by a Group Policy that by default disallows installation of printers that use kernel-mode drivers. Kernel-mode drivers have access to system-wide memory, and poorly written drivers can cause system failures. To allow installation of kernel-mode drivers, follow these steps:

1 Select Run from the Windows Start menu.

2 In the Open box, type `gpedit.msc` and click OK.

3 In the Group Policy console, expand Computer Configuration, Administrative Templates, and Printers.

4 Disable "Disallow Installation of Printers Using Kernel-Mode Drivers."

When you deploy applications that use the ability to save as PDF with the distill method, you must make sure your users have installed Ghostscript and have access to the *drivers* directory.

The Ghostscript files and the *drivers* directory must be in the same directory as the *pbdwn110.dll* runtime file. For example, if you deploy your application and *pbdwn110.dll* and the other runtime files in a directory called *MyApplication*, the Ghostscript files must be in *MyApplication\gsn.nn*, and the 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 downloading Ghostscript and redistributing the PostScript printer drivers.

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. 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. DataWindows 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 Web browser.
7. Use the browser's file open command to open the HTML file.

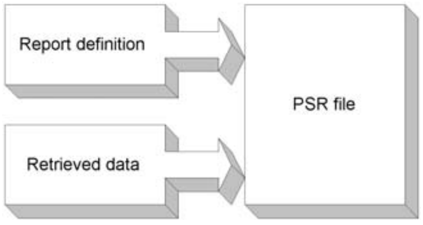
Working with PSR files

A PSR file is a special file with the extension PSR created by DataWindow Designer, 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 DataWindow definition (source and object) as well as the data contained in the DataWindow object when the PSR file was created.

Figure 4-1: PSR file



About reports
A report is the same as a nonupdatable DataWindow object. For more information, see “Reports versus DataWindow objects” on page 64.

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 133. 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.
2. Expand the General category and select the unit of measure you want to use to specify distances when working with the DataWindow object:
   - PowerBuilder (normalized) units
   - 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. The default color is the window background color.
5. If the DataWindow contains buttons, set the ShowBackColorOnXP property to make sure that the background color you select for the buttons displays on systems using the XP style.

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 Under the General category in the Properties window 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 Under the General category in the Properties window, 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.

  You can set colors in the Appearance category in the Properties window.

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 under the General category in the Properties window as described in Table 4-3.

Table 4-3: Options for grid DataWindow objects

Option	Result
GridColumnMove	True – Columns can be moved at runtime
GridLines	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

Option	Result
RowResize	True – Rows can be resized at runtime
SelectedMouse	True – Data can be selected at runtime (and, for example, copied to the clipboard)

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 window.

   You can set a pointer for the entire DataWindow object, specific bands, and specific controls.

2 Select the Pointer category for the DataWindow object and bands or the Appearance category for controls.

3 Choose the pointer from the list.

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.

2 In the Units box under the General category, 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 category.

The Print Specifications properties use the units of measure you specified.

4 Specify print specifications for the current DataWindow object.

See Table 4-4 for more information.

Table 4-4: Setting print specifications for DataWindow objects

Setting	Description
PrintButtons	Select True to display Button controls when you print the report. The default is to hide them.
PrintClipText	Select True 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.
PrintCollateCopies	Select True to collate copies when printing. Collating increases print time because the print operation is repeated to produce collated sets.
PrintColumns	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, specify the number of columns. See "Printing with newspaper-style columns" next.
PrintColumnsWidth	An integer that specifies the width of each column if you specified multiple columns.
PrintDefaultPrinter	Select False if a printer has been specified in the Printer Name filed and you do not want the report to be sent to the default system printer if the specified printer cannot be found. This field is True by default if a printer name is specified.
PrintDocumentName	Specify a name to be used in the print queue to identify the report.
PrintMarginBottom, PrintMarginLeft, PrintMarginRight, PrintMarginTop	Specify top, bottom, left, and right margins.
PrintOrientation	Choose one of the following: • Default: Uses the default printer setup. • Landscape: Prints the contents of the DataWindow object across the length of the paper. • Portrait: Prints the contents of the DataWindow object across the width of the paper.
PrintOverridePrintJob	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 True to override the default print job settings and use the print settings defined for this report.
PrintPaperSize	Choose a paper size.
PrintPaperSource	Choose a paper source.

Setting	Description
PrintPreviewButtons	Select True to display Button controls in Print Preview. The default is to hide them.
PrintPreviewOutline	Select True to display a blue outline to show the location of the margins.
PrintPrinterName	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.
PrintPrompt	Select True to display the standard Print Setup dialog box each time users make a print request.

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 runs (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 category.

4 Specify the number of columns across the page and the width of columns in the Columns and Columns Width properties.

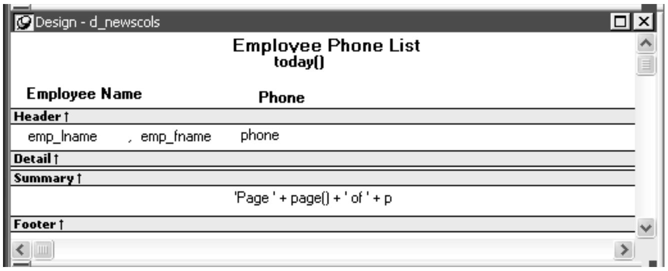
5 For each control in the DataWindow object that you do *not* want to have appear multiple times on the page (such as headers), select Properties from the control's pop-up menu and set the HideSnaked property to True under the General category.

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.



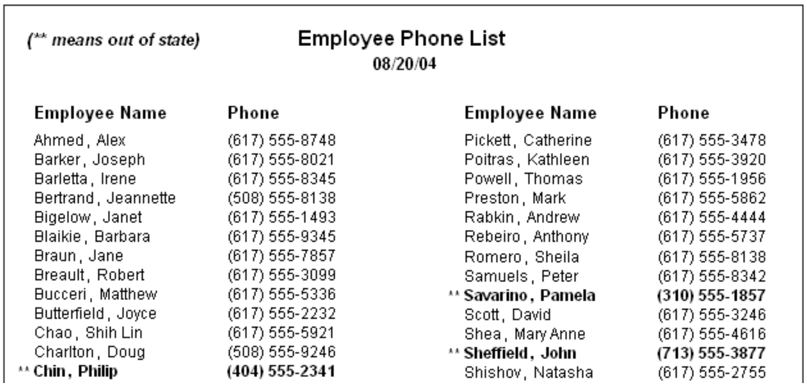
2 Under the Print Specifications category in the DataWindow object's Properties window, specify two columns across and a column width of 3.5 inches. (Make sure that Units is set to inches under the General category.)

3 To view the DataWindow object as it will be printed, place the pointer in the Preview view and select File>Print Preview.

  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 set the HideSnaked property.

The finished DataWindow object has one set of page heading information and two columns of column header and detail information.



*(** means out of state)*

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.

2 Type the new text in the Properties window.

❖ **To change the text properties for a text control in a DataWindow object:**

1 Select the text control.

2 Change the Font properties in the Properties window.

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 runs, 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 5, "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 the toolbar .

   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.

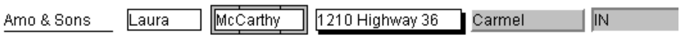
❖ **To specify prefixes for naming controls systematically in a DataWindow object:**

1 Select Design>Options from the menu bar and then select the Prefixes tab.

2 Change prefixes as desired and click OK.

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:



Border appearance varies
Changing the border style may not have the same effect on all Windows operating systems and display settings.

You can specify a border for one or more controls in the Properties window.

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. Set the HeightAutoSize property to True in the Layout category.
3. Set the AutoHScroll property to False under Edit in the Behavior category.

   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 Set the HeightAutoSize property to True in the General group.

In the Preview view, the band resizes based on the contents of the columns you defined as having their height sized automatically.

Using the RowHeight function with HeightAutoSize

When a detail band has HeightAutoSize 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 in the Position category in the Properties window, 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 HeightAutoSize columns without a HeightAutoSize 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 SQL Select painter. (If you used Quick Select to define the data source, this might be the first time you have seen the SQL 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 85.

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 5, "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 datatypes 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 SQL 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 SQL 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 3, “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 102.

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 7, "Displaying and Validating Data."

❖ **To store data in a DataWindow object:**

1 If the Data view is not already displayed, select View>DataWindow Painter Layout>Data from the menu bar.

   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 toolbar 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 toolbar 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 toolbar 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 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 form opens showing such a DataWindow, the data is already there. There is no need to issue Retrieve to get the data.

DataWindow Designer *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 Designer handles it the same as a DataWindow object that is not stored with data: DataWindow Designer 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>DataWindow Painter Layout>Column Specifications from the menu bar.
2. Select the Prompt check box next to each column for which you want to specify retrieval criteria at runtime.

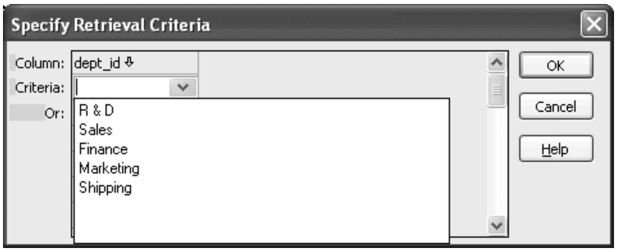
When you specify prompting for criteria, DataWindow Designer displays the Specify Retrieval dialog box just before a retrieval is to be done.

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 DropDownListBox 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 to display, set the CriteriaOverrideEdit property in the Behavior category in the column's Properties window.

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 setting CriteriaRequired to True in the Behavior category of the column's Properties window. 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 76 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 users page 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.

CHAPTER 5

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

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 146.

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.

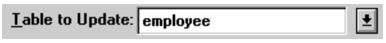
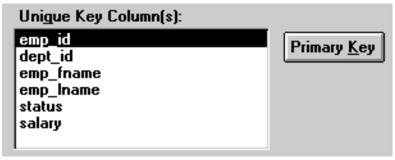


Table to Update: employee

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):



Unique Key Column(s):
emp_id
dept_id
emp_fname
emp_lname
status
salary

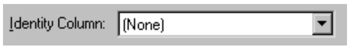
The key columns you select here must uniquely identify a row in the table. They can be the table's primary key, though they don't have to be.

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:



Identity Column: (None)

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.

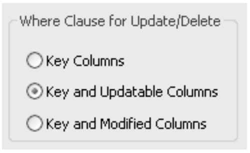
Updatable columns are displayed highlighted. Click a nonupdatable column to make it updatable. Click an updatable column to make it nonupdatable.

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:


Where Clause for Update/Delete

Key Columns [✓ Key and Updatable Columns] Key and Modified Columns

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 5-1 in the Where Clause for Update/Delete box. The results are illustrated by an example following the table.

Table 5-1: Specifying the WHERE clause for UPDATE and DELETE

Option	Result
Key Columns	The WHERE clause includes the key columns only. These are the columns you specified in the Unique Key Columns box. 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. **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). 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.
Key and Updatable Columns	The WHERE clause includes all key and updatable columns. 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.
Key and Modified Columns	The WHERE clause includes all key and modified columns. 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.

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

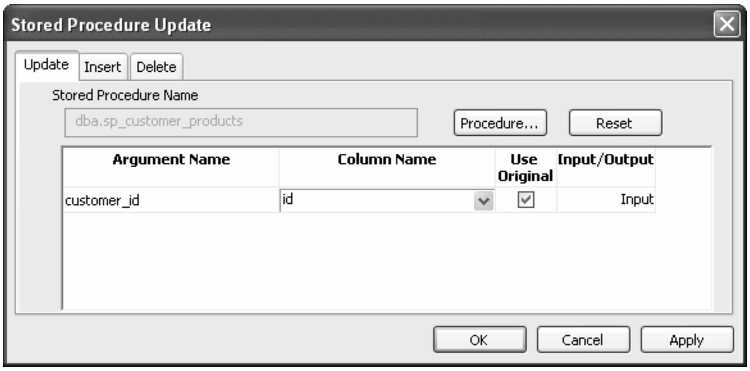
Updates to the database can be performed using stored procedures.

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.



Stored Procedure Name: dba.sp_customer_products

Argument Name	Column Name	Use Original	Input/Output
customer_id	id	✓	Input

❖ 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.

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 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.

Using a Web service to update the database

You can use a DataWindow with a Web service data source to update a database. Support for updating data requires one or more WSDL files that describe methods and parameters that can be called by the DataWindow engine for insert, delete, or update operations.

Generating or selecting an assembly

The WSDL files are not required on runtime computers. They are used to generate assembly files that are deployed with the application. If you have an existing assembly file that allows you to update data in your DataWindow objects, you can select that assembly instead of generating a new one from the Web Services Update dialog box. You can generate or select separate assemblies for insert, delete, and update operations.

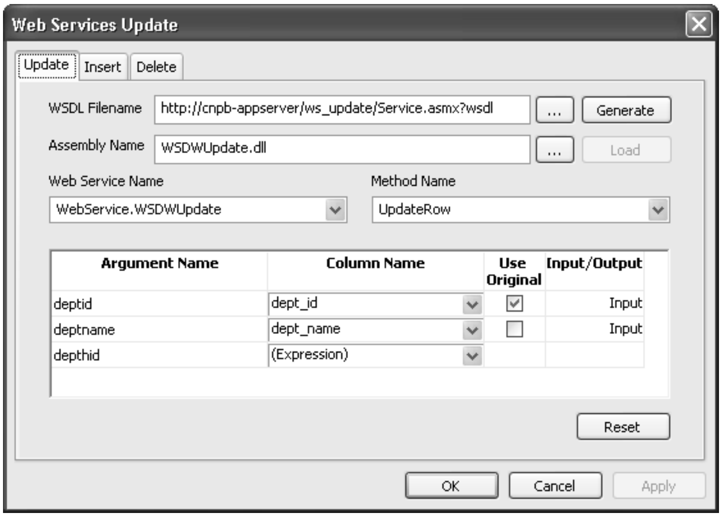
Insert, delete, and update operations

The insert, delete, and update operations imply different things depending on the original data source. When you insert a DataWindow row for an RDBMS, a new row is added to the database; when the data source is an array of structures, a new structure instance is added to the array; and when the data source is an array of simple types, a new instance of the simple type is added to the array. The delete operation removes a database row or an instance in an array, and the update operation modifies a database row or an instance in an array.

For each operation, you must map DataWindow column values or expressions to Web service input parameters. At runtime when performing one of these operations, the DataWindow binds column or expression values to parameters as instructed and calls the Web service method. The DataWindow engine does not know what actually happens in the Web service component (that is, how the component implements the update), only whether it returns a success or failure message.

Figure 5-1 displays the Web Service Update dialog box. You use this dialog box to bind to Web service parameters to DataWindow columns or expressions. Unlike the retrieve call, DataWindow update operations can handle bidirectional parameters. However, you can select an expression or computed column only for an update method input parameter.

Figure 5-1: Web Service Update dialog box



WSDL Filename: http://cnpb-appserver/ws_update/Service.asmx?wsdl

Assembly Name: WSDWUpdate.dll

Web Service Name: WebService.WSDWUpdate

Method Name: UpdateRow

Argument Name	Column Name	Use Original	Input/Output
deptid	dept_id	✓	Input
deptname	dept_name	☐	Input
depthid	(Expression)		

❖ **To use a Web service to update the database**

1 In the DataWindow painter, select Rows>Web Service Update to display the Web Service Update dialog box.

2 Select the tab for the Web service update method (Update, Insert, or Delete) with which you want to associate a Web service.

3 Click the browse button next to the WSDL Filename text box to browse to a WSDL file describing the Web service you want to use to update the DataWindow, and click OK.

You use a WSDL file to generate an assembly that you can deploy with your Web service DataWindow application. You can override the default assembly name that will be generated if you enter an existing assembly in the following step of this procedure.

If you already have an assembly that you want to use to update the DataWindow, you can skip the current step and select the assembly that you want in step 4.

You can use the Reset button to clear all entries in the Web Service Update dialog box.

4 (Optional) Type an assembly name in the Assembly Name text box to override a default assembly name that you want to generate from a WSDL file, or browse to an existing assembly file that describes the Web service you want to use to update the DataWindow, and click OK.

Although you can browse to any mapped directory to find an assembly file for update operations, you must make sure to copy the assembly under the current target directory. All assemblies for retrieving and updating a Web service DataWindow must be deployed to the same directory as the application executable file, or retrieve and update operations will not be able to work at runtime.

5 Click Generate if you want to generate and load an assembly file, or click Load if you entered an existing assembly file name in step 4.

After you click Generate, an assembly file is created with a default name from the WSDL file or from a name that you entered in the previous step.

After you generate the assembly from a WSDL file or load an existing assembly, the Web services in that file are added to the Web Service Name drop-down list and the methods for the Web services are added to the Method Name drop-down list.

6 Select a Web service name and method name from the list of Web services and methods.

The parameters used in the Web service method are displayed in the Argument Name list in the order in which they are defined. Column Name lists the columns used in your DataWindow object.

7 Associate a column in the DataWindow object or an expression with a method parameter.

If a Web service method 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 Web service method parameter
You must be careful to correctly match a column in the DataWindow object to a method parameter, since DataWindow Designer is able to verify only that datatypes match.

8 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 Web service parameter is used in the WHERE clause of an UPDATE or DELETE SQL statement for a Web service method. If you do not select Use Original, the parameter uses the new value entered for that column. Typically, you would use the new value when the Web service parameter is needed for an INSERT SQL statement for the method, or if it is set in an UPDATE SQL statement.

Regenerating an assembly

If you need to regenerate an assembly for a DataWindow that uses a Web service data source for retrieval, update, insert, or delete operations, you must add the following line to the [DataWindow] section of the *DW.INI* file:

```
GenerateWSAssembliesOnCompile=YES
```

After you set this property in the *DW.INI* file, DataWindow Designer regenerates the assembly on each compilation of the target containing the DataWindow.

The WebServiceException object

Because a DataWindow with a Web service data source does not pass back failure messages in a return code during retrieve, insert, or update operations, you must use the WebServiceException object to obtain such error information. The parameters in the following table are exposed in the WebServiceException object when an error occurs:

Argument	Description
operation	String for the type of operation (Retrieve, Update, Insert, Delete, Connect, or Disconnect)
rowNumber	Int32 for the row number or 0 if not applicable, such as when an error occurs during connection to the Web service
buffername	String for the name of the buffer being accessed while the error occurred (Primary, Filter, or Delete)
assembly	String for the name of the assembly being used
method	String for the name of the Web service method invoked
returnCode	Int32 for the return code from the Web service

CHAPTER 6

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

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 149.

❖ **To add a column from the data source to a DataWindow object:**

1 Select Column from the DataWindow Painter section of the Visual Studio toolbox.

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.

Insert columns instead of copying them

If you want to add a column from the DataWindow definition to a DataWindow, always use Insert>Control>Column. You might see unexpected results if you copy a column from one DataWindow object to another if they both reference the same column but the column order is different. This is because copying a column copies a reference to the column's id in the DataWindow definition.

Suppose d_first and d_second both have first_name and last_name columns, but first_name is column 1 in d_first and column 2 in d_second. If you delete the first_name column in d_second and paste column 1 from d_first in its place, both columns in d_second display the last_name column in the Preview view, because both columns now have a column id of 1.

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 DataWindow painter. You can add text anywhere you want to make the DataWindow object easier to understand.

❖ **To add text to a DataWindow object:**

1 Select Text from the DataWindow Painter section of the Visual Studio toolbox.

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 in the General category in the Properties window.

4 (Optional) Change the font, size, style, and alignment for the text in the Appearance category in the Properties window.

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 DataWindow Painter section of the Visual Studio toolbox.

2 Click where you want the control to display.

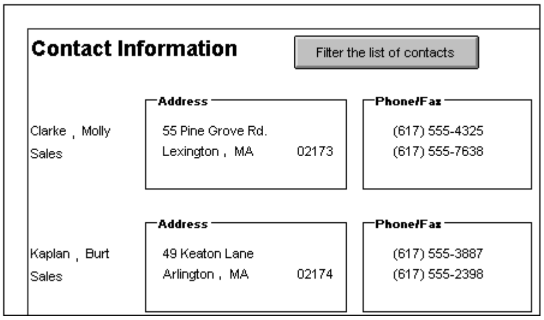
3 Resize or move the drawing control as needed.

4 Use the drawing control's Properties window 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

Filter the list of contacts

Clarke , Molly
Sales
Address
55 Pine Grove Rd.
Lexington , MA 02173
Phone/Fax
(617) 555-4325
(617) 555-7638

Kaplan , Burt
Sales
Address
49 Keaton Lane
Arlington , MA 02174
Phone/Fax
(617) 555-3887
(617) 555-2398

❖ **To add a group box to a DataWindow object:**

1 Select GroupBox from the DataWindow Painter section of the Visual Studio toolbox.

2 Click where you want the control to display.

3 With the group box selected, type the text to display in the frame in the General category in the Properties window.

4 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 Picture from the DataWindow Painter section of the Visual Studio toolbox.

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 image in its original size.

   You can use the mouse to change the size of the image in the DataWindow painter.

5 Set the Invert property in the Appearance category in the Properties window 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 33.

To compute a picture name at runtime, use the Bitmap function in the expression defining a computed field. If you change the image 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 image.

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 SQL 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 SQL 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 SQL 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 Computed Field from the DataWindow Painter section of the Visual Studio toolbox.

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 6-1 shows some examples of computed fields.

Table 6-1: 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

To display	Enter this expression	In this band
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 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 Average, Count, or Sum from the DataWindow Painter section of the Visual Studio toolbox.

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 Today or Pagination from the DataWindow Painter section of the Visual Studio toolbox.

The same options are available at the bottom of the Controls drop-down toolbar on the PainterBar.

2 Click anywhere in the DataWindow object.

If you selected Today, DataWindow Designer inserts a computed field containing this expression: `Today()`. For Pagination, the computed field contains this expression: `'Page ' + page() + ' of ' + pageCount()`.

Adding buttons to a DataWindow object

Buttons make it easy to provide command button actions in a DataWindow object. No coding is required. The use of Button controls in the DataWindow object, ensures that actions appropriate to the DataWindow object are included in the object itself.

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 Button from the DataWindow Painter section of the Visual Studio toolbox.
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 in the Properties window.
4. 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 181.

5. If you want to add a picture to the button, set the DefaultPicture property to True or enter the name of the Picture file to display on the button in the FileName field.
6. If you want to suppress event processing when the button is clicked at runtime, set SuppressEventProcessing to True.

When this option has been selected for the button and the button is clicked at runtime, only the action assigned to the button and the Clicked event are executed. The ButtonClicking and the ButtonClicked events are not triggered.

What happens if Suppress Event Processing is off

If Suppress Event Processing is off and the button is clicked, the Clicked and ButtonClicking events are fired. Code in the ButtonClicking event (if any) is executed. Note that the Clicked event is executed before the ButtonClicking event.

- If the return code from the ButtonClicking event is 0, the action assigned to the button is executed and then the ButtonClicked event is executed.
- If the return code from the ButtonClicking event is 1, neither the action assigned to the button nor the ButtonClicked event are executed.

Controlling the display of buttons in print preview and in printed output

You can choose whether to display buttons in print preview or in printed output. See "Defining print specifications for a DataWindow object" on page 140.

Actions assignable to buttons in DataWindow objects

Table 6-2 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 6-2: 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

Action	Effect	Value	Action return code
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.
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

Action	Effect	Value	Action return code
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 14, “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 Microsoft Tablet PC Web site at http://msdn2.microsoft.com/library/ms950406.aspx.

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, drag the InkPicture icon from the Toolbox to the painter. 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 window includes a Definition category where you can view or change the column definitions.

Adding OLE controls to a DataWindow object

You can add the following to a DataWindow object:

- A column that contains a database binary large object (a blob object) using OLE 2.0
- OLE 2.0 objects

Adding reports to a DataWindow object

You can nest reports (nonupdatable DataWindow objects) in a DataWindow object.

For information on nesting reports, see Chapter 10, “Using Nested Reports.”

Reorganizing controls in a DataWindow object

You can 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 137.

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 131.

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 toolbar.

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 toolbar to align controls.

   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 toolbar to space controls.

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 toolbar to size controls.

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 expand the Position category.
2. Select the Slide options you want:

Option	Description
Slide Left	Select True to 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	None – Do not slide the column or control 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. Directly Above – Slide the column or control up if there is nothing *directly above it* in the row above.

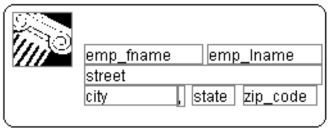
You can also use the toolbar to slide controls.

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.

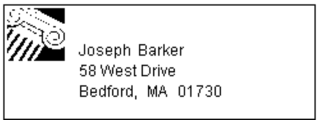
Example

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_lname, 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 6-3 shows the properties for each control in a DataWindow object that determine how it is positioned within the DataWindow object.

Table 6-3: 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.

Default positioning

DataWindow Designer uses the defaults shown in Table 6-4 when you place a new control in a DataWindow object.

Table 6-4: 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 and computed fields can be rotated from the original baseline of the text. The Escapement property 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.

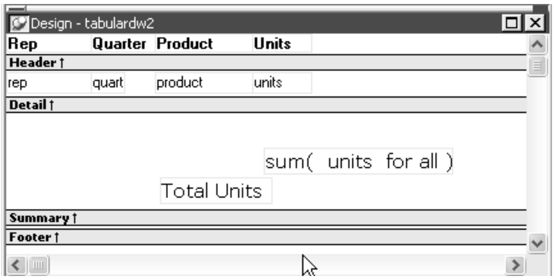
The following procedure includes design practices that help ensure that you get the final results you want. As you become more experienced, you can drop or alter some of the steps. The procedure recommends making the control movable in the Preview view, which is often helpful.

❖ **To rotate a control in a DataWindow object:**

1. Select the control in the Design view.
2. Make it movable by setting the Moveable property in the Layout category to True.
3. In Design view, enlarge the area in which the control is placed.

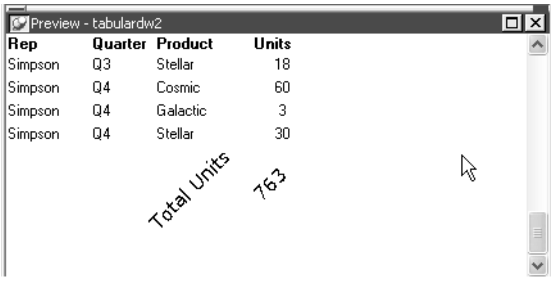
   For example, in a grid DataWindow object, make the band deeper and move the control down into the center of the band.
4. Display the Modify expression dialog box for the Escapement property. (Expand Font in the Appearance category and select the button next to the Escapement property.)
5. Specify the 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 because it makes it easier to position the control correctly. This example shows left-aligned text within two controls, a text control and a computed field.



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.



Rep	Quarter	Product	Units
Simpson	Q3	Stellar	18
Simpson	Q4	Cosmic	60
Simpson	Q4	Galactic	3
Simpson	Q4	Stellar	30

7 Drag and drop the control in the Design view until it is where you want it.

8 In the Design view, select the control that is being rotated 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.

CHAPTER 7 Displaying and Validating Data

About this chapter

This chapter describes how to customize your DataWindow object by modifying the display values in columns and specifying validation rules.

Contents

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 197.

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 208.

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 as well as a display format, then set the UseFormat property to True.

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 UseFormat property is not set (the default).

Validating data

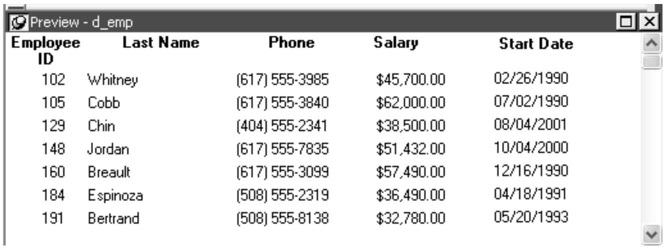
When data is entered in the Database painter or in a DataWindow object, DataWindow Designer evaluates the data against validation rules defined for that column. If the data is valid, DataWindow Designer accepts the entry; otherwise, DataWindow Designer displays an error message and does not accept the entry.

For more information, see “About validation rules” on page 225.

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 216.

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 set the UseFormat property.

For more information, see “About edit styles” on page 208.

❖ **To create a new display format:**

1 In the Database painter, open the Extended Attributes view, right-click Display Formats, and select Add from the pop-up menu.

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 201.

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 201.

❖ **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:**

- In the DataWindow painter, move the pointer to the column, select Properties from the column's pop-up menu, and then select Format in the Behavior category.

  Enter an appropriate format for the column. For more information, see "Defining display formats" on page 201.

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.

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 7-1 lists the blue, green, and red values you can use in the formula to create other colors.

Table 7-1: 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 7-2 lists characters that have special meaning in number display formats.

Table 7-2: 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. In a Web DataWindow, you must set ClientFormatting to true to display regional settings correctly.

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 7-3 shows how the values 5, –5, and .5 display when different format masks are applied.

Table 7-3: 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
$#,##0;($#,##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

Format	**5**	**-5**	**.5**
##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.

Special characters for string edit masks
String edit masks use different special characters. See "The EditMask edit style" on page 216.

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 7-4 shows characters that have special meaning in date display formats.

Table 7-4: 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 7-5 shows how the date Friday, January 30, 1998, displays when different format masks are applied.

Table 7-5: 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 7-6 shows characters that have special meaning in time display formats.

Table 7-6: 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 7-7 shows how the time 9:45:33:234567 PM displays when different format masks are applied.

Table 7-7: 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:ffffff	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.

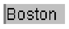
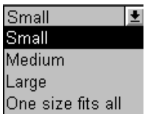
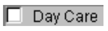
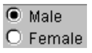
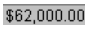
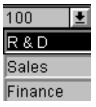
When edit styles are used

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 7-8 shows the available edit styles.

Table 7-8: 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, userscan 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

You work with edit styles in the Database painter and DataWindow painter.

What you do in the Database painter

In the Database painter, you can:

- Create, modify, and delete named edit styles

  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.

- Assign edit styles to columns

  These styles 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 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

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.

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.

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 View>Database Painter Layout>Extended Attributes. In the Extended Attributes view, select Display Formats and click New from the pop-up menu.
2. In the Object Details view, select the edit style type from the Style drop-down list.
3. Specify the properties of the edit style.

   For information, see "Defining edit styles" on page 212.

   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 Object Details view, modify the edit style as desired and click OK.

   For information, see "Defining edit styles" on page 212.

   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 category in the Properties window.
2. Select a style for the appropriate datatype from the list in the Style Name property.

   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 property.

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 Behavior category.

2 Select the type of edit style you want from the EditStyle drop-down list.

The information in the Behavior category changes to be appropriate to the type of edit style you selected.

3 Do one of the following:

- Select an edit style from the EditName list.
- 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 EditStyle drop-down list, expand the Edit property, 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, set Password to True.
- To allow users to tab to the column but not change the value, set DisplayOnly to True.
- To define a code table to determine which values are displayed to users and which values are stored in the database, set CodeTable to True and click the ellipsis button next to the Values property to enter display and data values for the code table.

  See "Defining a code table" on page 221.

❖ **To use the Edit edit style:**

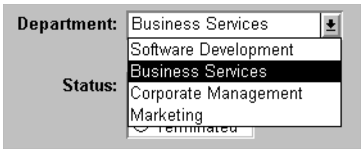
1 Select Edit from the EditStyle list, if it is not already selected.

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 EditStyle list and expand the DDLB property.

2 Select the appropriate properties.

3 Click the ellipsis button next to the Values property to open the Code Table dialog box.

4 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 221.

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:

❖ **To use the CheckBox edit style:**

1 Select CheckBox from the EditStyle list and specify properties for that style.

2 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.

---

3 In the CheckBox On and Off fields, enter the values you want put in the DataWindow buffer when the CheckBox is checked (on) or unchecked (off).

If you set 3 States to True, 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.

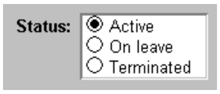
When users check or clear the check box at runtime, DataWindow .NET enters the appropriate data value in its buffer. When the UpdateData method is called, DataWindow .NET sends the corresponding data values to the database.

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. Set LeftText to False and clear the Text field, then specify Alignment>Center in the General category.

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 Select RadioButtons from the EditStyle list and specify properties for that style.

2 Specify how many radio buttons will display in the Columns box.

3 Click the ellipsis button next to the Values property to 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.

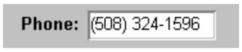
Using accelerator keys
To use an accelerator key on a radio button, enter an ampersand (&) in the Display Value before the letter that will be the accelerator key.

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:

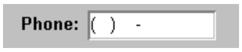


Edit masks consist of special characters that determine what can be entered in the column. They can also contain punctuation characters to aid users.

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

Most 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 201.

The special characters you can use in string edit masks are different from those you can use in string display formats.

Table 7-9: Special characters for string edit masks

Character	Meaning
!	Uppercase – displays all characters with letters in uppercase
^	Lowercase – displays all characters with letters in lowercase
#	Number – displays only numbers
a	Alphanumeric – displays only letters and numbers
X	Any character – displays all characters

If you use the "#" or "a" special characters in a mask, Unicode characters, spaces, and other characters that are not alphanumeric do not display.

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
```

- 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.
- The strings 00/00/00 or 00/00/0000 are interpreted as the `NULL` value for the column.

Using masks with "as is" characters

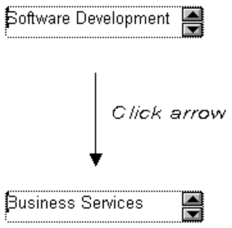
You can define a mask that contains "as is" characters that always appear in the control or column. For example, you might define a numeric mask such as `Rs0000.00` to represent Indian rupees in a currency column.

However, you cannot enter a minus sign to represent negative numbers in a mask that contains "as is" characters, and the # special character is treated as a 0 character. As a result, if you specify a mask such as ###,##0.00EUR, a value such as 45,000 Euros would display with a leading zero: 045,000.00EUR. Note that you must always specify a mask that has enough characters to display all possible data values. If the mask does not have enough characters, for example if the mask is #,##0.00 and the value is 45000, the result is unpredictable.

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 221.

❖ **To use an EditMask edit style:**

1. Select EditMask from the EditStyle list if it is not already selected.
2. Define the mask in the EditMask_Mask property.
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 allows for separate selections of the calendar month, year, and date. This option can also be set in code, as in this example for the birth_date column:

```
dw1.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, select HTMLGenClientFormatting in the Web Generation category of the DataWindow Properties window.

The drop-down calendar option is supported in all Web DataWindow rendering formats (HTML, XHTML, and XML).

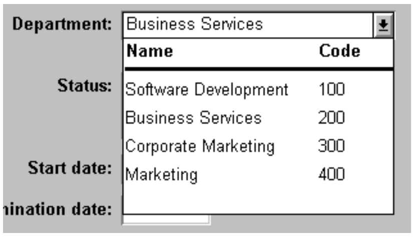
The DropDownDataWindow edit style

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:



Name	Code
Software Development	100
Business Services	200
Corporate Marketing	300
Marketing	400

❖ **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.

2 For the column in a second DataWindow getting its data from the d_dddw_dept DataWindow object, select the DropDownDW edit style.

3 Click the browse button next to the DDDW_Name property and select the DataWindow object that contains the data for the column from the list (in the example, d_dddw_dept). The list includes all the DataWindow objects in the current project.

4 In the Display Column property value, select the column containing the values that will display in the DataWindow object (in the example, dept_name).

5 In the Data Column property value, select the column containing the values that will be stored in the database (in the example, dept_id).

6 Specify other properties for the edit style.

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 in the Properties window 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://msdn2.microsoft.com/library/ms950406.aspx.

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 208.

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. Set the CodeTable property to true.
2. Click the browse button next to the Values property and 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, set the ValidateCode property to true.

   For more information, see "Validating user input" on page 224.

❖ **To define a code table as a property of the DropDownListBox edit style:**

1. Click the browse button next to the Values property and 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, set the AllowEdit property to false.

For more information, see “Validating user input” on page 224.

❖ **To define a code table as a property of the RadioButtons edit style:**

- Click the browse button next to the Values property and enter the display and data values for the code table.

❖ **To define a code table as a property of the DropDownDataWindow edit style:**

1. Click the browse button next to the DDDW_Name property and select the DataWindow object that contains the data for the column from the list.
2. In the Display Column property value, select the column containing the values that will display in the DataWindow object (in the example, dept_name).
3. In the Data Column property value, select the column containing the values that will be stored in the database (in the example, dept_id).
4. Specify the column that provides the display values in the Display Column box.
5. Specify the column that provides the data values in the Data Column box.
6. If you want to restrict input in the column to values in the code table, set the AllowEdit property to false.

❖ **To define a code table as a property of the EditMask edit style:**

1. Set the SpinControl property to true.
2. Set the CodeTable property to true.
3. Click the browse button next to the Values property and 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 7-10.

Table 7-10: 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, set the ValidateCode property to true.

If you have requested validation 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, set AllowEdit to false: users cannot type a value.

  Although users cannot type a value when Allow Edit is false, 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 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 208.

Understanding validation rules

Validation rules are criteria that a DataWindow object uses to validate data entered into a column by users. They are DataWindow .NET-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 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 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 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 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 Extended Attributes view in the Database painter, right-click Validation Rules and select New from the pop-up menu.

   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 230.

3. Click the Definition tab and define the expression for the rule.

   For information, see “Defining the expression” on page 228.

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. Double-click the validation rule you want to modify.
4. In the Validation Rule view, modify the validation rule as desired.

   For information, see "Defining the expression" on page 228 and "Customizing the error message" on page 230.

❖ **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 painter

If you are working in the DataWindow 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.

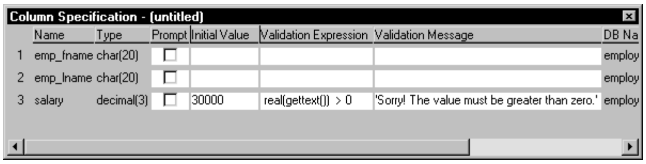
Defining a validation rule in the DataWindow painter

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 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>DataWindow Painter Layout>Column Specifications from the menu bar.

The Column Specification view displays.



Column Specification - (untitled)

	Name	Type	Prompt	Initial Value	Validation Expression	Validation Message	DB Na
1	emp_fname	char(20)	☐				employ
2	emp_lname	char(20)	☐				employ
3	salary	decimal(3)	☐	30000	real(gettext()) > 0	'Sorry! The value must be greater than zero.'	employ

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 230.

4 (Optional) Enter an initial value.

Used for current column 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>Database Painter Layout>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.

CHAPTER 8

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

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.

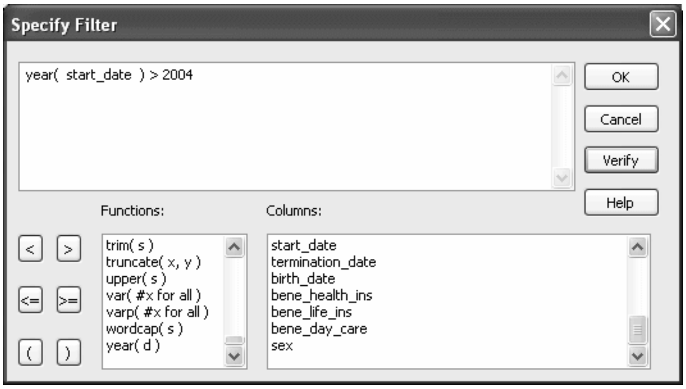
Filters do not affect which rows are retrieved
A filter operates against the retrieved data. It does not re-execute the SELECT statement.

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:



```
year( start_date ) > 2004
```

Functions:
trim(s)
truncate(x, y)
upper(s)
var(#x for all)
varp(#x for all)
wordcap(s)
year(d)

Columns:
start_date
termination_date
birth_date
bene_health_ins
bene_life_ins
bene_day_care
sex

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.

For information about expressions for filters, see the *DataWindow Object Reference*.

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 8-1 shows some examples of filters you might use.

Table 8-1: 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 dynamically

You can use the SetFilter and Filter methods in code to dynamically modify a filter that was set in the DataWindow painter. For information about SetFilter and Filter, see the online help.

Sorting rows

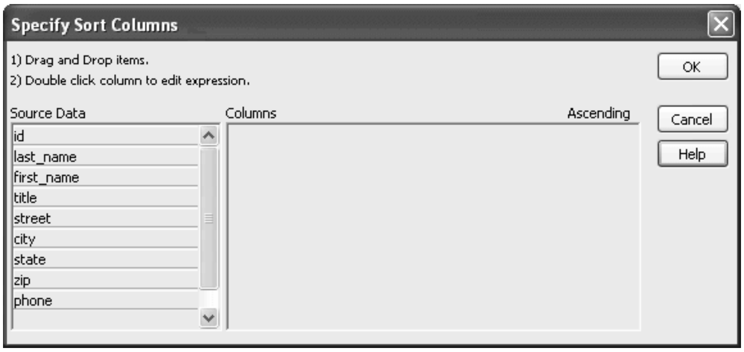
You can use an ORDER BY clause in the SQL SELECT statement for the DataWindow object to sort the data that is retrieved from the database. If you do this, the DBMS itself does the sorting and the rows are brought into DataWindow .NET already sorted.

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 .NET

❖ **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.

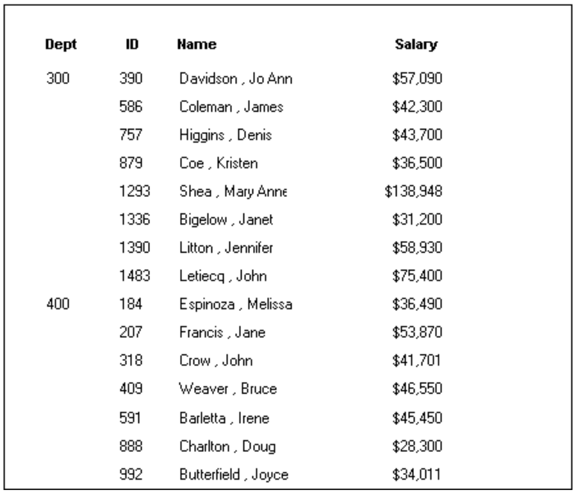
4 Click OK when you have specified all the sort columns and expressions.

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:

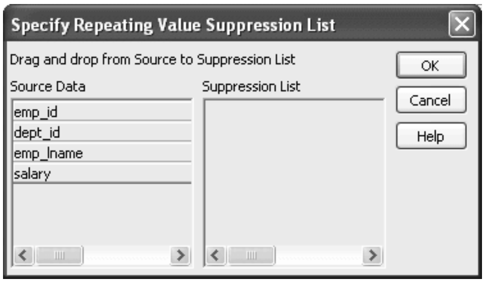


Dept	ID	Name	Salary
300	390	Davidson , Jo Ann	$57,090
	586	Coleman , James	$42,300
	757	Higgins , Denis	$43,700
	879	Coe , Kristen	$36,500
	1293	Shea , Mary Anne	$138,948
	1336	Bigelow , Janet	$31,200
	1390	Litton , Jennifer	$58,930
	1483	Letiecq , John	$75,400
400	184	Espinoza , Melissa	$36,490
	207	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:


Specify Repeating Value Suppression List
Drag and drop from Source to Suppression List
Source Data
Suppression List
emp_id
dept_id
emp_lname
salary
OK
Cancel
Help

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

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.

How groups are defined

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:

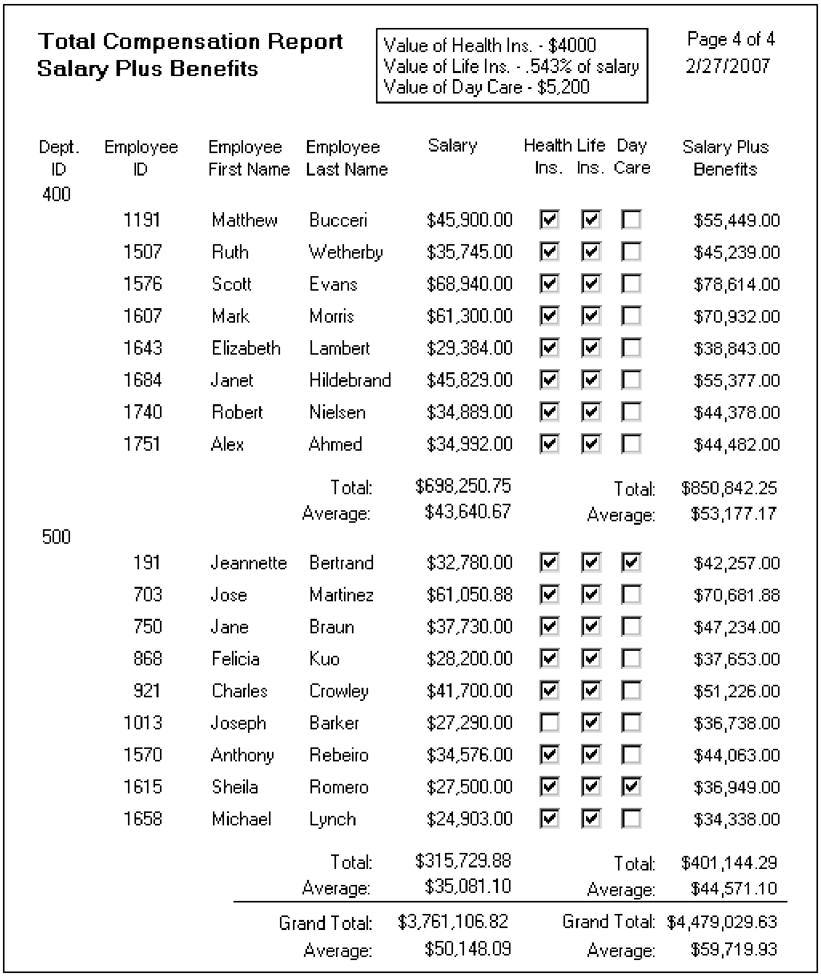
- Display the rows in each section
- Specify the information you want to display at the beginning and end of each section
- Specify page breaks after each break in the data
- Reset the page number after each break

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
Salary Plus Benefits

Value of Health Ins. - $4000
Value of Life Ins. - .543% of salary
Value of Day Care - $5,200

Page 4 of 4
2/27/2007

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	✓	✓	☐	$55,449.00
	1507	Ruth	Wetherby	$35,745.00	✓	✓	☐	$45,239.00
	1576	Scott	Evans	$68,940.00	✓	✓	☐	$78,614.00
	1607	Mark	Morris	$61,300.00	✓	✓	☐	$70,932.00
	1643	Elizabeth	Lambert	$29,384.00	✓	✓	☐	$38,843.00
	1684	Janet	Hildebrand	$45,829.00	✓	✓	☐	$55,377.00
	1740	Robert	Nielsen	$34,889.00	✓	✓	☐	$44,378.00
	1751	Alex	Ahmed	$34,992.00	✓	✓	☐	$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	✓	✓	✓	$42,257.00
	703	Jose	Martinez	$61,050.88	✓	✓	☐	$70,681.88
	750	Jane	Braun	$37,730.00	✓	✓	☐	$47,234.00
	868	Felicia	Kuo	$28,200.00	✓	✓	☐	$37,653.00
	921	Charles	Crowley	$41,700.00	✓	✓	☐	$51,226.00
	1013	Joseph	Barker	$27,290.00	☐	✓	☐	$36,738.00
	1570	Anthony	Rebeiro	$34,576.00	✓	✓	☐	$44,063.00
	1615	Sheila	Romero	$27,500.00	✓	✓	✓	$36,949.00
	1658	Michael	Lynch	$24,903.00	✓	✓	☐	$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 246).
- Use the TreeView presentation style (Chapter 12, "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.

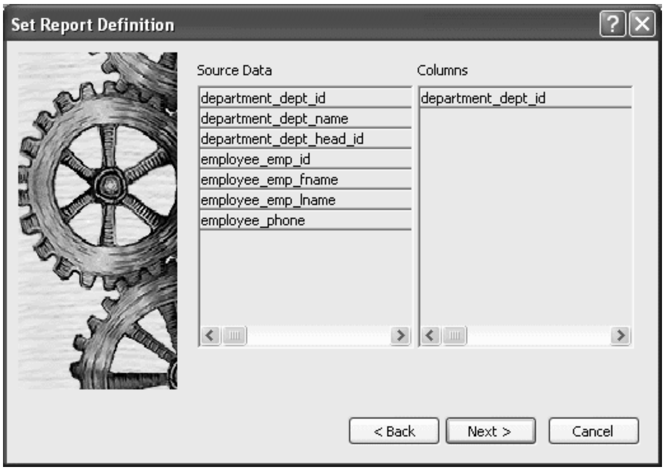
❖ **To create a basic grouped DataWindow object using the Group presentation style:**

1. In the Solution Explorer, right-click the library where you want to save the DataWindow object and select Add New Entry.
2. In the Add New Entry dialog box, select DataWindow Object from the categories list, select the Group presentation style, and click Add.
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:



Source Data: department_dept_id, department_dept_name, department_dept_head_id, employee_emp_id, employee_emp_fname, employee_emp_lname, employee_phone

Columns: department_dept_id

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 245.

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.

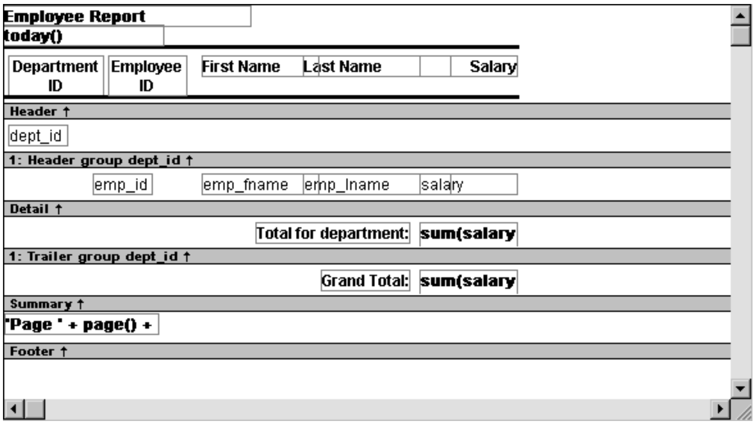
9 Click Next.

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:

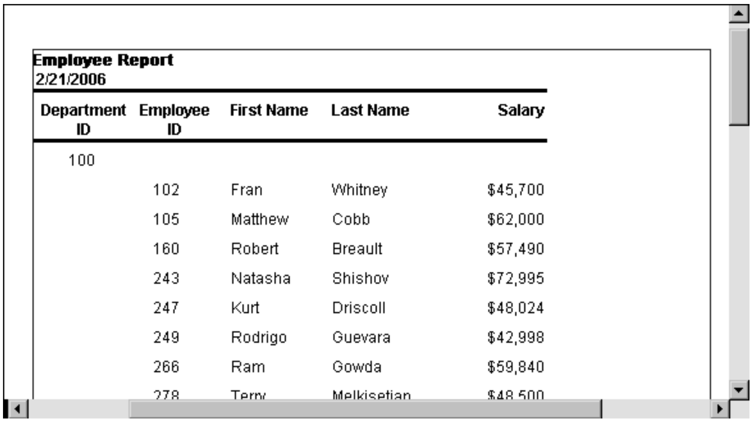


What DataWindow Designer does

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:



Employee Report
2/21/2006

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	Melkisetian	$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 GroupDefinition in the Properties window after you have finished using the Group wizard.

❖ **To use an expression for a group:**

1. Open the Properties window and select the group header band in the Design view.
2. Click the ellipsis button next to the GroupDefinition property in the General category 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 5, "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 242.
3. Set the ResetPageCount and NewPage properties.

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.

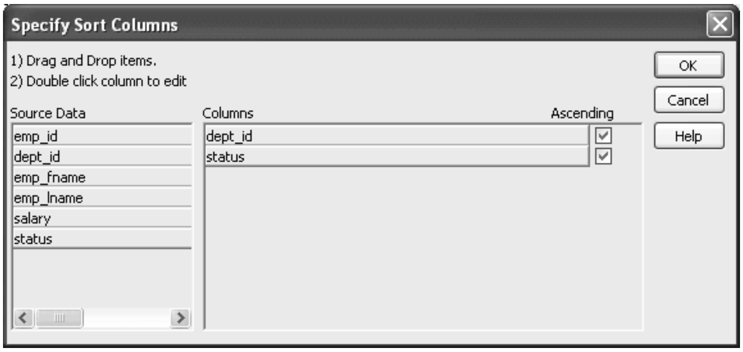
The rows are grouped first by department (group 1). Within department, rows are grouped by status (group 2). If you specify page breaks for the groups, a page break will occur when any of these values changes.

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 249.

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:



Columns	Ascending
dept_id	✓
status	✓

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 237.

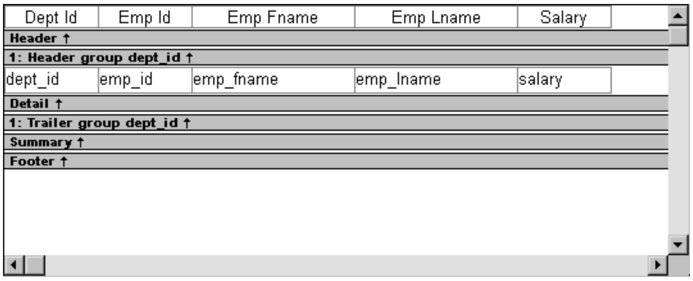
Rearranging the DataWindow object

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



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.

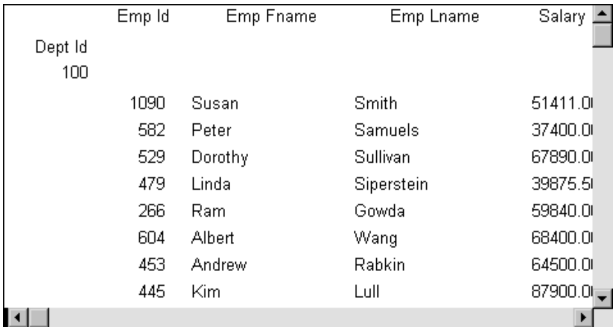
Using the group header band

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:



```
              Emp Id     Emp Fname         Emp Lname          Salary
  Dept Id
      100

                1090    Susan              Smith              51411.0
                 582    Peter              Samuels            37400.0
                 529    Dorothy            Sullivan           67890.0
                 479    Linda              Siperstein         39875.5
                 266    Ram                Gowda              59840.0
                 604    Albert             Wang               68400.0
                 453    Andrew             Rabkin             64500.0
                 445    Kim                Lull               87900.0
```

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 Suppress in the General category 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 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 .NET 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 Computed Field from the Visual Studio Toolbox.
2. Click in the Design view where you want the statistic.

   The Modify Expression dialog box displays.

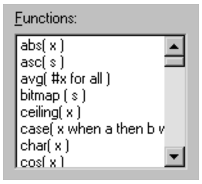
3 Specify the expression that defines the computed field (see below).

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 )
```

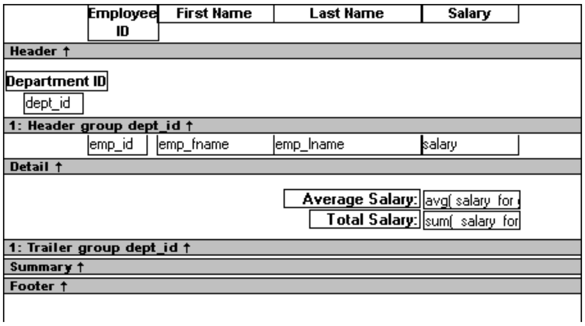
To get the average salary of employees in a department, specify:

```
Avg( Salary for group 1 )
```

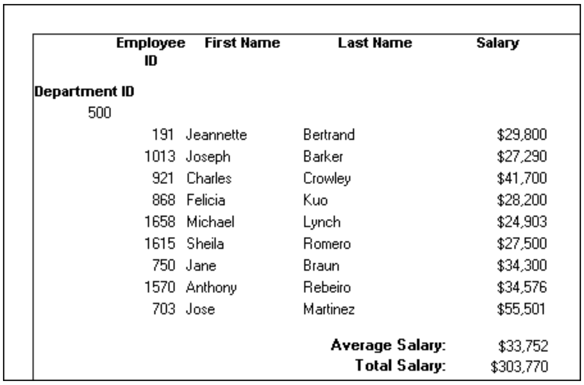
To get the total salary of employees in a department, specify:

```
Sum( Salary for group 1
```

The group trailer band in this example shows the average and total salary for the group.



At runtime, the average and total salaries are calculated and displayed:



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 249. 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.

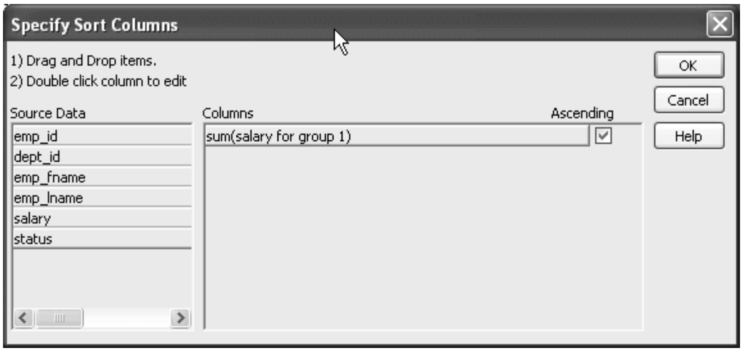
3 Click the ellipsis button next to the Group Sort property.

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.

CHAPTER 9

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

Highlighting information

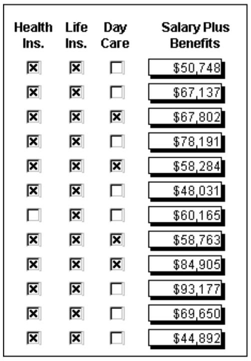
Every control in a DataWindow object has a set of properties that determines what the control looks like and where it is located. For example, the values in a column of data display in a particular font and color, in a particular location, with or without a border, and so on.

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:



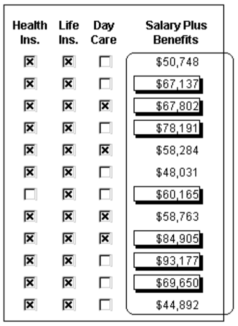
Health Ins.	Life Ins.	Day Care	Salary Plus Benefits
☒	☒	☐	$50,748
☒	☒	☐	$67,137
☒	☒	☒	$67,802
☒	☒	☐	$78,191
☒	☒	☒	$58,284
☒	☒	☐	$48,031
☐	☒	☐	$60,165
☒	☒	☒	$58,763
☒	☒	☒	$84,905
☒	☒	☐	$93,177
☒	☒	☐	$69,650
☒	☒	☐	$44,892

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
☒	☒	☐	$50,748
☒	☒	☐	$67,137
☒	☒	☒	$67,802
☒	☒	☐	$78,191
☒	☒	☒	$58,284
☒	☒	☐	$48,031
☐	☒	☐	$60,165
☒	☒	☒	$58,763
☒	☒	☒	$84,905
☒	☒	☐	$93,177
☒	☒	☐	$69,650
☒	☒	☐	$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).

A 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 .NET 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 .NET displays the value with the Shadow box border. If not (false), DataWindow .NET 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 window, 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.

For details on the values of properties that can be set using expressions, see "Supplying property values" on page 257.

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

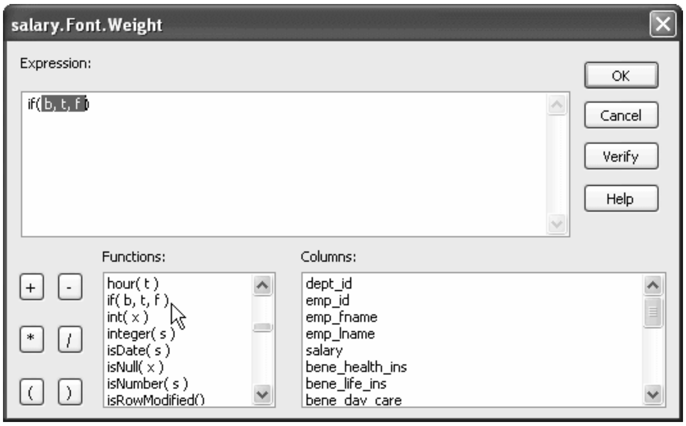
You can also programmatically modify the properties of controls in a DataWindow object at runtime. For more information, see the *Programmer's Guide*.

Modifying properties conditionally at runtime

"Modifying properties at runtime" on page 254 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 property you want to modify at runtime.
3. Click the button next to the property you want to change and select <Expression...>.
4. Scroll the list of functions in the Functions box until you see the IF function, and then select it:



```
if( b, t, f )
```

Functions:
hour(t)
if(b, t, f)
int(x)
integer(s)
isDate(s)
isNull(x)
isNumber(s)
isRowModified()

Columns:
dept_id
emp_id
emp_fname
emp_lname
salary
bene_health_ins
bene_life_ins
bene_day_care

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 257.

Set Font.Weight property to 700 for bold
Font properties such as Italic, Strikethrough, and Underline take a boolean value, but to specify a value for bold, you use the Font.Weight property, which takes a range of values. For values and an example, see "Font.Weight" on page 266.

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.

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 9-1 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.

In the Properties window, property names display without dots and underscores, and properties with dots, such as font.escapement, display in a tree view. Expand the font node to set the escapement property.

Table 9-1: Properties for controls in the DataWindow painter

Property	Category	Description
Background.Color	Appearance	Background color of a control
Border	Appearance	Border of a control
Brush.Color	Appearance	Color of a graphic control

Property	Category	Description
Brush.Hatch	Appearance	Pattern used to fill a graphic control
Color	Appearance	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)	Appearance	Rotation of a control
Font.Height	Appearance	Height of text
Font.Italic	Appearance	Use of italic font for text
Font.Strikethrough	Appearance	Use of strikethrough for text
Font.Underline	Appearance	Use of underlining for text
Font.Weight	Appearance	Weight (for example, bold) of text font
Format	Behavior	Display format for columns and computed fields
Height	Layout	Height of a control
Pen.Color	Appearance	Color of a line or the line surrounding a graphic control
Pen.Style	Appearance	Style of a line or the line surrounding a graphic control
Pen.Width	Appearance	Width of a line or the line surrounding a graphic control
Pointer	Appearance	Image to be used for the pointer
Protect	General	Whether a column can be edited
Timer_Interval	General	How often time fields are to be updated
Visible	General	Whether a control is visible
Width	Layout	Width of a control
X	Layout	X position of a control
X1, X2	Layout	X coordinates of either end of a line
Y	Layout	Y position of a control relative to the band in which it is located
Y1, Y2	Layout	Y coordinates of either end of a line

Background.Color

Description	Setting for the background color of a control.
Value	A number that specifies the control's background color.

For information on specifying colors, see “Specifying colors” on page 274.

The background color of a line is the color that displays between the segments of the line when the pen style is not solid.

If Background.Mode is transparent (1), Background.Color is ignored.

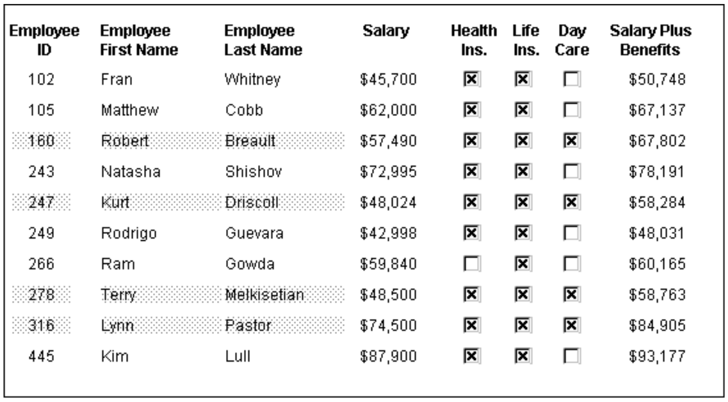
Example

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):

```
If(bene_day_care = 'Y', 15790320, 16777215)
```

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.

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	☒	☒	☐	$50,748
105	Matthew	Cobb	$62,000	☒	☒	☐	$67,137
160	Robert	Breault	$57,490	☒	☒	☒	$67,802
243	Natasha	Shishov	$72,995	☒	☒	☐	$78,191
247	Kurt	Driscoll	$48,024	☒	☒	☒	$58,284
249	Rodrigo	Guevara	$42,998	☒	☒	☐	$48,031
266	Ram	Gowda	$59,840	☐	☒	☐	$60,165
278	Terry	Melkisetian	$48,500	☒	☒	☒	$58,763
316	Lynn	Pastor	$74,500	☒	☒	☒	$84,905
445	Kim	Lull	$87,900	☒	☒	☐	$93,177

Border

Description

The type of border for the control.

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

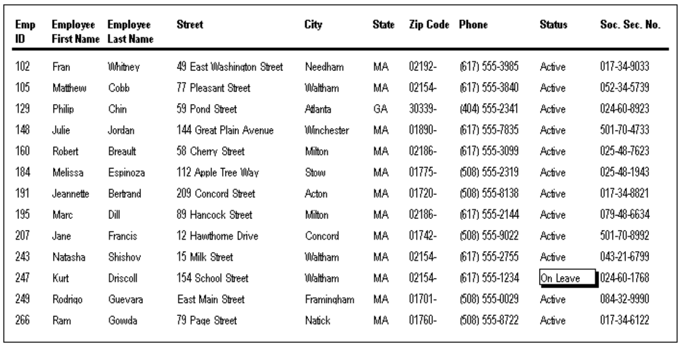
Example

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	Whitney	49 East Washington Street	Needham	MA	02192-	(617) 555-3985	Active	017-34-9033
105	Matthew	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	Espinoza	112 Apple Tree Way	Stow	MA	01775-	(508) 555-2319	Active	025-48-1943
191	Jeannette	Bertrand	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	Framingham	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.

Value

A number that specifies the color that fills the control.

For information on specifying colors, see “Specifying colors” on page 274.

Example

See the example for “Brush.Hatch” next.

Brush.Hatch

Description

Setting for the fill pattern of a graphic control.

Value

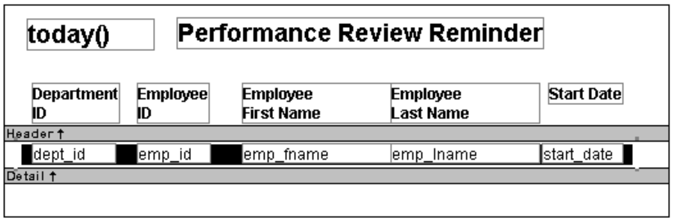
A number that specifies the pattern that fills the control. Values are:

`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 268.

Color

Description

The color of text for text controls, columns, and computed fields; background color for the DataWindow object; line color for graphs.

Value

A number that specifies the color used for text.

For information on specifying colors, see "Specifying colors" on page 274.

Example

The following statement is for the Color property of the emp_id, emp_fname, emp_lname, and emp_birth_date columns:

```
If(month(birth_date) = month (today()), 255, 0)
```

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.

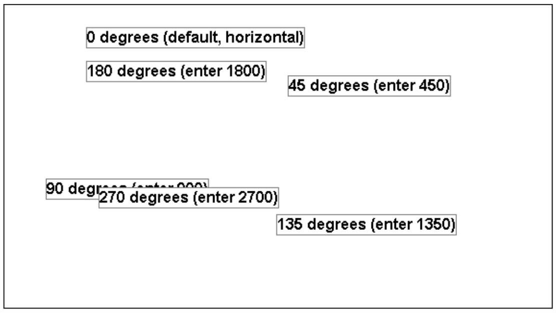
Value

An integer in tenths of degrees. For example, 450 means 45 degrees. 0 is horizontal.

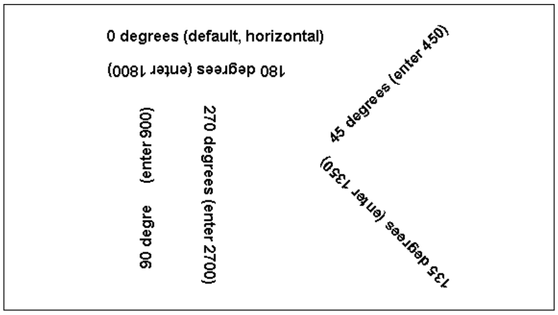
Example

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 window. 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.

Value

An integer in the unit of measure specified for the DataWindow object. Units of measure include Normalized 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

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):

```
If(GetRow() = 1, 500, 200)
```

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).

Font.Italic

Description

A number that specifies whether the text should be italic.

Value

Values are:

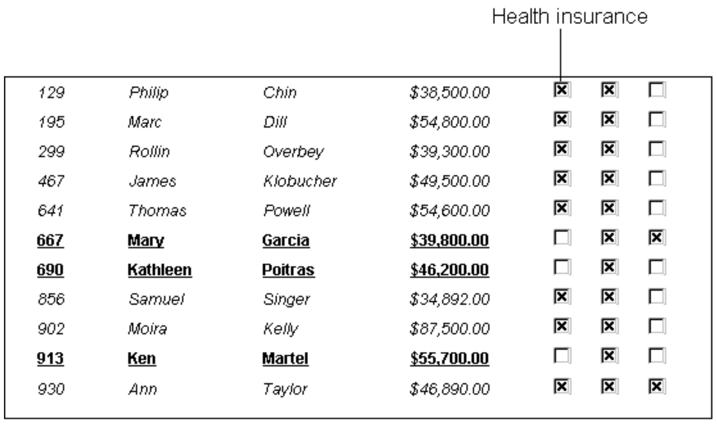
`0`—Not italic
`1`—Italic

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:



129	*Philip*	*Chin*	*$38,500.00*	☒	☒	☐
195	*Marc*	*Dill*	*$54,800.00*	☒	☒	☐
299	*Rollin*	*Overbey*	*$39,300.00*	☒	☒	☐
467	*James*	*Klobucher*	*$49,500.00*	☒	☒	☐
641	*Thomas*	*Powell*	*$54,600.00*	☒	☒	☐
667	**Mary**	**Garcia**	**$39,800.00**	☐	☒	☒
690	**Kathleen**	**Poitras**	**$46,200.00**	☐	☒	☐
856	*Samuel*	*Singer*	*$34,892.00*	☒	☒	☐
902	*Moira*	*Kelly*	*$87,500.00*	☒	☒	☐
913	**Ken**	**Martel**	**$55,700.00**	☐	☒	☐
930	*Ann*	*Taylor*	*$46,890.00*	☒	☒	☒

Font.Strikethrough

Description

A number that specifies whether the text should be crossed out.

Value

Values are:

- 0—Not crossed out
- 1—Crossed out

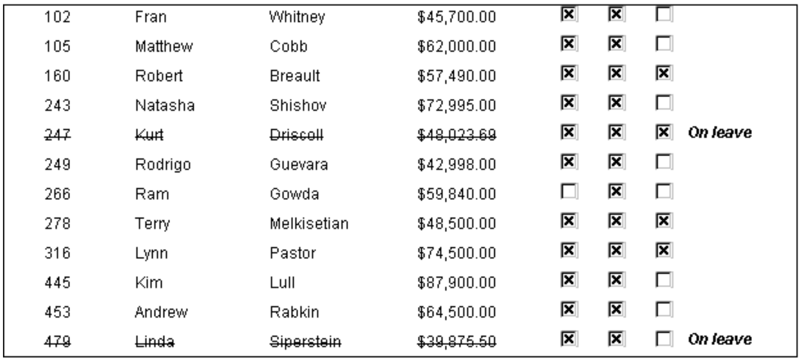
Example

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 271).

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	☒	☒	☐	
105	Matthew	Cobb	$62,000.00	☒	☒	☐	
160	Robert	Breault	$57,490.00	☒	☒	☒	
243	Natasha	Shishov	$72,995.00	☒	☒	☐	
~~247~~	~~Kurt~~	~~Driscoll~~	~~$48,023.69~~	☒	☒	☒	***On leave***
249	Rodrigo	Guevara	$42,998.00	☒	☒	☐	
266	Ram	Gowda	$59,840.00	☐	☒	☐	
278	Terry	Melkisetian	$48,500.00	☒	☒	☒	
316	Lynn	Pastor	$74,500.00	☒	☒	☒	
445	Kim	Lull	$87,900.00	☒	☒	☐	
453	Andrew	Rabkin	$64,500.00	☒	☒	☐	
~~479~~	~~Linda~~	~~Siperstein~~	~~$39,875.50~~	☒	☒	☐	***On leave***

Font.Underline

Description A number that specifies whether the text should be underlined.

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:

```
If(bene_health_ins = 'N', 1, 0)
```

For pictures of this example, see “Font.Italic” on page 264.

Font.Weight

Description The weight of the text.

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:

```
If(bene_health_ins = 'N', 700, 400)
```

For pictures of this example, see “Font.Italic” on page 264.

Format

Description

The display format for a column.

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 `Overpaid` for any salary greater than $60,000 and `Underpaid` 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.

Value

An integer in the unit of measure specified for the DataWindow object. Units of measure include PowerBuilder (normalized) 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 normalized units if the state column for the row has the value NY. Otherwise, the rectangle is 120 normalized units high:

```
if (state = 'NY', 160, 120)
```

Pen.Color

Description

The color of the line or the outline of a graphic control.

Value

A number that specifies the color of the line or outline.

For information on specifying colors, see “Specifying colors” on page 274.

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.

Value

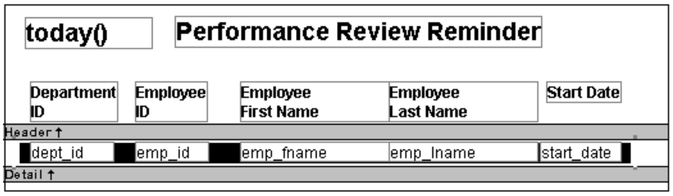
Values are:

`0`—Solid
`1`—Dash
`2`—Dotted
`3`—Dash-dot pattern
`4`—Dash-dot-dot pattern
`5`—Null (no visible line)

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 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)
```

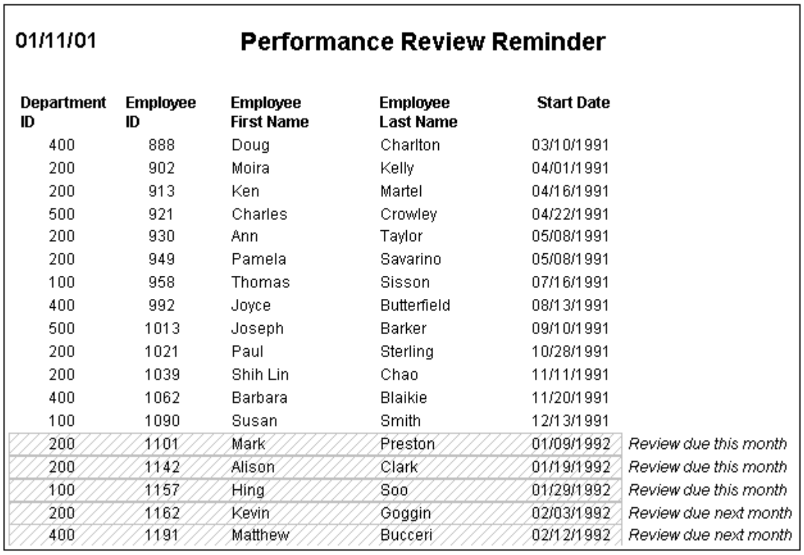
The following statement is for the Pen.Style property of the rectangle. If the month of the start date matches the current month or the next one, Pen.Style is set to Solid (0). If not, it is set to NULL (5), which means it will not show:

```
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 261.

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	*Review due this month*
200	1142	Alison	Clark	01/19/1992	*Review due this month*
100	1157	Hing	Soo	01/29/1992	*Review due this month*
200	1162	Kevin	Goggin	02/03/1992	*Review due next month*
400	1191	Matthew	Bucceri	02/12/1992	*Review due next month*

Pen.Width

Description

The width of the line or the outline of a graphic control.

Value

An integer in the unit of measure specified for the DataWindow object. Units of measure include Normalized units, thousandths of an inch (1000 = 1 inch), thousandths of a centimeter (1000 = 1 centimeter), or pixels.

Example

The following statement causes the width of a line to be 10 Normalized units if the state column for the row has the value NY. Otherwise, the line is 4 Normalized units wide:

```
If(state = 'NY', 10, 4)
```

Pointer

Description

The image used for the mouse pointer when the pointer is over the specified control.

Value

A string that specifies a value of the Pointer enumerated data type or the name of a cursor file (CUR) used for the pointer.

Values of the Pointer enumerated data type are:

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.

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.

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.

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 265.

Width

Description

The width of the control.

Value

An integer in the unit of measure specified for the DataWindow object. Units of measure include Normalized units, thousandths of an inch (1000 = 1 inch), thousandths of a centimeter (1000 = 1 centimeter), or pixels.

Example

The following statement causes the width of a rectangle to be 500 Normalized units if the state column for the row has the value NY. Otherwise, the rectangle is 1000 Normalized units wide:

```
if (state = 'NY', 500, 1000)
```

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.

Value

An integer in the unit of measure specified for the DataWindow object. Units of measure include Normalized 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)
```

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.

Value

Integers in the unit of measure specified for the DataWindow object. Units of measure include Normalized 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)
```

Y

Description

The distance of the control from the top of the band in which the control is located.

Value

An integer in the unit of measure specified for the DataWindow object. Units of measure include Normalized units, thousandths of an inch (1000 = 1 inch), thousandths of a centimeter (1000 = 1 centimeter), or pixels.

Y1, Y2

Description

The distance of each end of the specified line from the top of the band in which the line is located.

Value

Integers in the unit of measure specified for the DataWindow object. Units of measure include Normalized 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)
```

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 9-2 on page 274.

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 9-2 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 9-2: 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)

Color	Expression to enter	Number to enter	How the number is calculated
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)

CHAPTER 10 Using Nested Reports

About this chapter

This chapter provides information about creating reports that have other reports nested in them.

Contents

About reports and DataWindow objects

A report is the same as a nonupdatable DataWindow object.

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.

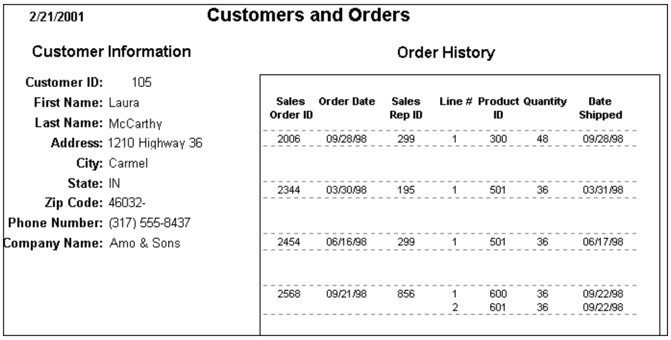
About placing a nested report within another report

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

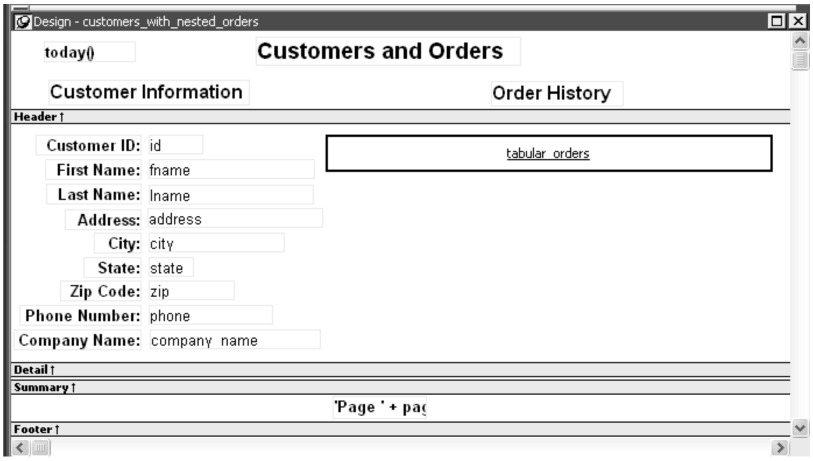
Customer ID:	105
First Name:	Laura
Last Name:	McCarthy
Address:	1210 Highway 36
City:	Carmel
State:	IN
Zip Code:	46032-
Phone Number:	(317) 555-8437
Company Name:	Amo & Sons

Order History

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	06/16/98	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:



Design - customers_with_nested_orders

today() Customers and Orders

Customer Information Order History

Header ↑

Customer ID:	id
First Name:	fname
Last Name:	lname
Address:	address
City:	city
State:	state
Zip Code:	zip
Phone Number:	phone
Company Name:	company_name

tabular_orders

Detail ↑

Summary ↑

'Page ' + pag

Footer ↑

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 287.

Limitation on nesting reports

For the most part you can nest the various types of report styles, but 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 In the Add New Entry dialog box, select the Composite DataWindow style.

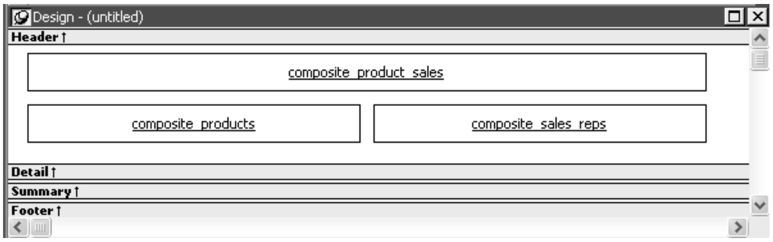
The wizard displays all reports (DataWindow objects) that are in the current project’s library search path.

2 Click the reports you want to include in the composite report and then click Next.

The wizard lists your choices.

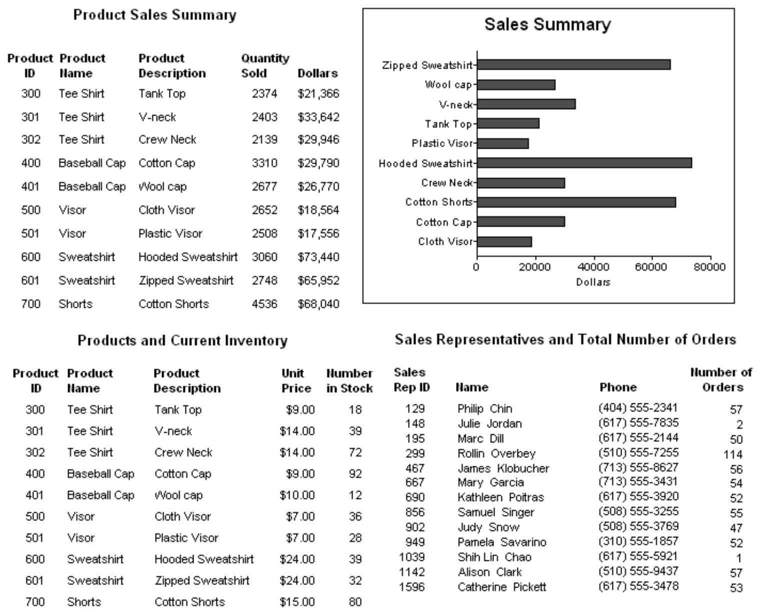
3 Click Finish.

DataWindow Designer places boxes for the selected reports in the Design view. In this example, you see three reports:



4 Select File>Save from the menu bar and assign a name to the composite report.

5 Look at the Preview view of the report:



Product Sales Summary

Product ID	Product Name	Product Description	Quantity Sold	Dollars
300	Tee Shirt	Tank Top	2374	$21,366
301	Tee Shirt	V-neck	2403	$33,642
302	Tee Shirt	Crew Neck	2139	$29,946
400	Baseball Cap	Cotton Cap	3310	$29,790
401	Baseball Cap	Wool cap	2677	$26,770
500	Visor	Cloth Visor	2652	$18,564
501	Visor	Plastic Visor	2508	$17,556
600	Sweatshirt	Hooded Sweatshirt	3060	$73,440
601	Sweatshirt	Zipped Sweatshirt	2748	$65,952
700	Shorts	Cotton Shorts	4536	$68,040

Products and Current Inventory

Product ID	Product Name	Product Description	Unit Price	Number in Stock
300	Tee Shirt	Tank Top	$9.00	18
301	Tee Shirt	V-neck	$14.00	39
302	Tee Shirt	Crew Neck	$14.00	72
400	Baseball Cap	Cotton Cap	$9.00	92
401	Baseball Cap	Wool cap	$10.00	12
500	Visor	Cloth Visor	$7.00	36
501	Visor	Plastic Visor	$7.00	28
600	Sweatshirt	Hooded Sweatshirt	$24.00	39
601	Sweatshirt	Zipped Sweatshirt	$24.00	32
700	Shorts	Cotton Shorts	$15.00	80

Sales Representatives and Total Number of Orders

Sales Rep ID	Name	Phone	Number of Orders
129	Philip Chin	(404) 555-2341	57
148	Julie Jordan	(617) 555-7835	2
195	Marc Dill	(617) 555-2144	50
299	Rollin Overbey	(510) 555-7255	114
467	James Klobucher	(713) 555-8627	56
667	Mary Garcia	(713) 555-3431	54
690	Kathleen Poitras	(617) 555-3920	52
856	Samuel Singer	(508) 555-3255	55
902	Judy Snow	(508) 555-3769	47
949	Pamela Savarino	(310) 555-1857	52
1039	Shih Lin Chao	(617) 555-5921	1
1142	Alison Clark	(510) 555-9437	57
1596	Catherine Pickett	(617) 555-3478	53

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.

6 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.

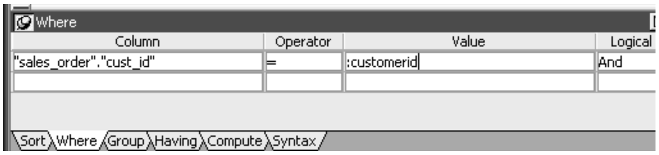
2 Define a retrieval argument for the nested report.

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 SQL Select painter.
- Select Design>Retrieval Arguments from the menu bar in the SQL 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*:

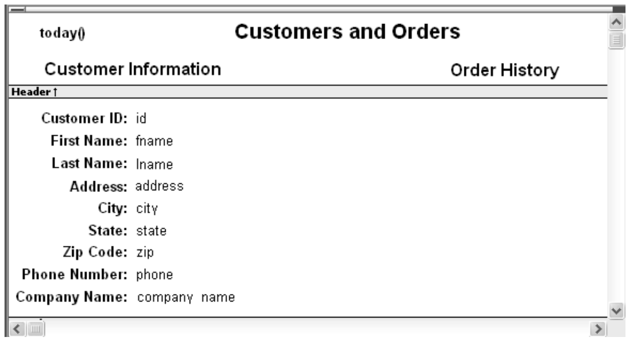


Column	Operator	Value	Logical
"sales_order"."cust_id"	=	:customerid	And

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:



today() Customers and Orders

Customer Information Order History

Customer ID:	id
First Name:	fname
Last Name:	lname
Address:	address
City:	city
State:	state
Zip Code:	zip
Phone Number:	phone
Company Name:	company_name

5 Select Report from the Visual Studio Toolbox.

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.

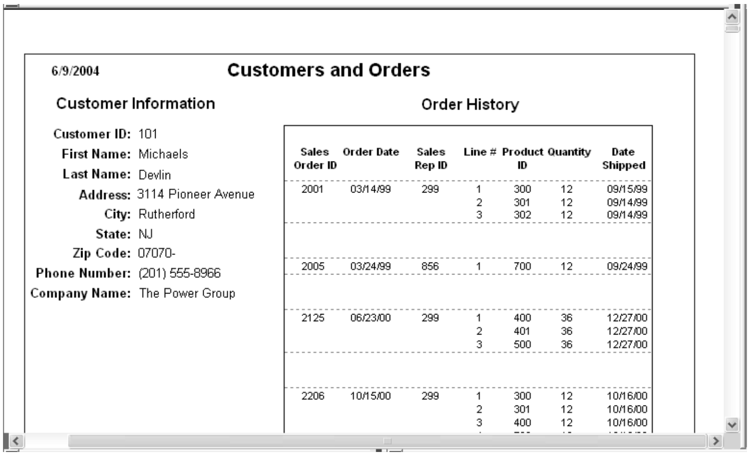
A box representing the report displays in the Design view.

8 With the report still selected, click the ellipsis button next to the Nest_Arguments property.

9 Supply the base report column or the expression that will supply the argument's value. To do this, double-click 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 In the Preview view, you can see what your report looks like:



6/9/2004

Customers and Orders

Customer Information

Customer ID:	101
First Name:	Michaels
Last Name:	Devlin
Address:	3114 Pioneer Avenue
City:	Rutherford
State:	NJ
Zip Code:	07070-
Phone Number:	(201) 555-8966
Company Name:	The Power Group

Order History

Sales Order ID	Order Date	Sales Rep ID	Line #	Product ID	Quantity	Date Shipped
2001	03/14/99	299	1	300	12	09/15/99
			2	301	12	09/14/99
			3	302	12	09/14/99
2005	03/24/99	856	1	700	12	09/24/99
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 Report from the Visual Studio Toolbox.

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.

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 window for it.

Many of the options in the Properties window are described in Chapter 4, “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 285
- “Modifying the definition of a nested report” on page 286
- “Adding another nested report to a composite report” on page 286
- “Supplying retrieval arguments to relate a nested report to its base report” on page 287
- “Specifying criteria to relate a nested report to its base report” on page 288
- “Using options for nested reports” on page 289

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 window, General category, click the button next to the DataObject property.

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 and then close it.

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 281 or “Placing an unrelated nested report in another report” on page 283.

❖ **To add another nested report to a composite report:**

1 Open the composite report.

2 Select Report from the Visual Studio Toolbox.

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 281.)

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 database parameter to 0, which is the default.
- You must enable caching in the database profile. Set the SQLCache database parameter to the number of levels of nesting plus 5.

  For more information, see the description of the SQLCache and DisableBind database 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 window for the composite report, select the General category. 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 281.

2 Select the report and click the ellipsis button next to the Nest_Arguments property.

3 Supply the base report column or the expression that will supply the argument's value. To do this, double-click 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 button next to the Criteria property in the Properties window.

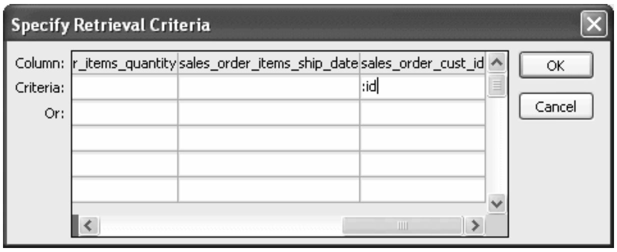
The Criteria property provides a way for you to specify how information from the base report will supply the retrieval criteria to the nested report.

2 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:



Column:	r_items_quantity	sales_order_items_ship_date	sales_order_cust_id
Criteria:			:id
Or:			

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 HeightAutoSize option

HeightAutoSize 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 a HeightAutoSize option. The option is off by default and must be on for the HeightAutoSize option for the nested report to work properly.

❖ **To change the HeightAutoSize option for a nested report:**

1. In the Design view, select the nested report.
2. In the Properties window, select the Position category.
3. Set the HeightAutosize property to True or False.

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 text column with an autosized height.

Using the Slide options

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 SlideLeft option is on by default for grid and crosstab style reports and off by default for all others. Having SlideLeft on for grid and crosstab ensures that these reports break horizontally on whole columns and not in the middle of a column.
- The SlideUp AllAbove and DirectlyAbove 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 189.

Using the NewPage option (composite only)

The NewPage 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 window, select the General category.
3. Set the NewPage property to True.

Using the TrailFooter 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 window, select the General category.
3. Set the TrailFooter property to False.

   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.

CHAPTER 11 Working with Crosstabs

About this chapter

This chapter describes how to build crosstabs.

Contents

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.

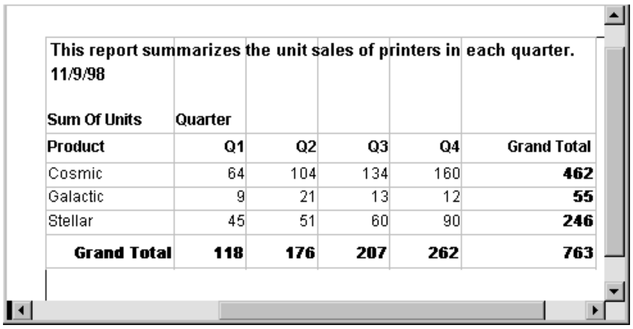
An example

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 14, “Working with Graphs.”)

Table 11-1: 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				
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. In the Solution Explorer, right-click the library where you want to save the DataWindow object and select Add New Entry.
2. In the Add New Entry dialog box, select DataWindow Object from the categories list and select the Crosstab DataWindow style, provide a name for the DataWindow object, and click Add.
3. On the Choose Data Source for Crosstab DataWindow page, specify the data you want retrieved into the DataWindow object.

   For more information, see Chapter 3, “Defining DataWindow Objects.”
4. In the Define Crosstab Rows, Columns, Values page, enter the definitions for the columns, rows, and cell values in the crosstab.

   See “Associating data with a crosstab” on page 296.
5. Click Next.
6. Choose Color and Border settings and click Next.
7. Review your specifications and click Finish.

   DataWindow Designer creates the crosstab.
8. (Optional) Specify other properties of the crosstab.

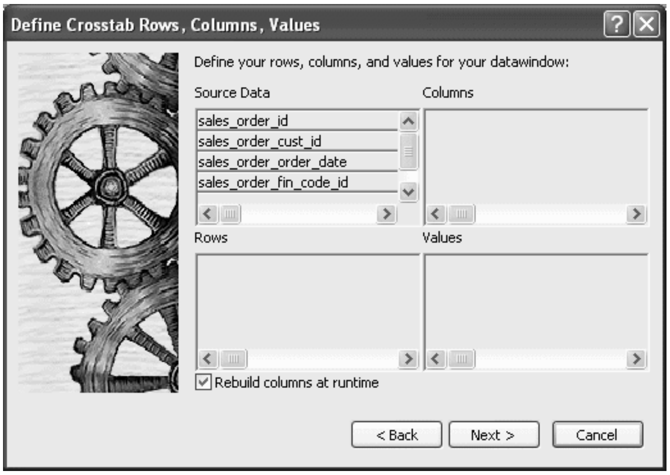
   See “Enhancing crosstabs” on page 302.

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 check box is selected.

For information about static crosstabs, see "Creating static crosstabs" on page 311.

4 Click Next.

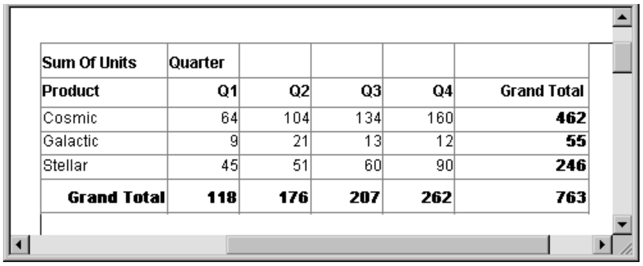
Specifying the information

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 7, "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.

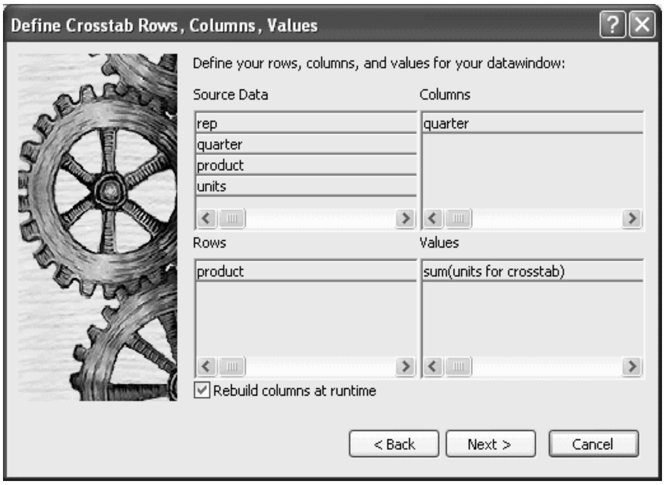
2. Specify the expression and click OK.

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 297, 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.


Define Crosstab Rows, Columns, Values
Define your rows, columns, and values for your datawindow:
Source Data
Columns
Rows
Values
< Back
Next >
Cancel

Source Data: rep, quarter, product, units

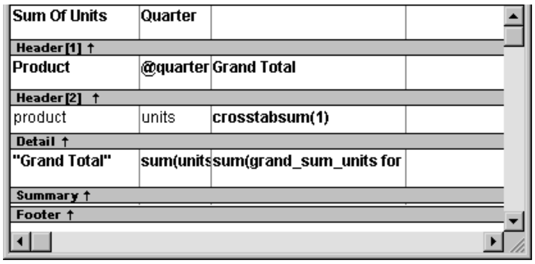
Columns: quarter

Rows: product

Values: sum(units for crosstab)

[✓] Rebuild columns at runtime

In the Design view, the crosstab looks like this:

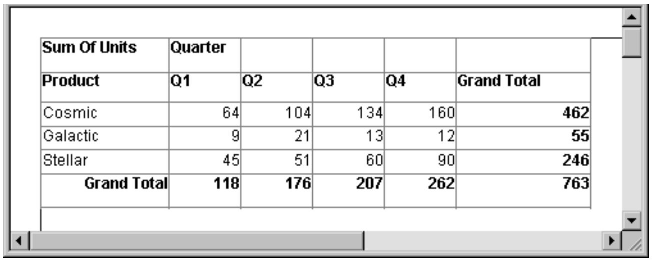


Sum Of Units	Quarter		
Header[1] ↑			
Product	@quarter	Grand Total	
Header[2] ↑			
product	units	crosstabsum(1)	
Detail ↑			
"Grand Total"	sum(units	sum(grand_sum_units for	
Summary ↑			
Footer ↑			

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**

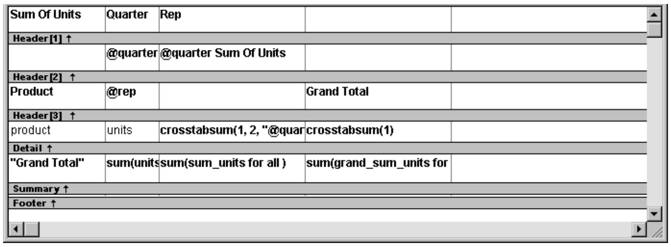
- Because quarter was selected as the Columns definition, there is one column in the crosstab for each unique quarter (Q1, Q2, Q3, and Q4).
- Because product was selected as the Rows definition, there is one row in the crosstab for each unique product (Cosmic, Galactic, and Stellar).
- Because `sum(units 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.

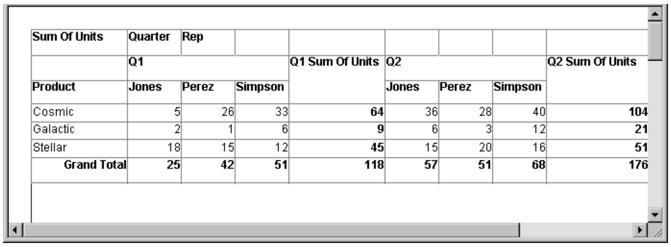
For example, consider a crosstab that has the same specification as the crosstab in “Viewing the crosstab” on page 299, except that two database columns, quarter and rep, have been dragged to the Columns box.

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(units	sum(sum_units for all)	sum(grand_sum_units for	
Summary ↑				
Footer ↑				

This is what you see at runtime:



Sum Of Units	Quarter	Rep						
	Q1			Q1 Sum Of Units	Q2			Q2 Sum Of Units
Product	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. If the Preview view is not open, select View>DataWindow Painter Layout>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 4, “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 304
- “Changing the names used for the columns and rows” on page 304
- “Defining summary statistics” on page 305
- “Cross-tabulating ranges of values” on page 308
- “Creating static crosstabs” on page 311
- “Using property conditional expressions” on page 312

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

Table 11-2 lists basic crosstab properties. These properties are in the General category in the Properties window.

Table 11-2: Basic properties for crosstabs

Option	Result
GridLines	*On* – Grid lines always display. *Off* – Grid lines never display (columns cannot be resized at runtime). *Display Only* – Grid lines display only when the crosstab displays online. *Print Only* – Grid lines display only when the contents of the crosstab are printed.
GridColumnMove	Columns can be moved at runtime.
SelectedMouse	Data can be selected at runtime (and, for example, copied to the clipboard).
RowResize	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 at any time 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 296.
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.

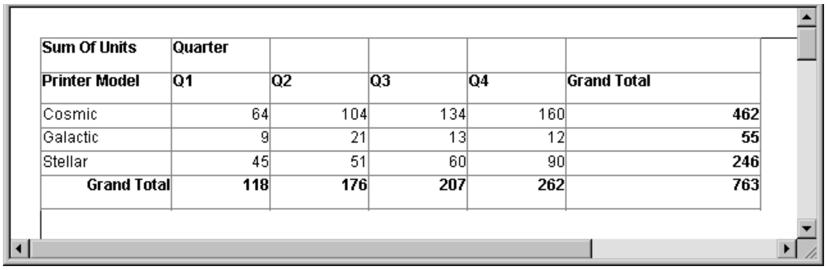
   The New Name dialog box displays:
2. Specify the name you want used to label the corresponding column. You can have multiple-word labels by using underscores: underscores are replaced by spaces in the Design view and at runtime.
3. Click OK.

   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 Computed Field from the Visual Studio Toolbox.

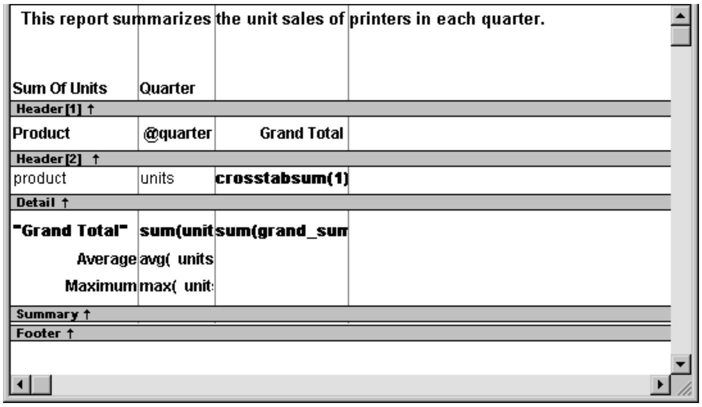
3 Click the cell in the summary band where you want the summary to display.

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.

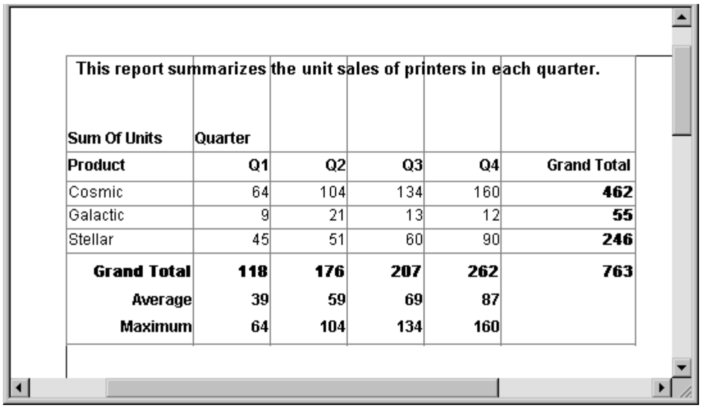
For example, this is a crosstab that has been enhanced to show averages and maximum values for each column. This is the Design view:



This report summarizes the unit sales of printers in each quarter.

Sum Of Units	Quarter	
Header[1] ↑		
Product	@quarter	Grand Total
Header[2] ↑		
product	units	**crosstabsum(1)**
Detail ↑		
"Grand Total"	**sum(unit**	**sum(grand_sum**
Average	avg(units	
Maximum	max(unit:	
Summary ↑		
Footer ↑		

This is the crosstab at runtime:



This report summarizes the unit sales of printers in each 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**
Average	39	59	69	87	
Maximum	64	104	134	160	

❖ **To define a row summary:**

1 Select Computed Field from the Visual Studio Toolbox.

2 Click the empty cell to the right of the last column in the detail band.

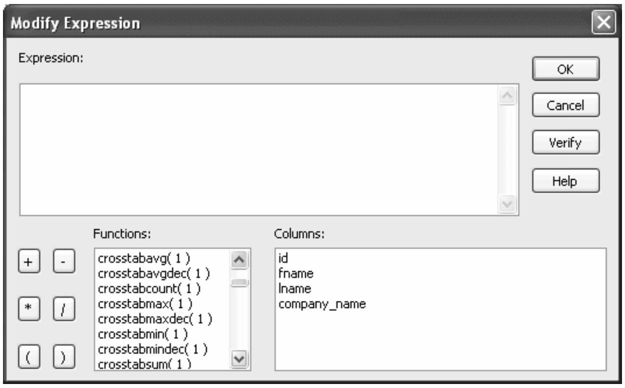
The Modify Expression dialog box displays.

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:



```
crosstabavg( 1 )
crosstabavgdec( 1 )
crosstabcount( 1 )
crosstabmax( 1 )
crosstabmaxdec( 1 )
crosstabmin( 1 )
crosstabmindec( 1 )
crosstabsum( 1 )
```

```
id
fname
lname
company_name
```

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 4, “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
   ```

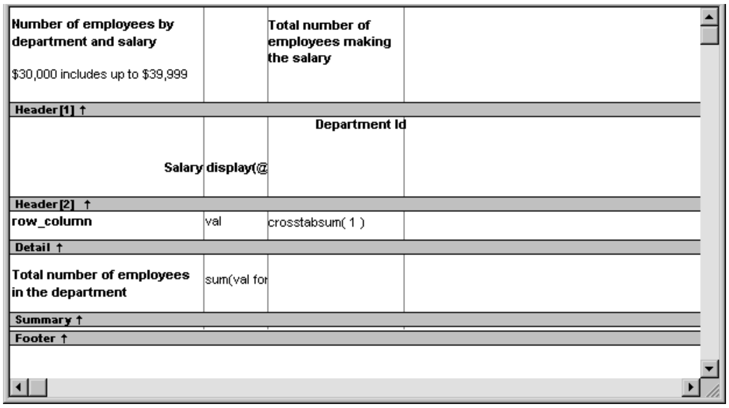
   In the Values box, use:

   ```
   count(int(salary/10000) * 10000 for crosstab)
   ```

   For more on providing expressions in a crosstab, see “Using expressions” on page 298.

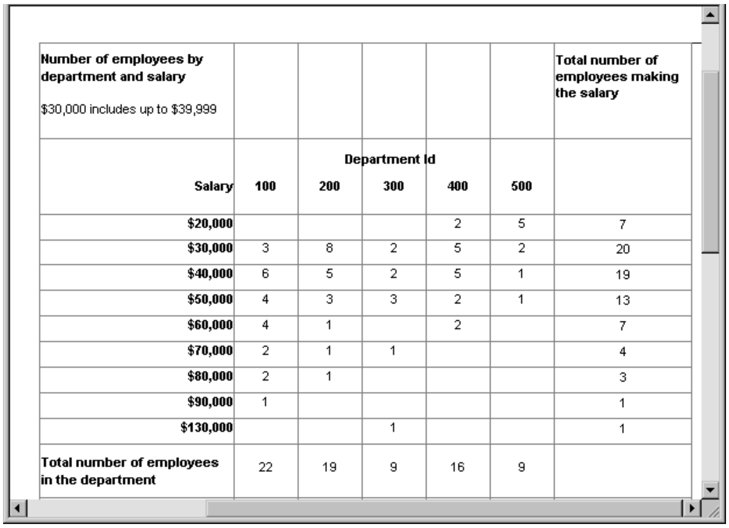
4 Click OK.

This is the result in the Design view:



Number of employees by department and salary $30,000 includes up to $39,999		**Total number of employees making the salary**	
Header[1] ↑			
Salary	**display(@**	**Department Id**	
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 $30,000 includes up to $39,999						**Total number of employees making the salary**
			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 expand the Format category in the Properties window.
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 .NET 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 .NET 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.

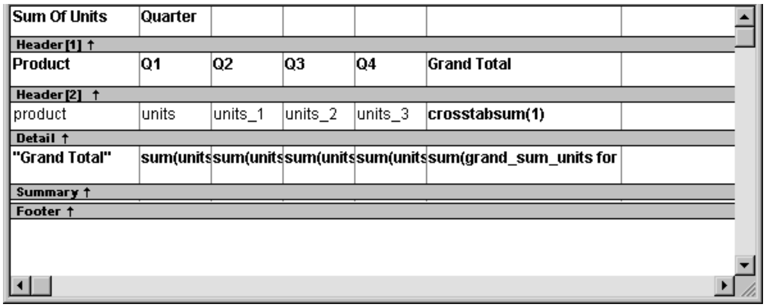
❖ **To create a static crosstab:**

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

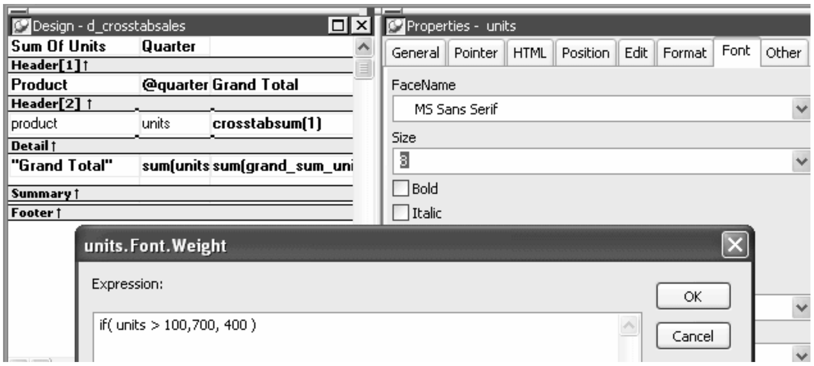
You can modify the properties of any of the columns in a static crosstab. You can modify the properties of each column individually, since each column is displayed in the workspace as part of the crosstab's definition. For example, in the printer crosstab you can directly modify the way values are presented 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:



Design - d_crosstabsales

Sum Of Units	Quarter	
Header[1]↑		
Product	@quarter	Grand Total
Header[2]↑		
product	units	crosstabsum(1)
Detail↑		
"Grand Total"	sum(units	sum(grand_sum_uni
Summary↑		
Footer↑		

Properties - units

FaceName: MS Sans Serif

Size: 8

☐ Bold

☐ Italic

units.Font.Weight

Expression:

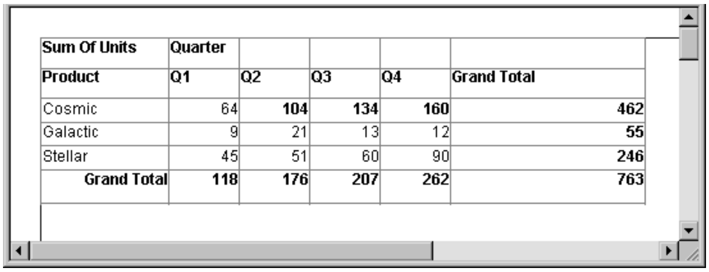
```
if( units > 100,700, 400 )
```

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 9, “Highlighting Information in DataWindow Objects.”

CHAPTER 12 Working with TreeViews

About this chapter

This chapter describes how to build and use DataWindow objects in DataWindow Designer using the TreeView presentation style.

Contents

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.

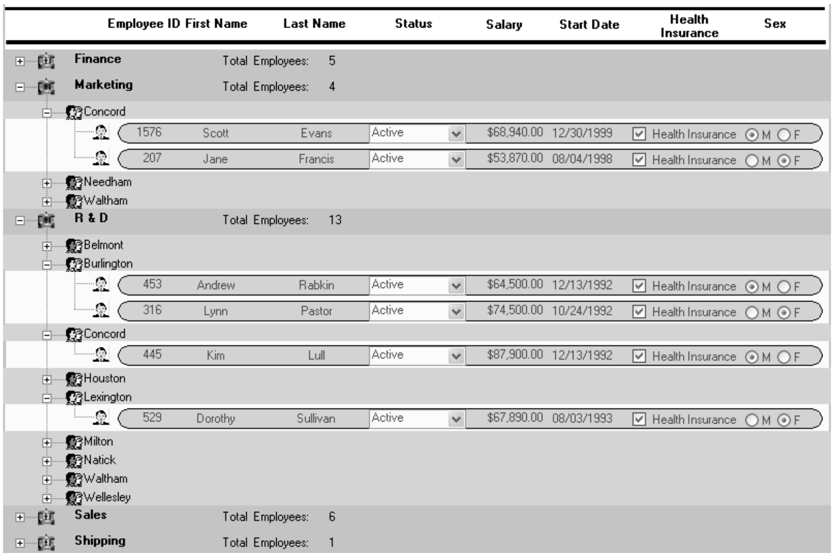
In the TreeView DataWindow, each parent node contains other nodes called child nodes. You can display parent nodes—nodes that contain child nodes—in expanded or collapsed form.

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 316.

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	✓ Health Insurance	[✓ M] F
	207	Jane	Francis	Active	$53,870.00	08/04/1998	✓ Health Insurance	M [✓ F]
⊞ Needham								
⊞ Waltham								
⊟ R & D	Total Employees: 13							
⊞ Belmont								
⊟ Burlington								
	453	Andrew	Rabkin	Active	$64,500.00	12/13/1992	✓ Health Insurance	[✓ M] F
	316	Lynn	Pastor	Active	$74,500.00	10/24/1992	✓ Health Insurance	M [✓ F]
⊟ Concord								
	445	Kim	Lull	Active	$87,900.00	12/13/1992	✓ Health Insurance	[✓ M] F
⊞ Houston								
⊟ Lexington								
	529	Dorothy	Sullivan	Active	$67,890.00	08/03/1993	✓ Health Insurance	M [✓ 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 321. For information about setting properties in the DataWindow painter, see “Setting properties for the TreeView DataWindow” on page 325.

You can use TreeView DataWindow methods to expand and collapse TreeView nodes, and you can write code for TreeView DataWindow events that are fired when a node is expanded or collapsed.

Creating a TreeView DataWindow

❖ **To create a TreeView DataWindow:**

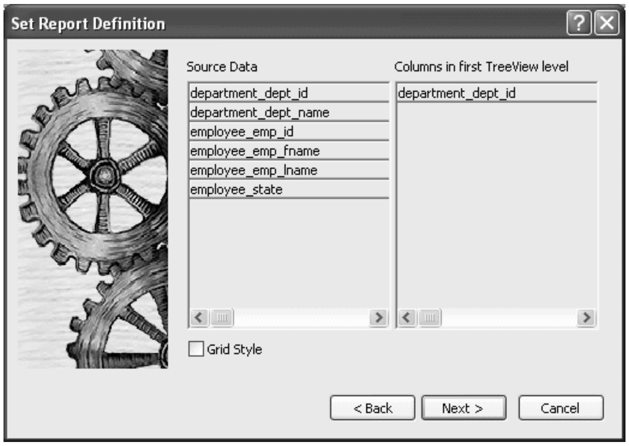
1. In the Solution Explorer, right-click the library where you want to save the DataWindow object and select Add New Entry.
2. In the Add New Entry dialog box, select DataWindow Object from the categories list and select the TreeView DataWindow style, provide a name for the DataWindow object, and click Add.
3. Select the data source you want to use.

   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:



department_dept_id
department_dept_name
employee_emp_id
employee_emp_fname
employee_emp_lname
employee_state

department_dept_id

Grid Style

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 320.

The sample DataWindow shown in "Example" on page 316 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.

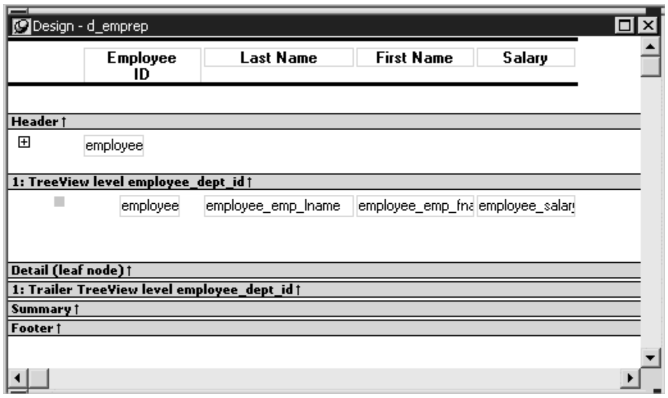
The DataWindow painter Design view displays. For information about the Design view, see "TreeView DataWindow Design view" on page 324. For information about adding additional levels, see "Adding and deleting TreeView levels" on page 321.

What DataWindow Designer does

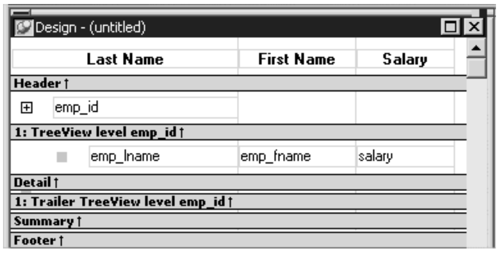
As a result of your specifications, DataWindow Designer generates a TreeView DataWindow object and creates:

- 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.

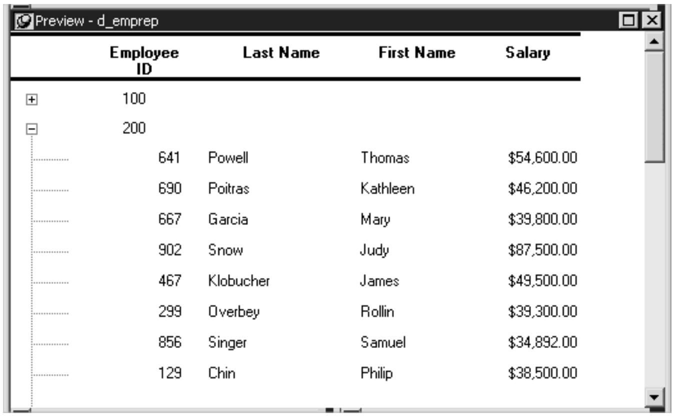
Here is the sample TreeView DataWindow object in the Design view:



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:



Employee ID	Last Name	First Name	Salary
100			
200			
641	Powell	Thomas	$54,600.00
690	Poitras	Kathleen	$46,200.00
667	Garcia	Mary	$39,800.00
902	Snow	Judy	$87,500.00
467	Klobucher	James	$49,500.00
299	Overbey	Rollin	$39,300.00
856	Singer	Samuel	$34,892.00
129	Chin	Philip	$38,500.00

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 in the General category in the Properties window after you finish using the TreeView wizard.

❖ **To use an expression for a TreeView column name:**

1 Open the Properties window and click the TreeView level band in the Design view.

2 Click the ellipsis button next to the GroupDefinition property 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 5, "Controlling Updates in DataWindow objects."

Adding and deleting TreeView levels

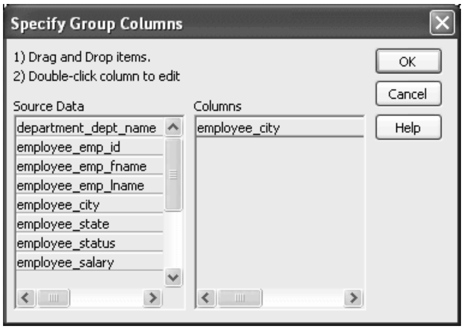
You add and delete TreeView levels using the Rows menu in the DataWindow painter.

❖ **To create an additional level in a TreeView DataWindow:**

1. Open the TreeView DataWindow if it is not already open.
2. Select Rows>Create Group 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 316, the second level has a single column, the employee_city column.



Source Data: department_dept_name, employee_emp_id, employee_emp_fname, employee_emp_lname, employee_city, employee_state, employee_status, employee_salary

Columns: employee_city

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 326.

❖ **To delete a level in a TreeView DataWindow:**

1 Select Rows>Delete Group 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 in the following ways:

- Use the SelectTreeNode method to select a tree node.
- Set the TreeSelectNodeByMouse property to True and then click 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

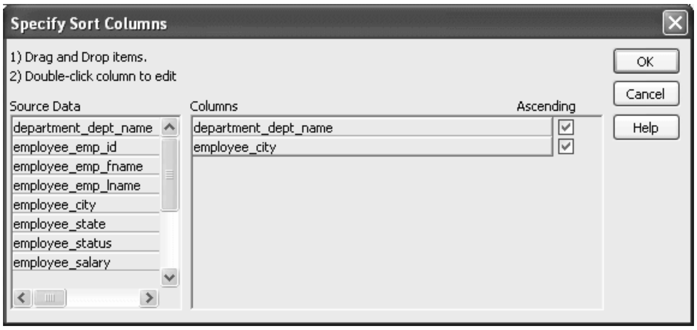
❖ **To sort the rows within levels in a TreeView DataWindow:**

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 316 has department name as the first level and the employee's city of residence as the second level.



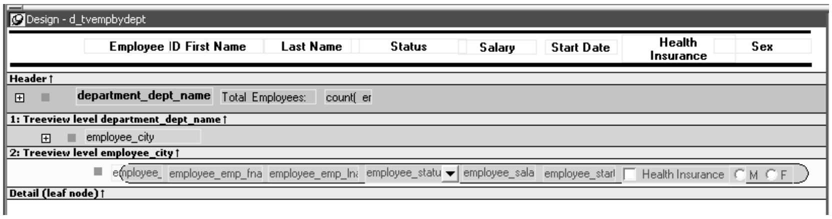
Source Data	Columns	Ascending
department_dept_name	department_dept_name	✓
employee_emp_id	employee_city	✓
employee_emp_fname		
employee_emp_lname		
employee_city		
employee_state		
employee_status		
employee_salary		

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.

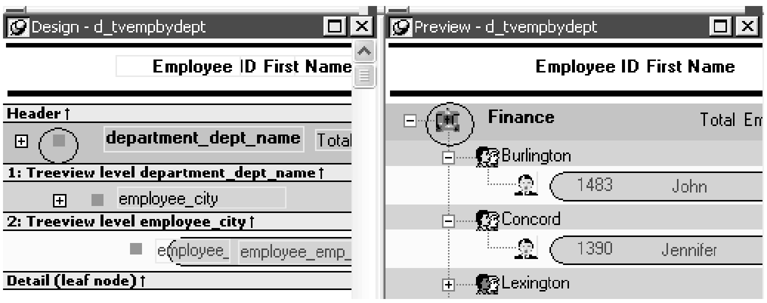
By default, the controls in the header band are the heading text of the detail band columns, and the controls in the detail (leaf node) band are all the column controls except for the first-level columns (in the 1:Treeview level band) that you selected when you used the TreeView wizard. Columns that 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.

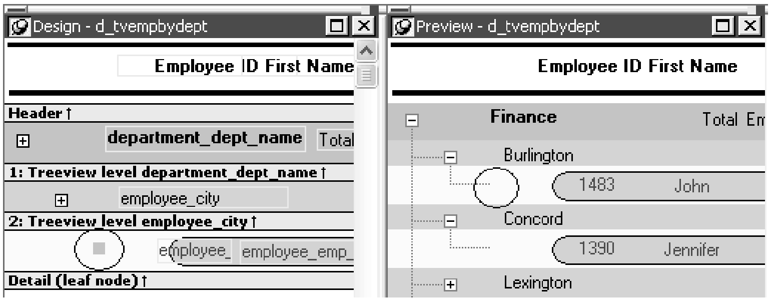
Icons in the Design view

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 316, 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 in the General category in the Properties window for the DataWindow object.

The grid-related properties display only if you select the Grid Style check box when you define the TreeView DataWindow:

Property	Description
GridLines	*On* – Grid lines always display. *Off* – Grid lines never display (columns cannot be resized 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.
GridColumnMove	Columns can be moved at runtime.
RowResize	Rows can be resized at runtime.
SelectedMouse	Data can be selected at runtime and, for example, copied to the clipboard.
TreeDefaultExpandToLevel	Expand to TreeView level 1, 2, or 3.
TreeIndent	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.
TreeSelectNodeByMouse	Whether a Tree node is selected by clicking the Tree node with the mouse.
TreeShowConnectLines	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.
TreeShowLeafNodeConnecLines	Whether lines display that connect the leaf nodes in the detail band rows.
TreeShowTreeNodeIcon	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.
TreeStateIconAlignMode	Align the state icon in the middle (0), at the top (1), or at the bottom (2).

Setting TreeView level properties

In the Properties window for a band, you can specify expanded and collapsed icons for each TreeView level. You access the Properties window by clicking the bar identifying the band for that level in the Design view in the DataWindow painter. You can also access the Properties window 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 324.)

❖ **To modify properties for a level in a TreeView DataWindow:**

- Select Rows>Edit Group 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.

The properties that are specific to a TreeView level band are at the bottom of the Properties window:

Property	Description
TreeLevelCollapsedTreeNodeIconName	The file name of the tree node icon in a TreeView level band when it is in the collapsed state.
TreeLevelExpandedTreeNodeIconName	The file name of the tree node icon in a TreeView level band when it is in the expanded 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 window.

If you want to hide tree nodes in the detail band, set the Height property to 0. The only property that is specific to the TreeView DataWindow is located at the bottom of the Properties window:

Property	Description
TreeLeafTreeNodeIconName	The file name of the tree node icon in the detail band. You can use a quoted expression.

TreeView DataWindow examples

The examples in this section demonstrate how you might use the TreeView DataWindow.

- 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.

- 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 examples 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.

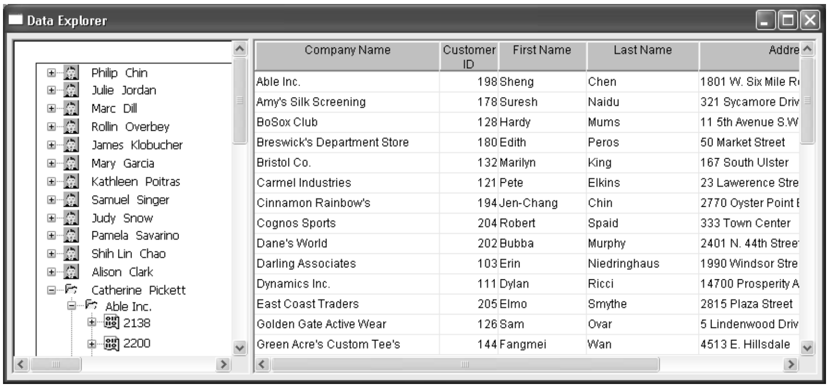
  You create the second level using the Rows>Create Group menu item in the DataWindow painter.

- The third level (level 3) is the sales order ID.

  You also create the third level using the Rows>Create Group menu item in the DataWindow painter.

Data Explorer sample

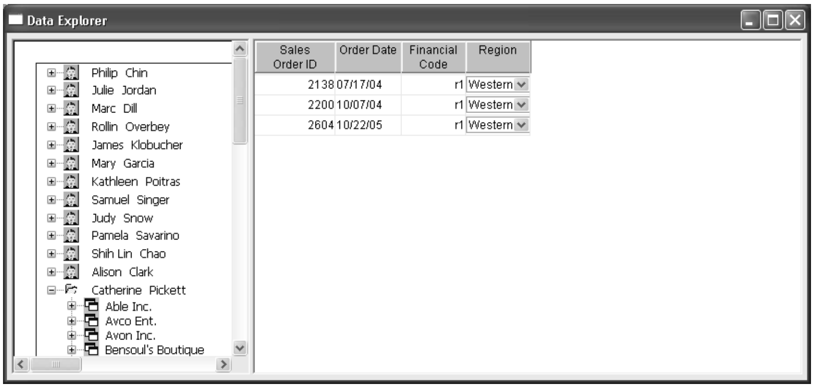
Clicking on each TreeView level displays details in a DataWindow on the right. For example, if you click a name in the TreeView DataWindow on the left, detailed customer data displays in the DataWindow on the right.



- Philip Chin
- Julie Jordan
- Marc Dill
- Rollin Overbey
- James Klobucher
- Mary Garcia
- Kathleen Poitras
- Samuel Singer
- Judy Snow
- Pamela Savarino
- Shih Lin Chao
- Alison Clark
- Catherine Pickett
  - Able Inc.
    - 2138
    - 2200

Company Name	Customer ID	First Name	Last Name	Addre
Able Inc.	198	Sheng	Chen	1801 W. Six Mile R
Amy's Silk Screening	178	Suresh	Naidu	321 Sycamore Driv
BoSox Club	128	Hardy	Mums	11 5th Avenue S.W
Breswick's Department Store	180	Edith	Peros	50 Market Street
Bristol Co.	132	Marilyn	King	167 South Ulster
Carmel Industries	121	Pete	Elkins	23 Lawerence Stre
Cinnamon Rainbow's	194	Jen-Chang	Chin	2770 Oyster Point
Cognos Sports	204	Robert	Spaid	333 Town Center
Dane's World	202	Bubba	Murphy	2401 N. 44th Stree
Darling Associates	103	Erin	Niedringhaus	1990 Windsor Stre
Dynamics Inc.	111	Dylan	Ricci	14700 Prosperity A
East Coast Traders	205	Elmo	Smythe	2815 Plaza Street
Golden Gate Active Wear	126	Sam	Ovar	5 Lindenwood Driv
Green Acre's Custom Tee's	144	Fangmei	Wan	4513 E. Hillsdale

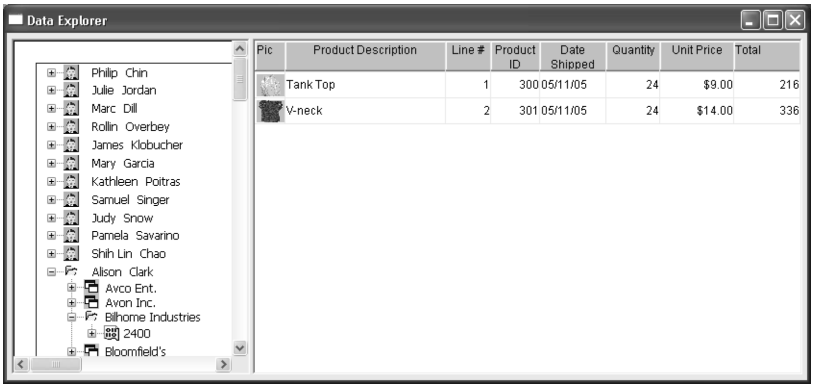
You can click on any TreeView level in the Data Explorer. If you click a company name in the TreeView DataWindow on the left (for example, Able Inc., under Catherine Pickett), order information displays on the right.



Philip Chin
Julie Jordan
Marc Dill
Rollin Overbey
James Klobucher
Mary Garcia
Kathleen Poitras
Samuel Singer
Judy Snow
Pamela Savarino
Shih Lin Chao
Alison Clark
Catherine Pickett
- Able Inc.
- Avco Ent.
- Avon Inc.
- Bensoul's Boutique

Sales Order ID	Order Date	Financial Code	Region
2138	07/17/04	r1	Western
2200	10/07/04	r1	Western
2604	10/22/05	r1	Western

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.

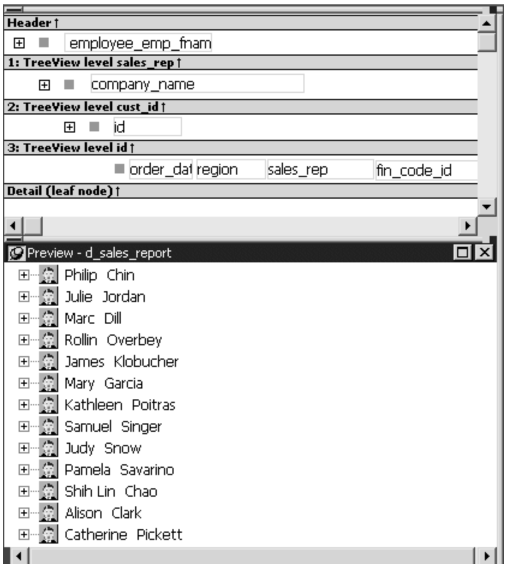


Philip Chin
Julie Jordan
Marc Dill
Rollin Overbey
James Klobucher
Mary Garcia
Kathleen Poitras
Samuel Singer
Judy Snow
Pamela Savarino
Shih Lin Chao
Alison Clark
- Avco Ent.
- Avon Inc.
- Bilhome Industries
  - 2400
- Bloomfield's

Pic	Product Description	Line #	Product ID	Date Shipped	Quantity	Unit Price	Total
	Tank Top	1	300	05/11/05	24	$9.00	216
	V-neck	2	301	05/11/05	24	$14.00	336

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

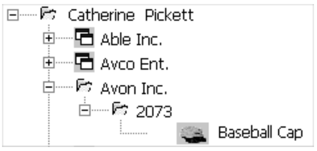
The code in the Clicked event uses the ObjectAtPointer.Band property to determine which DataWindow to display. Clicking on some editable items in the detail DataWindow opens a window in which you can manipulate the data.

The PopMenu 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

When you first run the Data Linker, no data displays on the right side of the window.

To use the Data Linker, you first expand an employee name and a company's data in the TreeView DataWindow.

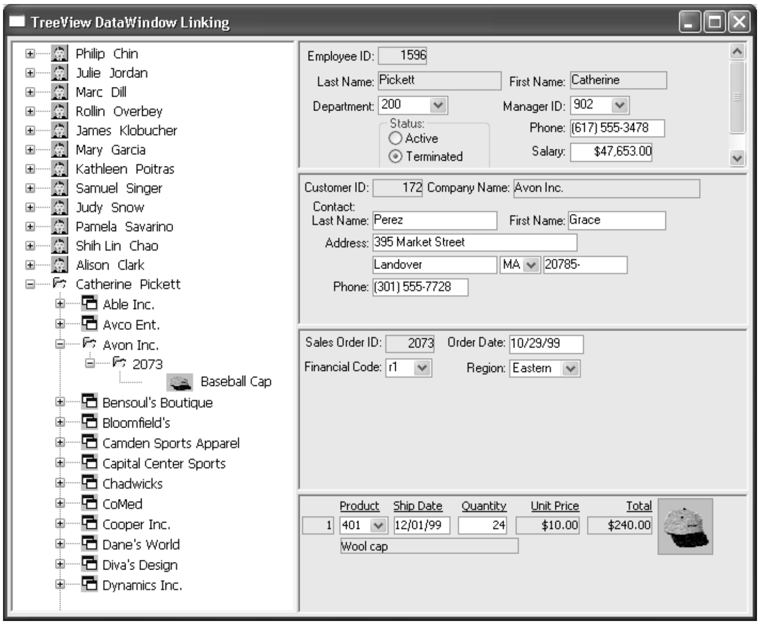


Catherine Pickett
- Able Inc.
- Avco Ent.
- Avon Inc.
  - 2073
    - Baseball Cap

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.



Philip Chin
Julie Jordan
Marc Dill
Rollin Overbey
James Klobucher
Mary Garcia
Kathleen Poitras
Samuel Singer
Judy Snow
Pamela Savarino
Shih Lin Chao
Alison Clark
Catherine Pickett
- Able Inc.
- Avco Ent.
- Avon Inc.
  - 2073
    - Baseball Cap
- Bensoul's Boutique
- Bloomfield's
- Camden Sports Apparel
- Capital Center Sports
- Chadwicks
- CoMed
- Cooper Inc.
- Dane's World
- Diva's Design
- Dynamics Inc.

Employee ID: 1596
Last Name: Pickett First Name: Catherine
Department: 200 Manager ID: 902
Status: Active [✓ Terminated] Phone: (617) 555-3478
Salary: $47,653.00

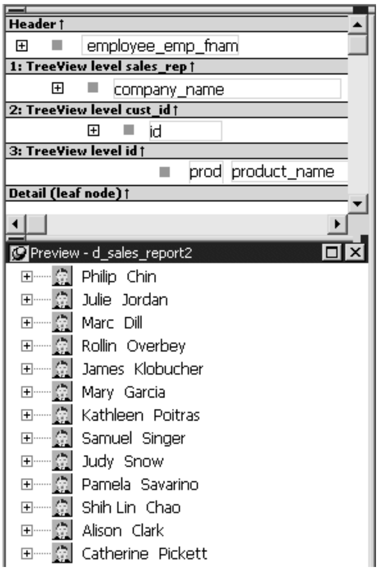
Customer ID: 172 Company Name: Avon Inc.
Contact: Last Name: Perez First Name: Grace
Address: 395 Market Street
Landover MA 20785-
Phone: (301) 555-7728

Sales Order ID: 2073 Order Date: 10/29/99
Financial Code: r1 Region: Eastern

	Product	Ship Date	Quantity	Unit Price	Total
1	401	12/01/99	24	$10.00	$240.00
	Wool cap				

Data Linker TreeView DataWindow

Here is the TreeView DataWindow used in the Data Linker sample.



Philip Chin
Julie Jordan
Marc Dill
Rollin Overbey
James Klobucher
Mary Garcia
Kathleen Poitras
Samuel Singer
Judy Snow
Pamela Savarino
Shih Lin Chao
Alison Clark
Catherine Pickett

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 the ObjectAtPointer.Band property to determine which DataWindow to display.

CHAPTER 13

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

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 &amp;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 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 341.

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.

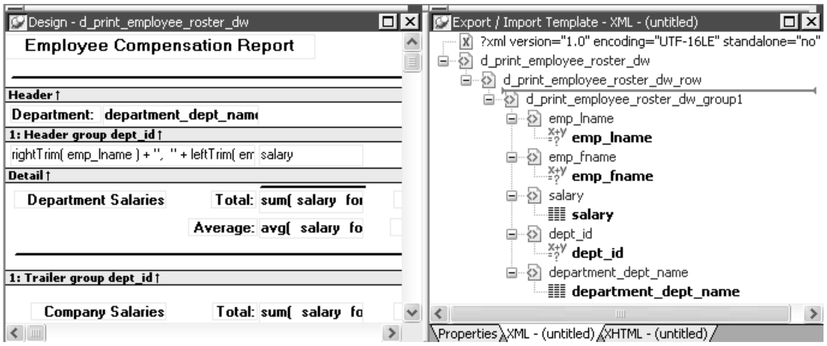
Only the mapping of column names to element and attribute names is used for import. All other information in the template is ignored.

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. The view does not display by default. To display it, select View>DataWindow Painter Layout>Export/Import Template>XML.

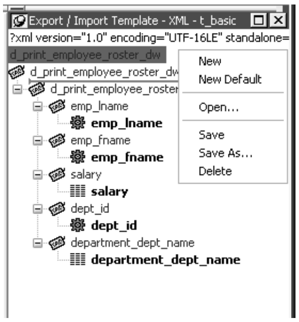
When you create a new DataWindow object, DataWindow Designer displays a default template in the Export/Import Template view. You can edit only one template at a time in the view, but you can create multiple templates and save them with the DataWindow object. Each template is uniquely associated with the DataWindow object open in the painter.

The default template has one element for each column in the DataWindow object.


Design - d_print_employee_roster_dw
Employee Compensation Report
Header
Department: department_dept_name
1: Header group dept_id
rightTrim(emp_lname) + ", " + leftTrim(em salary
Detail
Department Salaries Total: sum(salary for
Average: avg(salary fo
1: Trailer group dept_id
Company Salaries Total: sum(salary fo
Export / Import Template - XML - (untitled)
?xml version="1.0" encoding="UTF-16LE" standalone="no"
d_print_employee_roster_dw
d_print_employee_roster_dw_row
d_print_employee_roster_dw_group1
emp_lname
emp_lname
emp_fname
emp_fname
salary
salary
dept_id
dept_id
department_dept_name
department_dept_name
Properties
XML - (untitled)
XHTML - (untitled)

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. The end tags of elements and the markup delimiters used in an XML document do not display.

Table 13-1 shows the icons used in the Export/Import Template view.

Table 13-1: 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

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.

Creating new base templates

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.

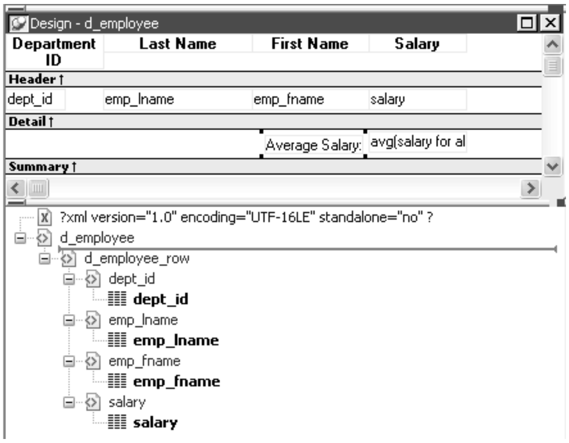
For example, if the DataWindow object is named d_name, the default template has this structure:

```
<?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:



```
?xml version="1.0" encoding="UTF-16LE" standalone="no" ?
d_employee
    d_employee_row
        dept_id
            dept_id
        emp_lname
            emp_lname
        emp_fname
            emp_fname
        salary
            salary
```

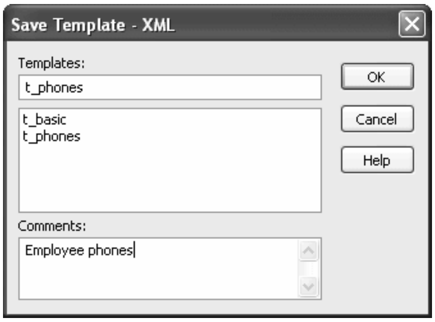
The SQL syntax is:

```
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.



t_phones

t_basic
t_phones

Employee phones

The template is stored inside the DataWindow object in the PBL.

After saving a template with a DataWindow object, you can export the definition to a text file using the Export item on the DataWindow object's pop-up menu in the Solution Explorer. For example, this is part of the source for a DataWindow that has two templates. The templates have required elements only:

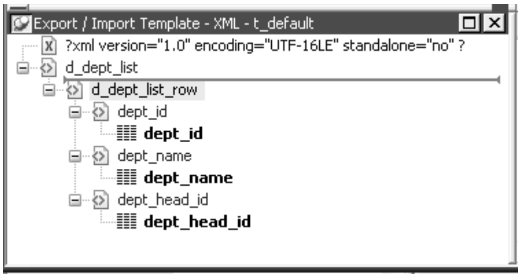
```
export.xml(usetemplate="t_address"
   template=(comment="Employee Phone Book"
      name="t_phone" xml="<d_emplist><d_emplist_row
      __pbband=~"detail~"/></d_emplist>")
   template=(comment="Employee Address Book"
      name="t_address" xml="<d_emplist><d_emplist_row
      __pbband=~"detail~"/></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 354.

The Detail section contains the row data, and is generated iteratively for each row in the DataWindow object.


Export / Import Template - XML - t_default
?xml version="1.0" encoding="UTF-16LE" standalone="no" ?
d_dept_list
d_dept_list_row
dept_id
dept_id
dept_name
dept_name
dept_head_id
dept_head_id

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 13-2. Only the root element is required:

Table 13-2: 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 345.
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 346.
Comments	See “Comments” on page 351.
Processing instructions	See “Processing instructions” on page 352.
Root element (start tag)	See “Root element” on page 347.
Group header elements	See “Generating group headers” on page 354.
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 13-3.

Table 13-3: Items permitted in the Detail section of an XML document

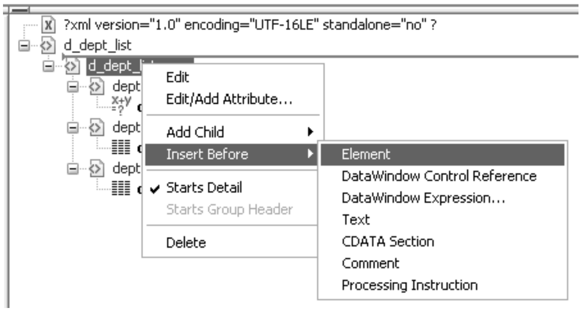
Item	Details
Detail Start element	See “The Detail Start element” on page 342.
Child or sibling elements to the Detail Start element	To add a sibling to the Detail Start element, add a child to its parent (the root element by default).
Control references	These references are in text format and can include references to column, text, computed field, and report controls. See “Controls” on page 348. Nested report controls can only be referenced as child elements. See “Composite and nested reports” on page 349.

Item	Details
DataWindow expressions	See “DataWindow expressions” on page 348.
Literal text	Literal text does not correspond to a control in the DataWindow object.
Comments	See “Comments” on page 351.
Processing instructions	See “Processing instructions” on page 352.
CDATA sections	See “CDATA sections” on page 351.
Attributes	You can assign attributes to all element types. See “Attributes” on page 349.

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 357.

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. 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`.

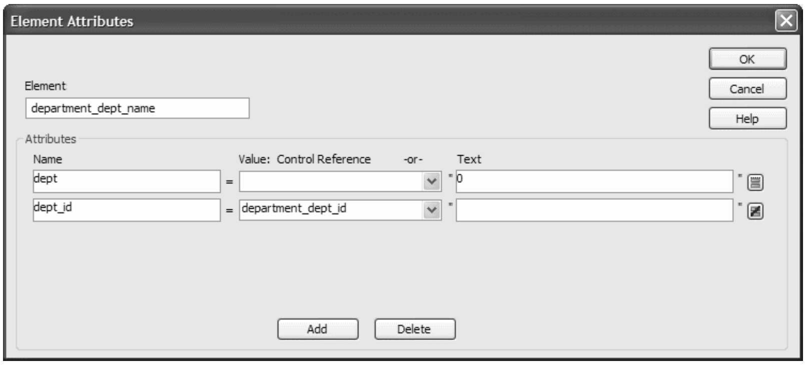
However, if the export template uses an expression that combines a column with a Date or DateTime datatype with a string, the entire expression is exported as a string and the regional settings in the Windows registry are used to format the date and time.

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.



Element: department_dept_name

Attributes

Name		Value: Control Reference	-or- Text
dept	=		"0"
dept_id	=	department_dept_id	""

The expression button shows an equals sign if an expression has been entered, and a 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:

  ```
  ds1.SaveAs("C:\TEMP\Temp.xml", Xml)
  ```

- Using the Describe method:

  ```
  ls_xmlstring = dw1.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 339.

Setting data export properties

The Data Export category in the Properties window lets you set properties for exporting data to XML.

In addition to the properties that you can set n this category, 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 361.

The Use Template property

The names of all templates that you create and save for the current DataWindow object display in the ExportXMLUseTemplate 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 code. For example, an XML publishing engine would change templates dynamically to create different presentations of the same data.

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>DataWindow Painter Layout>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.

Template used when saving

When the DataWindow is saved as XML, DataWindow Designer uses the template specified in the UseTemplate 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 UseTemplate property might not be the template currently displayed in the Export/Import Template view, so you should check the value of the UseTemplate 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.

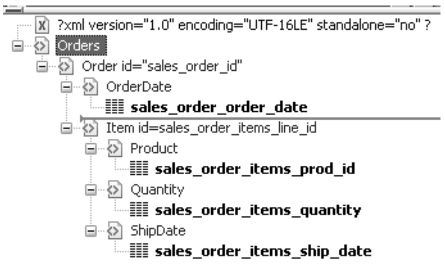
2 In the Data Export category in the properties window, select the current template from the ExportXMLUseTemplate drop-down list.

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, 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:



```
?xml version="1.0" encoding="UTF-16LE" standalone="no" ?
Orders
    Order id="sales_order_id"
        OrderDate
            sales_order_order_date
        Item id=sales_order_items_line_id
            Product
                sales_order_items_prod_id
            Quantity
                sales_order_items_quantity
            ShipDate
                sales_order_items_ship_date
```

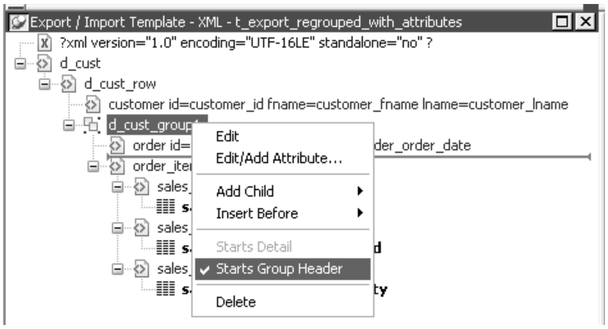
The root element in the Header section of the template, Orders, has a child element, Order. Order has an id attribute whose value is a control reference to the column sales_order_id. Order also has a child element, OrderDate, that contains a column reference to the sales_order_order_date column. These 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 Export.XML.HeadGroups 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>
         ...
```

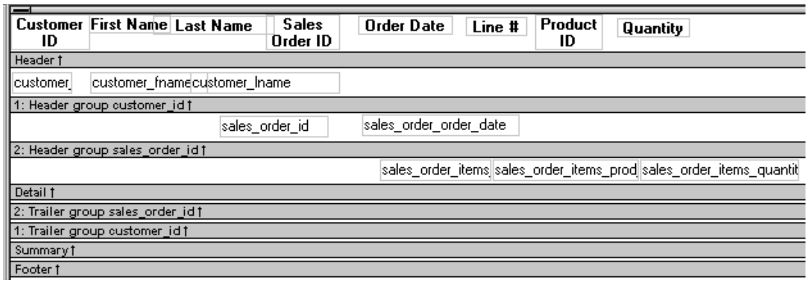
For DataWindow objects with more than one group, when you generate a new default template, each group after the first is identified with a special icon and a check on the pop-up menu next to the Starts Group Header item.


Export / Import Template - XML - t_export_regrouped_with_attributes

```
?xml version="1.0" encoding="UTF-16LE" standalone="no" ?
d_cust
  d_cust_row
    customer id=customer_id fname=customer_fname lname=customer_lname
    d_cust_group
      order id=                    der_order_date
      order_ite
        sales_
          s
        sales_
          s                        d
        sales_
          s                        ty
```

When the Export.XML.HeadGroups property is set to True, 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:



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>
```

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, set the Export.XML.IncludeWhitespace property. 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 360.

To specify how metadata should be saved, select a value from the ExportXMLMetaDataType 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 ExportXMLSaveMetaData 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 [<!ELEMENT dtdinternal
(dtdinternal_row*)>
<!ELEMENT dtdinternal_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)>
]>
<dtdinternal>
   <dtdinternal_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>
   </dtdinternal_row>
```

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 342.

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 UseTemplate drop-down list in the Data Import category in the Properties window.

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 category also contains a property that enables you to create a trace log of the import. See "Tracing import" on page 369.

Example

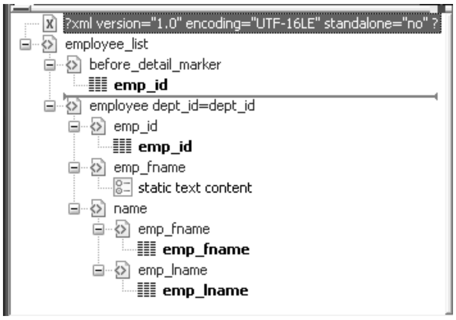
This example uses a DataWindow object that includes the columns emp_id, emp_fname, emp_lname, and dept_id. The template used in this example includes only these columns. Any other columns in the DataWindow remain empty when you import using this template.

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 in the Data Import category in the Properties window 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 UseTemplate 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.

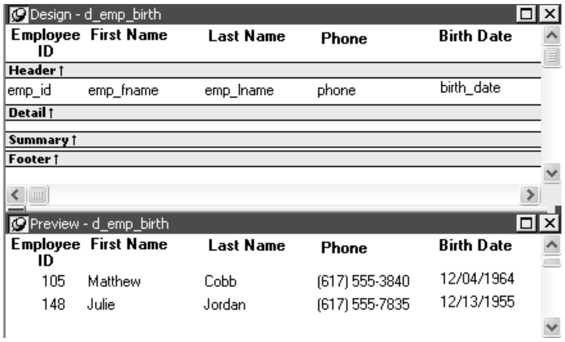
Elements with non-text content

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 of 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.



Employee ID	First Name	Last Name	Phone	Birth Date
105	Matthew	Cobb	(617) 555-3840	12/04/1964
148	Julie	Jordan	(617) 555-7835	12/13/1955

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, set the ImportXMLTrace property in the Data Import category in the Properties window and specify the name and location of the log file in the ImportXMLTraceFile property. 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 an import method to import the data, you must specify Xml as the *importtype* argument. For example:

```
ImportString(xmlString, FileSaveAsType.Xml);
```

If you omit the *importtype* argument, the trace file is not created.

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 363. 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
```

CHAPTER 14 Working with Graphs

About this chapter

This chapter describes how to build and use graphs in DataWindow Designer.

Contents

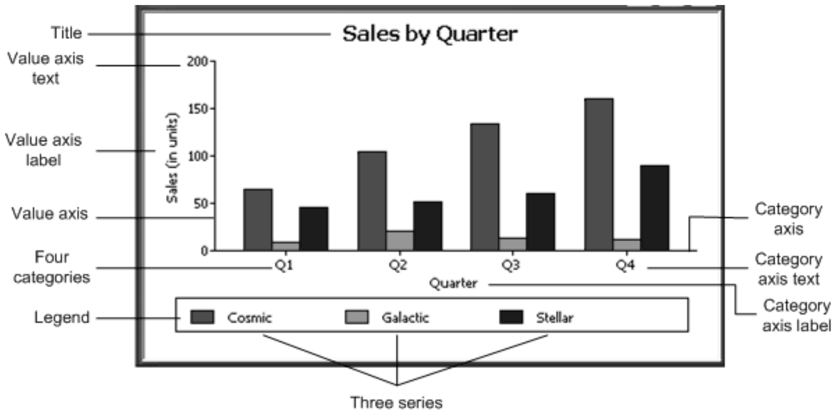
About graphs

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. 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 14-1: 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

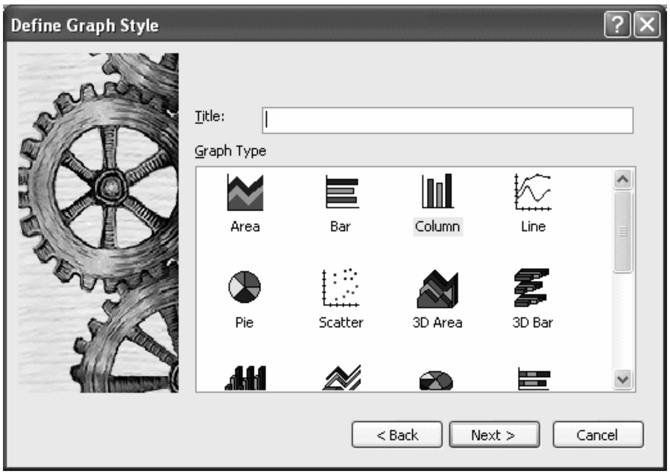
Table 14-2 lists the parts of a typical graph.

Table 14-2: 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 category in the Properties window for the graph.



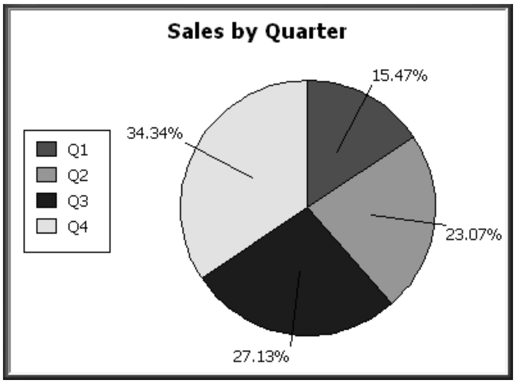
Area, bar, column, and line graphs

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.

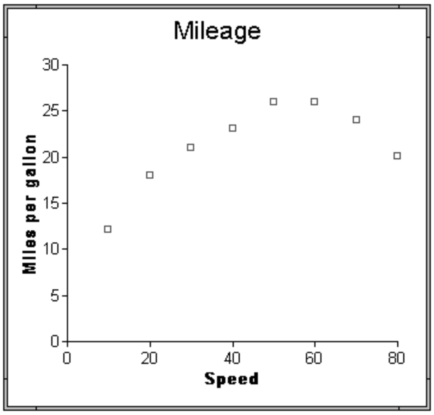
Scatter graphs do not use categories. Instead, numeric values are plotted along both axes—as opposed to other graphs, which have values along one axis and categories along the other axis.

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

Speed	Mileage
60	26
70	24
80	20

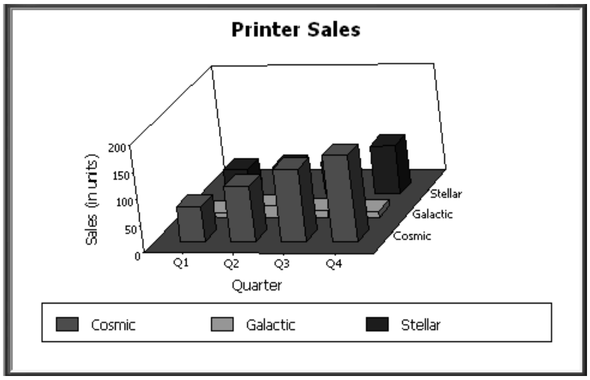
Here is the data in a scatter graph:



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.

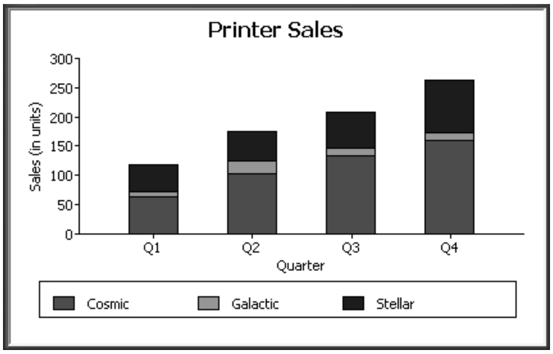
Three-dimensional graphs

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 390.

Placing a graph in a DataWindow object

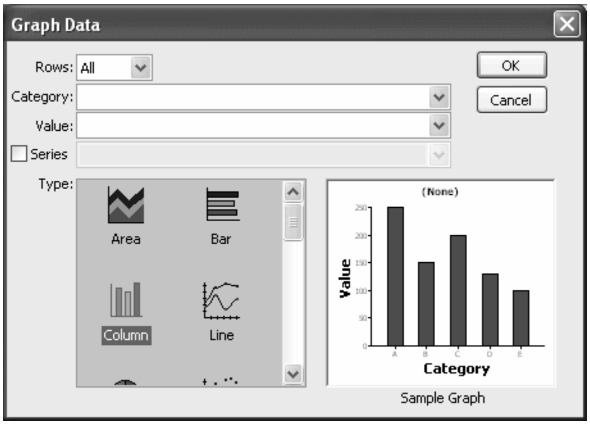
❖ **To place a graph in a DataWindow object:**

1 Open or create the DataWindow object that will contain the graph.

2 Select Graph from the Visual Studio Toolbox.

3 Click where you want the graph.

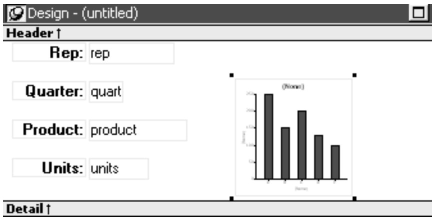
DataWindow Designer displays the Graph Data dialog box:



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 383.

The Design view now contains a representation of the graph:



5 Specify the graph's properties in the Properties window.

Using the graph's Properties window

A graph has a Properties window in which you can specify the data as well as the other properties of the graph.

❖ **To display the graph's Properties window:**

- Select Properties from the graph's pop-up menu.

The Properties window for a graph has several categories in which you specify information about the graph. Table 14-3 lists the property categories that contain properties that are specific to graphs and describes what each property category specifies.

Table 14-3: Property categories for graphs

Category	What it specifies
Appearance	Various appearance-related properties, including border, graph colors, whether to size the graph to the full screen display, suppression in newspaper columns, and the pointer to use when the mouse is positioned over the graph.
Axis	Labels, scale, information about major and minor divisions for the category axes.
Data	Where to get the graph's data.
General	Graph type, title, legend location. For 3D graphs, perspective, rotation, and elevation. For bar graphs, overlap, spacing and depth of bars.
Layout	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.
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.

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 Layout category or the Appearance category in the Properties window.

2 Select the settings for the following options in the Position category:

Table 14-4: Settings in the Position category for graphs

Setting	Meaning
Layer	*Background* — The graph displays behind other elements in the DataWindow object. *Band* — 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. *Foreground* — (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.
SlideLeft, SlideUp	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 189.
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 in the General category:

Table 14-5: Size and position settings in the Appearance category

Setting	Meaning
SizeToDisplay	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 category in the Properties window.

2 Fill in the boxes as described in the sections that follow, and click OK.

Specifying which rows to include in a graph

The DataRows drop-down list allows you to specify which rows of data are graphed at any one time:

Table 14-6: 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 in the Data category in the Properties window, you can select a column name from a drop-down list.

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 221. For more about LookupDisplay, see the *DataWindow Object Reference*.

Specifying the values

DataWindow Designer populates the DataValues 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 window). 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, set the DataSeriesOnOff property to true and specify the column that will provide the series values. You can select column names from the DataSeries 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 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.

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 DataValues box as you do in the DataSeries box. The first value in the DataValues box corresponds to the first series identified in the DataSeries box, the second value corresponds to the second series, and so on. The example about graphing actual and projected sales in “Examples” on page 385 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 14-7: The Printer table in the EAS Demo DB

Rep	Quarter	Product	Units
Simpson	Q1	Stellar	12
Jones	Q1	Stellar	18
Perez	Q1	Stellar	15

Rep	Quarter	Product	Units
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

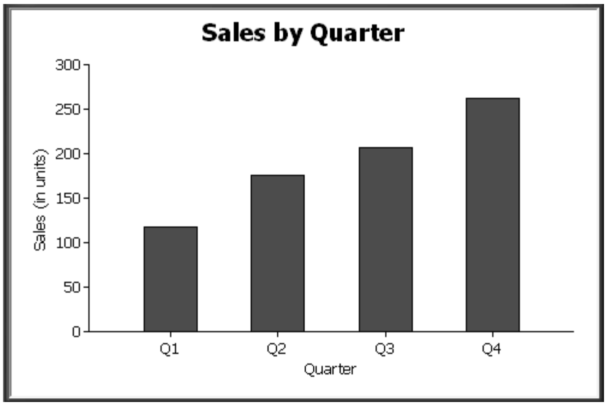
To graph total sales of printers in each quarter, retrieve all the columns into a DataWindow object and create a graph with the following settings in the Data category in the Properties window:

- Set DataRows to `All`
- Set DataCategory to `quarter`
- Set DataValues to `sum(units for graph)`

Set the DataSeriesOnOff property to False and leave the DataSeries property empty.

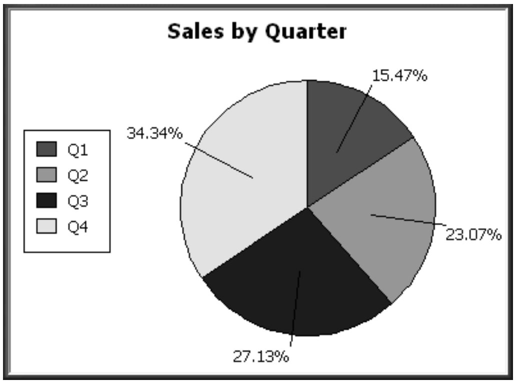
The Quarter column serves as the category. Because the Quarter column has four values (Q1, Q2, Q3, and Q4), there will be four categories along the Category axis. You want only one series (total sales in each quarter), so you can leave the DataSeries box empty, or type a string literal to identify the series in a legend. Setting DataValues 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

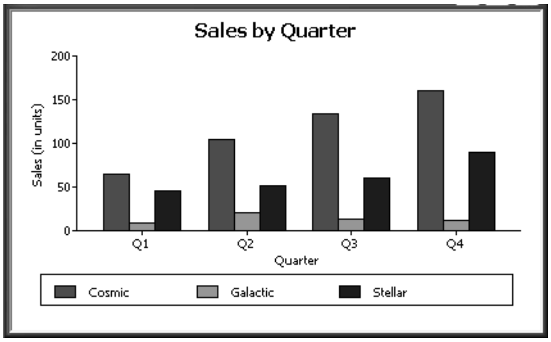
To graph total quarterly sales of each printer, retrieve all the columns into a DataWindow object and create a graph with the following settings in the Data category in the Properties window:

- Set DataRows to `All`
- Set DataCategory to `quarter`
- Set DataValues to `sum(units for graph)`

- Set DataSeriesOnOff to true
- Set DataSeries 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 DataValues 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:

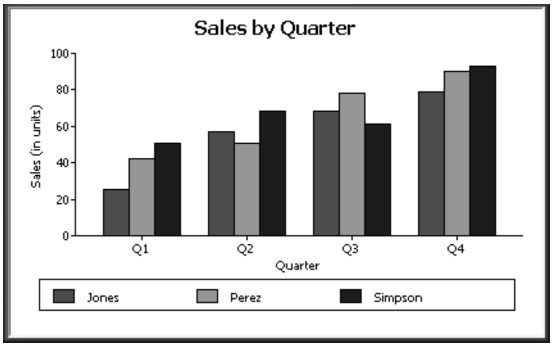


Graphing unit sales by representative

To graph quarterly sales made by each representative, create a graph with the following settings in the Data category in the Properties window:

- Set DataRows to `All`
- Set DataCategory to `quarter`
- Set DataValues to `sum(units for graph)`
- Set DataSeriesOnOff to true
- Set DataSeries to `rep`

Here is the resulting graph:



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:

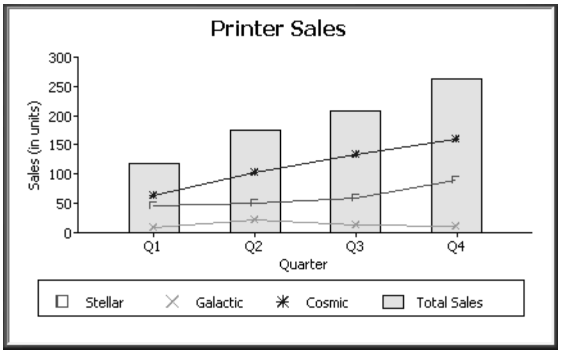
  "@overlay~t*SeriesLabel* "

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 387, but use the following expression in the Series box:

```
"Total Sales", "@overlay~t" + product
```

Here is the resulting graph:



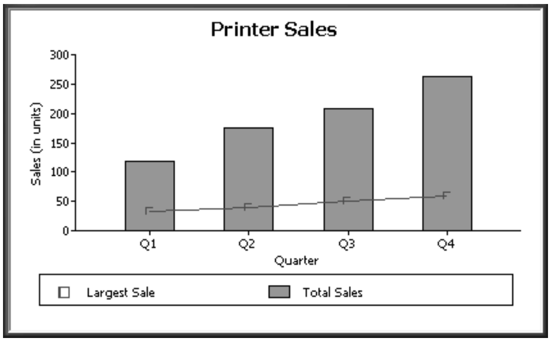
To graph unit sales of printers by quarter and overlay the largest sale made in each quarter, change the Value expression to this:

```
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 In the Solution Explorer, right-click the library where you want to save the DataWindow object and select Add New Entry.

2 In the Add New Entry dialog box, select DataWindow Object from the categories list and select the Graph DataWindow style, provide a name for the DataWindow object, and click Add.

3 On the Choose Data Source for Graph DataWindow page, specify the data you want retrieved into the DataWindow object.

   For more information, see Chapter 3, “Defining DataWindow Objects.”

4 On the Define Graph Data page, enter the definitions for the series, categories, and values, as described in “Associating data with a graph” on page 383, and click Next.

   Note that when using the Graph presentation style, the graph always graphs all rows; you cannot specify page or group.

5 On the Define Graph Style page, enter a title for the graph, select a graph type, and click Next.

6 On the Ready to Create Graph DataWindow page, 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.

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 window.

Using the General category in the graph's Properties window

You name a graph and define its basic properties in the General category in the graph's Properties window.

❖ **To specify the basic properties of a graph:**

- Select Properties from the graph's pop-up menu and then select the General category in the Properties window.

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.

Defining a graph's title

The title displays at the top of the graph.

❖ **To specify a graph's title:**

- In the General properties category for the graph, enter a title in the Title property.

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 393.

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:**

- In the General properties category for the graph, select a graph type from the GraphType 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:**

- In the General properties category 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 393.

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 In the General properties category for the graph, adjust the point of view along the three dimensions of the graph:

   - To change the perspective, specify a number between 1 and 100. The larger the number, the smaller the graph appears. The default is 2.
   - To rotate the graph, specify a number between -90 and 90. The default is -20.
   - To change the elevation, specify a number between 1 and 100. The default is 20.

2 Define the depth of the graph (the percent the depth is of the width of the graph) by .specifying a number between 1 and 100. The default is 100.

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:**

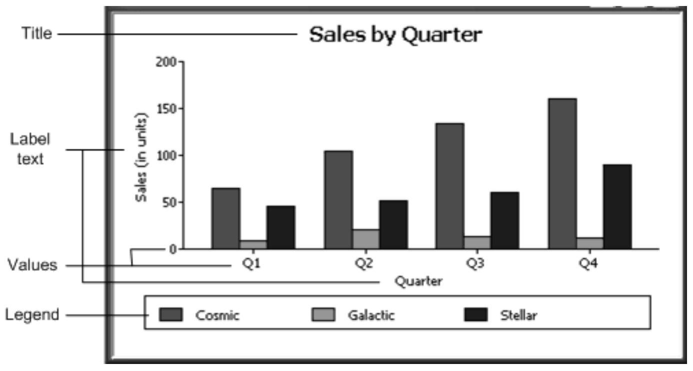
1 In the Axis category, select the axis for which you want to specify sorting.

2 Scroll to Sort and select Ascending, Descending, 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 In the Text category, select a text element from the list in the TextObject drop-down list.

2 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 In the Text category, select a text element from the list in the TextObject drop-down list.

2 Set the DispattrAutoSize property to true (this is the default).

❖ **To specify a font size for a text element in a graph:**

1 In the Text category, select a text element from the list in the TextObject drop-down list.

2 Set the DispattrAutoSize property to False.

3 Select the Font size in the DispattrFontHeight property.

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. In the Text category, select a text element from the list in the TextObject drop-down list.
2. Specify the rotation you want in the DispattrFontEscapement property 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. In the Text category, select a text element from the list in the TextObject drop-down list.
2. Type a display format in the DispattrFormat property.

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. In the Text category, select a text element from the list in the TextObject drop-down list.
2. Click the ellipsis button next to the DispattrDisplayExpression property.

   The Modify Expression dialog box displays.
3. Specify the expression.

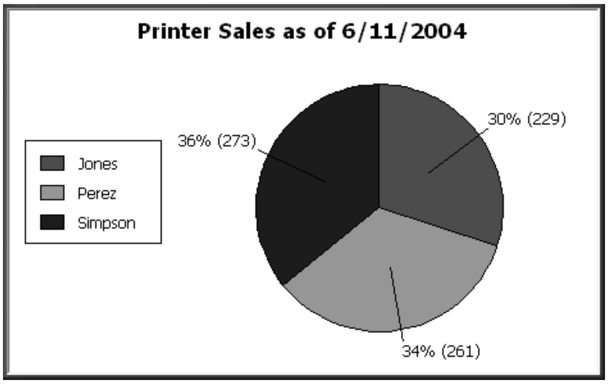
   You can paste functions, column names, and operators. Included with column names in the Columns box are statistics about the columns, such as counts and sums.
4. Click OK to return to the graph's Properties window.

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 14-8 in the General category.

Table 14-8: Overlap and spacing properties for bar and column charts

Property	Meaning
OverlapPercent	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.

Specifying axis properties

Graphs have two or three axes. You specify the axes' properties in the Axis category in the graph's Properties window.

❖ **To specify properties for an axis of a graph:**

1. Select Properties from the graph's pop-up menu and then select the Axis category in the Properties window.
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 14-9 shows the two kinds of text associated with an axis.

Table 14-9: 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 393.

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 14-10.

Table 14-10: 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 14-11 to define the scaling used along numeric axes.

Table 14-11: 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.
RoundTo, RoundToUnit	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.
MinimumValue, MaximumValue	The smallest and largest numbers to appear on the axis (disabled if you have selected Autoscale).
ScaleType	Specifies linear or logarithmic scaling (common or natural).
ScaleValue	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 385, 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:**

1 To divide an axis into a specific number of major divisions, type the number of divisions you want in the MajorDivisions property.

   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 property. For example, if you enter 2, DataWindow Designer labels every second tick mark for the major divisions.

2 To use minor divisions, which are divisions within each major division, type the appropriate number in the MinorDivisions property. To use no minor divisions, leave the number 0.

When using logarithmic axes
If you want minor divisions, specify 1; otherwise, specify 0.

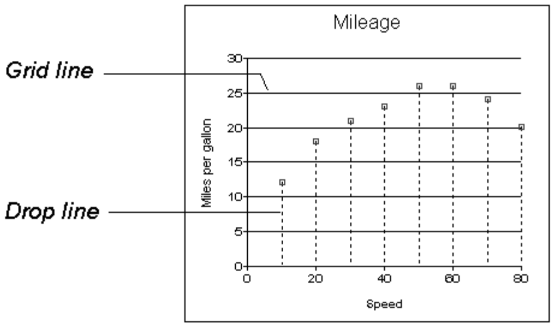
Representing divisions with grid and drop lines

You can specify lines to represent the divisions as described in Table 14-12 and illustrated in Figure 14-1.

Table 14-12: 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 14-1: Grid and drop lines in a graph



Using line styles

You can define line styles for the components of a graph listed in Table 14-13.

Table 14-13: 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 category in the Properties window.

2 Select a stock pointer from the list.

PART 3

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.

APPENDIX A Identifiers

About this chapter

You use identifiers to name objects. This chapter describes valid identifiers.

Contents

Rules

Identifiers:

- Must start with a letter
- Can have up to 255 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 14-14: Reserved words

alias	else	insert	procedure	this
and	elseif	into	protected	throw
autoinstantiate	end	intrinsic	protectedread	throws
call	enumerated	is	protectedwrite	to
case	event	last	prototypes	trigger
catch	execute	library	public	true
choose	exit	loop	readonly	try
close	external	namespace	ref	type
commit	false	native	return	until
connect	fetch	next	rollback	update
constant	finally	not	rpcfunc	updateblob
continue	first	of	select	using
create	for	on	selectblob	variables
cursor	forward	open	shared	while
declare	from	or	static	with
delete	function	parent	step	within
describe	global	post	subroutine	xor
descriptor	goto	prepare	super	_debug
destroy	halt	prior	system	
disconnect	if	private	systemread	
do	immediate	privateread	systemwrite	
dynamic	indirect	privatewrite	then	

APPENDIX B

The Extended Attribute System Tables

About this appendix

This appendix describes each column in the extended attribute system tables.

Contents

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, normalized 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

Column	Column name	Description
11	pbh_fhgt	Headings font height, normalized 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)
17	pbh_ffce	Headings font typeface
18	pbl_fhgt	Labels font height, normalized 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

Column	Column name	Description
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, normalized units
14	pbc_wdth	Column width, normalized units
15	pbc_ptrn	Validation rule name
16	pbc_bmap	Bitmap/picture (Y=Yes, N=No)
17	pbc_init	Initial value
18	pbc_cmnt	Column comments
19	pbc_edit	Edit style name
20	pbc_tag	(Reserved)

Table B-4: The PBCatFmt table

Column	Column name	Description
1	pbf_name	Display format name
2	pbf_frmt	Display format
3	pbf_type	Datatype 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	Datatype 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 “Edit style types for the PBCatEdt table” next)
3	pbe_type	Edit style type (see Table B-7)
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

Column	Column name	Description
6	pbe_flag	Edit style flag (edit style type dependent)
7	pbe_work	Extra field (edit style type dependent)

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	*Text*	85	1	1	*Flag*	
MyEdit	*OnValue*	85	1	2	0	
MyEdit	*OffValue*	85	1	3	0	
MyEdit	*ThirdValue*	85	1	4	0	

Table B-9: Values used in CheckBox edit style sample

Value	Meaning
Text	CheckBox text
OnValue	Data value for On state
OffValue	Data value for Off state
ThirdValue	Data value for Third state (this row exists only if 3 State is checked for the edit style—bit 30 of *Flag* is 1)

Value	Meaning
Flag	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. 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	*Columns*	86	1	1	*Flag*	
MyEdit	*Display1*	86	1	2	0	
MyEdit	*Data1*	86	1	3	0	
MyEdit	*Display2*	86	1	4	0	
MyEdit	*Data2*	86	1	5	0	

Table B-11: Values used in RadioButton edit style sample

Value	Meaning
Columns	Character representation (in decimal) of number of columns (buttons) across.
Display1	Display value for first button.
Data1	Data value for first button.
Display2	Display value for second button.
Data2	Data value for second button. Display and data values are repeated in pairs for each radio button defined in the edit style.

Value	Meaning
Flag	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. 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	*Limit*	87	1	1	*Flag*	*Key*
MyEdit	*Display1*	87	1	2	0	
MyEdit	*Data1*	87	1	3	0	
MyEdit	*Display2*	87	1	4	0	
MyEdit	*Data2*	87	1	5	0	

Table B-13: Values used in DropDownListBox edit style sample

Value	Meaning
Limit	Character representation (in decimal) of the *Limit* value.
Key	One-character accelerator key.
Display1	Display value for first entry in code table.
Data1	Data value for first entry in code table.
Display2	Display value for second entry in code table.
Data2	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
Flag	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. 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	*DataWin*	88	1	1	*Flag*	*Limit*
MyEdit	*DataCol*	88	1	2	0	*Key*
MyEdit	*DisplayCol*	88	1	3	0	*Width%*

Table B-15: Values used in DropDownDataWindow edit style sample

Value	Meaning
DataWin	Name of DataWindow object to use.
DataCol	Data column from DataWindow object.
DisplayCol	Display column from DataWindow object.

Value	Meaning
Limit	Character representation (in decimal) of *Limit* value.
Key	One-character accelerator key.
Width%	Width of the dropdown part of the DropDownDataWindow in %.
Flag	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. 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 7, “Displaying and Validating Data.”

Table B-16: Sample row in PBCatEdt for an Edit edit style

Name	Edit	Type	Cntr	Seqn	Flag	Work
MyEdit	*Limit*	89	1	1	*Flag*	*Key*
MyEdit	*Format*	89	1	2	0	*Focus*
MyEdit	*Display1*	89	1	3	0	
MyEdit	*Data1*	89	1	4	0	
MyEdit	*Display2*	89	1	5	0	
MyEdit	*Data2*	89	1	6	0	

Table B-17: Values used in Edit edit style sample

Value	Meaning
Limit	Character representation (in decimal) of *Limit* value.
Key	One-character accelerator key.
Format	Display format mask.
Focus	Character "1" if Show Focus Rectangle is checked. NULL otherwise.
Flag	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. 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 7, “Displaying and Validating Data.”

Table B-18: Sample row in PBCatEdt for an EditMask edit style

Name	Edit	Type	Cntr	Seqn	Flag	Work
MyEdit	*Format*	90	1	1	*Flag*	*DtFcKy*
MyEdit	*Range*	90	1	2	0	*SpinInc*
MyEdit	*Display1*	90	1	3	0	
MyEdit	*Data1*	90	1	4	0	
MyEdit	*Display2*	90	1	5	0	
MyEdit	*Data2*	90	1	6	0	

Table B-19: Values used in EditMask edit style sample

Value	Meaning
Format	Display format mask.
DtFcKy	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 DataTime= "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
Range	Character representation (in decimal) of spin control range. The min value and max value are tab-delimited. Example: "1[tab]13" means min = 1, max = 13
SpinInc	Character representation (in decimal) of spin increment.
Display1	Display value for first entry in code table.
Data1	Data value for first entry in code table.
Display2	Display value for second entry in code table.
Data2	Data value for second entry in code table. Display and data values are repeated in pairs for each entry in the code table.
Flag	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. 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

Index