DB2 Query Management Facility

# DB2 QMF Visionary Developer's Guide

*Version 8 Release 1*

SC18-9093-01

DB2 Query Management Facility

# DB2 QMF Visionary Developer's Guide

*Version 8 Release 1*

SC18-9093-01

**Note:**

Before using this information and the product it supports, be sure to read the general information under ″Notices.″

**Second Edition (June 2004)**

This edition applies to IBM DB2 QMF for Windows and IBM DB2 QMF for WebSphere, Version 8, Release 1, a feature of QMF Distributed Edition Version 8.1, 5724-E86, and a feature of the QMF Family with Version 8.1 of DB2 for z/OS, 5625-DB2, and to all subsequent releases and modifications until otherwise indicated in new editions.

# Contents

# Tables

# Figures

# About this book

This book contains information about how to create applications using QMF™ Visionary Studio. QMF Visionary Studio is a suite of design tools for creating business analysis applications. QMF Visionary lets you create navigable and graphical presentations that are tied to current data in your databases.

## Who should read this book

This book is written for application developers who want to develop dynamic, informative visual presentations of business data. The book is also written for database administrators who work with the application developers.

Application developers should be technologically adept and knowledgeable about writing SQL queries.

Database administrators should have experience and expertise with database technology and should be able to help the business domain experts with the SQL programming language, database schema issues, and storage and retrieval strategies.

This book is written with the assumption that the two types of users have the following combined backgrounds:

- Some experience analyzing your company's business data and its data model
- Some experience working with relational databases or exposure to database concepts
- Experience using Microsoft® Excel, Microsoft PowerPoint, or some other data-analysis and presentation tool for the desktop
- Some understanding of functions and expressions used to define attributes
- Some exposure to or experience with SQL programming language

## How this book is organized

This book introduces the QMF Visionary Studio interface and shows you how to build your own QMF Visionary applications, called *worlds*. Because QMF Visionary Studio is a creative environment, this book provides guidelines rather than comprehensive reference to all procedures you can perform using this tool. For additional detailed procedures, refer to the QMF Visionary Studio online help and the *DB2 QMF Visionary Getting Started Guide*. For a lesson-based introduction to building QMF Visionary worlds, refer to the

QMF Visionary online tutorial. QMF Visionary Studio online help and the tutorial are accessible from the Help menu.

This book includes the following chapters:

- Chapter 1, "Concepts in QMF Visionary Studio," on page 1, provides an overview of the process of developing a QMF Visionary world.
- Chapter 2, "Designing a QMF Visionary world," on page 5, describes how to plan your QMF Visionary world. Planning allows you to determine your design requirements and to consider design alternatives.
- Chapter 3, "Preparing your data," on page 27, describes how to create filtered views of your database data—called *workbooks*—and how to create SQL queries for your world.
- Chapter 4, "Creating scenes," on page 43, describes how to create a scene using the Scene editor and other QMF Visionary Studio tools, as well as how to create a dynamic scene. The chapter also describes advanced layout features and other graphic enhancements.
- Chapter 5, "Creating navigation," on page 63, describes the navigation features you can add to a scene to enable your users to travel between scenes and zoom in for more detailed data. The chapter also describes how to maneuver through your world as you develop it.
- Chapter 6, "Custom graphic features," on page 77, describes how to add custom graphic features, such as ActiveX controls and color maps, to enhance your worlds visually or provide greater functionality.
- Chapter 7, "Object properties and events," on page 91, describes the object-based structure of worlds and how to change objects by changing their properties and how to add events to trigger actions to events.
- Chapter 8, "Managing and publishing worlds," on page 107, describes how to manage worlds using the World Structure editor and how to publish worlds.

## Major components of QMF Visionary

This book is written for users of the QMF Visionary Developer package, which contains the following QMF Visionary applications:

- **QMF Visionary Studio** The authoring platform for developing and publishing worlds.
- **QMF Visionary WorldView** A stand-alone application for viewing and navigating published worlds.

This book is written primarily for users of QMF Visionary Studio and is an authoring guide. Other product components are discussed in this book only in the context of developing worlds using QMF Visionary Studio.

## System requirements and software dependencies

See the *ReadMe* file for a list of operating systems on which QMF Visionary Studio is supported and a list of database servers that QMF Visionary supports.

## Conventions used in this book

This book uses the following highlighting conventions:

- **Boldface type** indicates user interface controls such as name of fields, icons, or menu choices.
- `Monospaced type` indicates system messages, command syntax, and examples of text that you enter exactly as shown.
- *Italic type* indicates variables that you should replace with a value. It is also used to indicate book titles and to emphasize significant words.

## Additional documentation

QMF Visionary documentation is provided in a variety of formats:

- **Documentation**. The documentation set for QMF Visionary includes the following documents, in addition to this book:
  - *DB2 QMF Visionary Getting Started Guide*
  - *DB2 QMF Visionary Studio Quick Reference*
- **DB2 QMF Visionary Tutorial**. This tutorial is available from the QMF Visionary Studio Help menu.
- **Online help**. This facility provides both general and context-sensitive help.
- **ReadMe file**. This file is located in the directory where the product is installed. Examine this file carefully because it contains vital information about application and performance issues.

## How to order DB2® QMF books

To order hard copies, contact your IBM® representative or visit the IBM Publications Center on the world wide web at:

http://www.elink.ibmlink.ibm.com/applications/public/applications/publications/cgibin/pbi.cgi. Or, you can call 1-800-879-2755 in the United States or any of its territories.

## How to send comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book, go to http://www.ibm.com/software/data/qmf/support.html, and click on Feedback.

# Chapter 1. Concepts in QMF Visionary Studio

This chapter provides an overview of the process of developing a QMF Visionary world. This chapter is based on the assumption that you have basic knowledge of QMF Visionary concepts. For an overview of QMF Visionary concepts, see the *DB2 QMF Visionary Getting Started Guide*.

In this chapter:

- QMF Visionary world development overview
- Concurrent development

## QMF Visionary world development overview

To create and deploy a QMF Visionary world, complete the following major tasks:

- Design the world

  Includes identifying your business questions, creating a data model, designing scenes and navigation, and creating a storyboard.
- Create the world

  Includes establishing connections to data sources, creating a workbook, creating queries, creating scenes, adding objects to scenes, formatting objects, adding navigation features, and mapping objects to data.
- Publish the world

  Includes specifying the deployment configuration, user authentication, database connection options, limiting returned rows, and testing the compiled world.
- Deploy the world

  Includes installing the appropriate database connectivity on each client computer and making the published world file available on a networked drive or on each client computer.

The following table shows the major development tasks, the QMF Visionary tools you use to complete them, and where each task is documented.

*Table 1. Major development tasks*

| Major task | QMF Visionary tools | More information |
|---|---|---|
| Design the world | N/A | Chapter 2, "Designing a QMF Visionary world" |

*Table 1. Major development tasks (continued)*

| Major task | QMF Visionary tools | More information |
|---|---|---|
| Create the world | QMF Visionary Studio | Chapter 3, "Preparing your data"<br>Chapter 4, "Creating scenes"<br>Chapter 5, "Creating navigation"<br>Chapter 6, "Custom graphic features"<br>Chapter 7, "Object properties and events" |
| Publish and deploy the world | QMF Visionary Studio | Chapter 8, "Managing and publishing worlds" |

## Concurrent development

Multiple developers can work on different parts of the same QMF Visionary world simultaneously. When you open a QMF Visionary world, QMF Visionary Studio loads all of that world's files.

QMF Visionary worlds consists of the following file types:

- **Project (.vpx)**. Contains world-level attributes, global resources, and references to all component files.
- **Workspace (.vwx)**. Contains information about the active state of QMF Visionary Studio's user interface and personalized project settings for a user. This file is created and saved on each user's local view or file system.
- **Component (.vcx)**. Contains information about a single scene, query, or stock image. The filename for each component file is based on the name of the corresponding scene, query or stock image.

To avoid component naming conflicts, each world must have its own directory. You cannot save a world in a directory that already contains a QMF Visionary world.

Multiple developers can work on the project at the same time, as long as they are not working on the same component files or modifying the project file.

Modifications to the project file include:

- **Components**. Adding, deleting, or renaming a component or its filename
- **Global resources**. Adding, deleting, or altering a global parameter, color map, color sequence, color scheme, stock image, or viewer class.

## Saving and inserting components

When you save a world, you can choose to save only those components you have been working on. To save a component, right-click its name in the World Manager and select **Save** *Scenename*. To select multiple components to save at once, click **File —> Save**.

You can use a component in more than one world, for example, you might use the same stock image in several worlds. To use a component from another world, click **Insert —> Component** and select the component. When you save the world, the new component is saved in the same directory.

## Using a source code control system

You can use a source code control system to manage concurrent development. QMF Visionary Studio integrates with any commercially available source code control system.

When you use a source code control system, you typically put the world project file and the world component files in it, but not the world workspace file. The world workspace file is not shared among users; it is stored locally for each user.

## Multiple-developer file status indicators

For each developer of the same QMF Visionary world, QMF Visionary detects changes other users make to that world and issues alerts in QMF Visionary Studio:

- If another user saves a world project or component file that you are viewing in QMF Visionary Studio, QMF Visionary displays a message indicating that the file has been modified and asking you if you want to reload it.
- If you have already edited (but not saved) a component that is saved by another user, QMF Visionary asks you if you want to reload the file and lose the changes you have made.
- If you attempt to save a world project or component file that has been altered since you checked out the file, QMF Visionary displays a message box indicating that the file has been modified and asks if you want to lose your changes or overwrite the other change.

# Chapter 2. Designing a QMF Visionary world

This chapter describes how to plan your QMF Visionary world. Planning allows you to determine your design requirements and to consider design alternatives. Planning can also speed development work. This chapter also discusses the planning tool called a *storyboard*.

In this chapter:

- Identifying business questions
- Creating a storyboard
- Displaying data
- Using primitive graphics
- Using parameters
- Using navigation
- Using controls
- Design tips
- Storyboard example

These sections are provided to assist you in choosing the features you want for your QMF Visionary world.

## Identifying business questions

Before you create a QMF Visionary world, identify your business questions. What does your Board of Directors, business associate, or customer want to know? Here are some examples:

- How do gross sales fluctuate over the year?
- How do sales for each store location vary?
- How do sales of individual types of merchandise vary for each store?
- How do sales of specific merchandise correlate with advertising spending? (Which of our advertising plans is actually bringing in more receipts? Which is the most cost-efficient?)
- Is there merchandise we should discontinue because it is not selling well or the profit-margin is too low?
- Are we ordering enough of a popular product for a particular location?
- What is the organizational structure of a particular department?
- How many hits does our Web site get every day?

All of these questions can be answered and monitored in a QMF Visionary world. Talk to the customers of your QMF Visionary world and determine early in the process what questions you need to answer.

**Note:** Stay flexible about your design; the design typically changes as you work on a world. You might discover that there are other important business observations that might enhance your world. Or you might discover QMF Visionary features that enhance your world's navigation, appearance, or interactivity.

## Creating a storyboard

To begin planning your QMF Visionary world, you can create a *storyboard*. A storyboard is a plan for a media presentation, and it usually shows the basic contents and flow of the presentation. Like an application developer uses a functional specification, you use a storyboard to describe the basic features and functionality of your world.

For example, suppose that the QMF Visionary developer gathered information and determined the following key business questions:

- How are store locations performing throughout the United States?
- How did each store perform over the last twelve months?
- For each store, which product categories are most and least profitable?

To display this in a QMF Visionary world, the developer decided to create three linked scenes, as shown below.

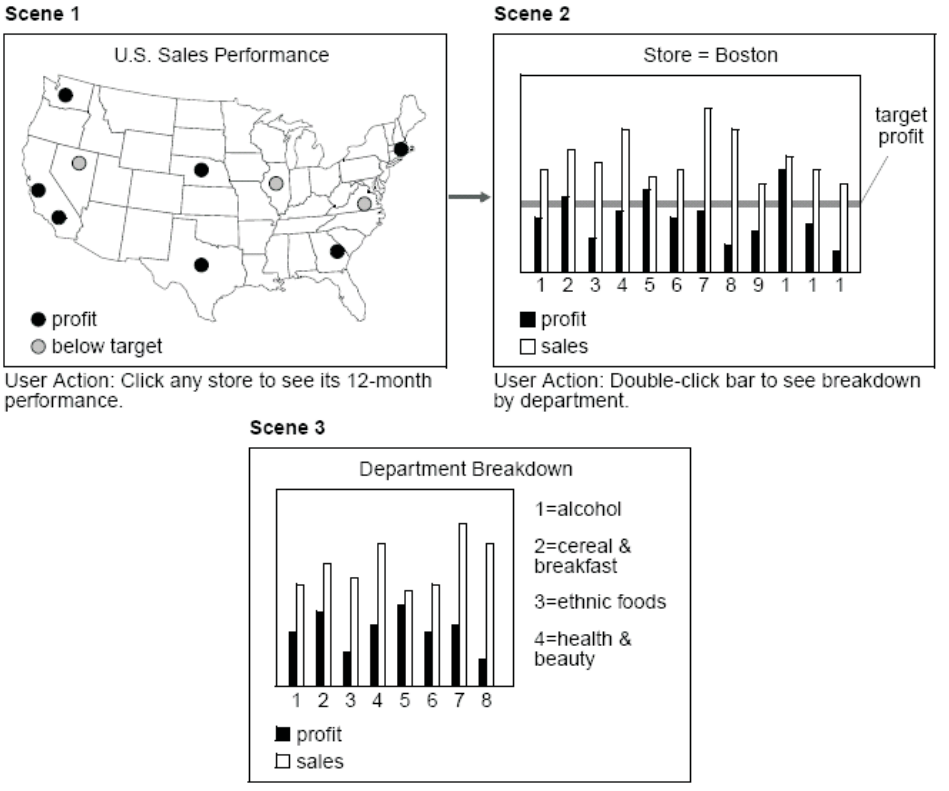



*Figure 1. Grocery store storyboard*

Scene 1 presents the big picture by showing a map of all stores in the United States and uses a code to show which stores are meeting their target profits. To jump to Scene 2, the user can double-click one of the store icons on the map.

Scene 2 contains a bar chart showing the sales and profits for each of the previous twelve months and a line that represents the target profit for that twelve-month period. To jump to Scene 3, the user can click a bar in the chart.

Scene 3 contains another bar chart, this one showing the sales and profits for one store during one month, broken down according to the department.

The QMF Visionary developer might later add more detail to the world to provide the means for analyzing additional business factors such as demographic differences between customers at different store locations, how

special events such as promotions or holidays affect sales of particular products, or how staff incentives affect overall performance at stores.

To create a useful and comprehensive storyboard for a QMF Visionary world, you should specify the following:

- The objects in the scene
- How users navigate between and within scenes
- How database data is presented in layouts
- Points where users can access further information
- Actions that result from user events
- Keys for symbols used

You can use both simple and sophisticated tools to create storyboards. You can sketch one using pencil and paper. You can use a presentation tool like PowerPoint to create sample layouts, with supporting detail on notes pages. You can also put together Web pages to show actual jumps and navigation flow.

**Note:** Consider using a separate sheet of paper for each scene so that you can easily rearrange the order of scenes. If you are using pencil and paper to create your storyboard, consider using sticky notes so you can easily rearrange elements in each scene.

When creating a storyboard, you might find it helpful to perform the tasks described in the following sections:


- "Listing the categories for your presentation" on page 8
- "Exploring QMF Visionary layouts, controls, and navigation features" on page 9
- "Locating the data" on page 9
- "Creating a query/layout matrix" on page 9
- "Exploring your data" on page 10


## Listing the categories for your presentation

As you look at the business questions you want to answer in your QMF Visionary world, determine the categories of data represented by those questions. For example, the data categories for a retail world would include:

- Time (months, quarters, weeks)
- Locations (store names, states, cities, zip codes, latitude and longitude)
- Merchandise (types, departments, names, item codes, vendor or manufacturer names)
- Customers (demographics, locations)

These categories will help you decide what queries you want to write, and also which data will be displayed in which layouts.

## Exploring QMF Visionary layouts, controls, and navigation features

As you generate the notes that will become your storyboard, think about how you want to display your data, what actions you want to provide for your users, and how your users will navigate through the world:

- Displaying data

  You display a single column of SQL data using primitives and other objects. To display more than one column of SQL data, use 2-D layouts or the List and Combo controls. To display other data, use 3-D ActiveX control layouts. Explore the different display options that QMF Visionary provides and decide which ones best suit the kind of analysis you want to show.

  For more information, see "Displaying data" on page 10..

- Capturing user input

  You can capture user preferences with standard Windows controls. Look at the Controls palette on the Palette Manager to explore your options.

  For more information, see "Using controls" on page 23.

- Passing information

  You can pass information acquired from user actions to affect the display of a world, the contents of a scene, or the execution of a query.

  For more information, see "Using parameters" on page 19.

- Navigating around worlds

  You can use navigation features to allow users to move around worlds to get to new information.

  For more information, see "Using navigation" on page 21.

## Locating the data

In many cases, your business data is stored in several database tables and perhaps in several databases. After you have determined the data you want for your QMF Visionary world, make a list of the databases, tables, and perhaps the columns in those tables, that you intend to query.

The list also makes it easier for you to create *workbooks* (filtered views of your databases). See "Using workbooks" on page 30 for information about creating and using workbooks.

## Creating a query/layout matrix

Because data and how it is displayed are so closely linked, one of the best ways to plan the layouts for your worlds is to plan your queries and layouts in tandem.

Some people work better by drawing the graphs and charts they want and then composing queries to supply the data. Other developers might work better by composing queries first.

You can create a query/layout matrix, or spreadsheet, to keep track of each query and its associated layout. Again, your design might change as you try different layouts. The queries also might change as you add features that require query parameters, or when you discover a need for different data from your database.

### Exploring your data

You might find it useful to connect to a database and explore your data using QMF Visionary's query tools. Using workbooks and Query wizards, you can retrieve broad result sets of data and discover how you might want to narrow your queries.

For example, you might retrieve all grocery departments to see how many departments there are. You might decide to separate specialty departments such as liquor and gifts from staple departments such as produce and meats: this allows you to tighten the scope of each layout and prevent information crowding.

Or, you might want to retrieve sales revenue totals for all stores and determine the range of totals before you decide what upper and lower limits you want to specify in a bar chart.

**Note:** Limit the number of categories per layout to a manageable size. As you gain experience, you will know better what an optimal number of categories is for each type of layout.

## Displaying data

You can display data from your database queries using the following categories of QMF Visionary objects:

- **2-D layouts**. Includes charts, patterns, hierarchies, forms, and maps. Layouts contain data templates that determine how data points are displayed. You can edit data templates to customize the appearance of layouts.
- **List and Combo controls**. Controls do not contain data templates. You have limited ability to alter the display of data points in controls.
- **3-D ActiveX control layouts**. Includes many 3D charts.

You might know precisely which QMF Visionary layout or control you want to use, or you might want to try out your data in different formats. QMF

Visionary allows you to explore your data—and the business implications of the data—by trying different layouts or controls for the same query.

For example, you might want to show how each type of a product sold in your business contributes to the overall revenue. A pie chart is a good way to show shares, or percentages, of the sum total.

If you then display the units sold and shipped from each category in a bar chart, you can draw conclusions about whether your shipping costs might be increasing in the future. You can create an XY chart to see the shipping cost trend line and make a logical prediction.

This section provides a guide to the type of business analysis scenarios you might want to display and the QMF Visionary layouts and controls available to you to create your visual analyses. The Layout wizard helps you create each layout and control, and online help for the wizard pages provides extra assistance and tips on how to complete each.

The following objects are available to display data from SQL queries:

- All layouts on the Layouts palette
- The List and Combo controls on the Controls palette

The controls on the ActiveX Controls palette. The following objects are available on the ActiveX Controls palette to display data from queries:

- Pie3D
- Pie3DSmooth
- Torus3D
- Bar3D
- Column3D
- Oblique3D
- Area3D
- Ribbon3D
- Surface3D
- Point3D
- Bubble3D
- Stackbar3D
- StackColumn3D
- StackLine3D
- StackArea3D
- Stock3D
- FloatBar3D

- Radar
- Polar
- Shmoo3D
- Legend3D

## Displaying data using charts

QMF Visionary provides charts for SQL data allow you to display trends, compare different categories of items, to compare one trend with another, or to compare the various parts of an entire revenue or expenditure category.

*Table 2. QMF Visionary charts*

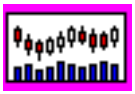
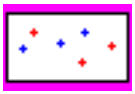
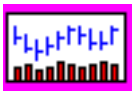
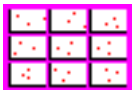
| Layout and Icon | Palette | Description |
|---|---|---|
| Candlestick chart  | Layouts | Displays gains and losses in stock prices, or other value-based indexes, and volumes over time. The x-axis represents time, and two y-axes represent price and volume. The color of the candlestick shows either gain or loss: typically, green for gain and red for loss. |
| Scatter chart  | Layouts | Displays data in a two-dimensional graph in rectangular coordinates. You can use the scatter chart for statistical analyses, such as correlation studies. |
| Stock chart  | Layouts | Displays fluctuations in stock prices, or other value-based indexes, and volumes over time. The x-axis represents time, and two y-axes represent price and volume. |
| Timeline chart  | Layouts | Displays a time series. The x-axis is a date-based axis, and the y-axis can be assigned to a numeric value. You can use the timeline chart to show a trend among data values returned in a sorted order by date. |
| XY chart  | Layouts | Displays data in a two-dimensional graph; data points are connected by a line, in the order the data was retrieved. You can use XY charts to show correlations between two variables and trends in these correlations. |
| Multivariate chart  | Layouts | Displays several scatter charts in a grid. You can use the multivariate chart to analyze three or more independent variables. |

*Table 2. QMF Visionary charts (continued)*

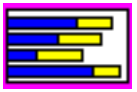
| Layout and Icon | Palette | Description |
|---|---|---|
| Bar chart  | Layouts | Displays horizontal columns, or bars, to represent values measured on the x-axis that correspond to categories displayed on the y-axis. You can use bar charts to analyze trends over time or to compare quantities of different items. |
| Bar3D chart  | ActiveX Controls | Displays horizontal columns, or bars, to represent values measured on the x-axis that correspond to categories displayed on the y-axis. You can use bar charts to analyze trends over time or to compare quantities of different items. |
| Column chart  | Layouts | Displays vertical columns, or bars, to represent values measured on the y-axis that correspond to categories displayed on the x-axis. You can use column charts to analyze trends over time or to compare quantities of different items. |
| Column3D chart  | ActiveX Controls | Displays vertical columns, or bars, to represent values measured on the y-axis that correspond to categories displayed on the x-axis. You can use column charts to analyze trends over time or to compare quantities of different items. |
| Pie chart  | Layouts | Displays a circle cut into wedges; each wedge represents one row returned from a query. You can use the pie chart to examine shares or percentages of a whole. |
| Pie3D chart  | ActiveX Controls | Displays a circle cut into wedges; each wedge represents one row returned from a query. You can use the pie chart to examine shares or percentages of a whole. |
| Pie3DSmooth chart  | ActiveX Controls | Displays a circle cut into wedges; each wedge represents one row returned from a query. You can use the pie chart to examine shares or percentages of a whole. |
| Area3D chart  | ActiveX Controls | Displays areas that begin from the bottom of the chart with drops in the area. You can use Area 3D charts for examining change over time. |

*Table 2. QMF Visionary charts (continued)*

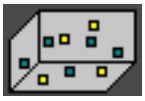
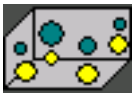
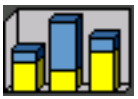
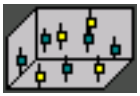
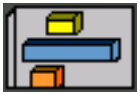
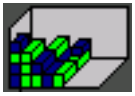
| Layout and Icon | Palette | Description |
|---|---|---|
| Oblique3D chart  | ActiveX Controls | Displays one or more vertical bars along a labeled axis with one or more planes. |
| Torus3D chart  | ActiveX Controls | Displays data in a 3D chart in the shape of a doughnut. Each piece of the torus represents one data point. You can use the torus chart to examine shares or percentages of a whole. |
| Ribbon3D chart  | ActiveX Controls | Displays one or more horizontal ribbons along a labeled axis with drops in the ribbons. |
| Point3D chart  | ActiveX Controls | Displays data values as points in the chart. |
| Surface3D chart  | ActiveX Controls | Displays a surface that connects adjacent data values. |
| Bubble 3D chart  | ActiveX Controls | Displays data as bubbles in the chart. The size of the bubble is determined by a data value. |
| StackBar 3D chart  | ActiveX Controls | Displays one or more horizontal bars that consist of stacked data values. |
| StackColumn 3D chart  | ActiveX Controls | Displays one or more vertical bars that consist of stacked data values. |
| StackLine 3D  chart | ActiveX Controls | Displays one or more horizontal ribbons that consist of stacked data values. |

*Table 2. QMF Visionary charts (continued)*

| Layout and Icon | Palette | Description |
|---|---|---|
| StackArea 3D chart  | ActiveX Controls | Displays an area that consists of stacked data values. |
| Stock 3D chart  | ActiveX Controls | Displays one or more vertical bars on a labeled axis. The stock 3D chart shows open, high, low, close, trading volume, and trading data for a stock. |
| FloatBar 3D chart  | ActiveX Controls | Displays floating horizontal bars. This chart displays time ranges and is useful for creating charts on schedules. |
| Radar chart  | ActiveX Controls | Displays multi-series data points in a radar screen format. |
| Polar chart  | ActiveX Controls | Displays data points in a 360-degree radar screen format. |
| Shmoo 3D chart  | ActiveX Controls | Displays multi-series data in a cube-like format. Each cell of the cube is tested to determine if it passes a given condition. Pass and fail cells are color coded. The first set of failed cells can be highlighted. |
| Legend 3D  | ActiveX Controls | Displays a legend in a 3D format. The legend can display multiple rows and columns with different mark shapes and controlled legend cell size. |

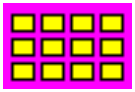
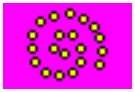
In addition to these layouts, you can also create comparisons by adding layers to some 2-D layouts. *Layers* are additional data templates added to a single layout: for example, an XY chart with more than one line representing more than one set of query results. For more details on layers, see "Using layers in a layout" on page 53.

## Displaying data using patterns

QMF Visionary provides various pattern layouts you can use to display SQL data in a visually appealing way. Patterns, such as a matrix, are typically used to create a gallery of objects, such as images or logos.

For example, your enterprise might own several subsidiaries, each with a distinct corporate logo and trademark. You might want to display this information in a QMF Visionary world that describes the entire business organization. You could enhance this kind of data display by adding levels of detail, so that viewers could drill down and learn more about each company. See “Creating levels of detail in a data template” on page 71.

*Table 3. QMF Visionary patterns*

| Layout and Icon | Palette | Description |
|---|---|---|
| Horizon pattern  | Layouts | Displays data values in decreasing sizes into an infinite horizon. Zooming in reveals more data in the distance, and zooming out shows more data closer to the viewer. You can use the horizon pattern when you want to display data in an appealing way, but without mathematical relation or analysis. |
| Matrix pattern  | Layouts | Displays data in a two-dimensional array of cells. You can use the matrix pattern to display data values as if they were in a gallery. |
| Spiral Pattern  | Layouts | Displays data in an infinite, inward spiral in which each data value becomes smaller and smaller. Zooming in reveals more data further into the spiral, and zooming out shows more data on the outer portion of the spiral. You can use the spiral pattern when you want to display data in an appealing way, but without mathematical relation or analysis. |
| Table pattern  | Layouts | Displays a bordered table consisting of rows and columns; similar to an HTML table. You can display the results of a query or enter data manually in each cell. You can include a title, headers, and footers, if desired. **Note:** Use the Table menu to perform tasks such as inserting rows, columns, or cells. |

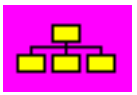
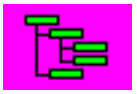
## Displaying data using hierarchies

A common hierarchy in business is an organization chart. However, there are other ways that hierarchies can help you visually analyze SQL data.

For example, you can use a hierarchy to examine how various ingredients contribute to each spending area in your business. You might also want to look at broad demographics for your sales and then see a breakdown of each demographic group's buying.

Hierarchies allow you to see data in groups and subgroups. There is no mathematical analysis comparing one group to another, but you can explore the constituents of any group.

*Table 4. QMF Visionary hierarchy layouts*

| Layout and Icon | Palette | Description |
|---|---|---|
| Cluster graph  | Layouts | Displays data in a recursive and circular set of hubs with spokes. Each hub represents a data point; the spokes show the relationship between the hub value and its satellites. Cluster graphs automatically expand as the viewer of the layout zooms in. |
| Organization chart  | Layouts | Displays data in a tree structure that reads from top to bottom. Optional expander buttons allow users to expand or collapse levels of the chart. You can use the organization chart to show a cascading set of one-to-many relationships, such as a personnel hierarchy. |
| Tree chart  | Layouts | Displays data in a tree structure that reads from left to right. You can use the tree chart to show a cascading set of parent-child relationships. Levels on the tree chart can be manually expanded by clicking expander buttons. |

## Displaying data using standard controls

You can use certain standard controls to display a single column of SQL data in a list from which the user can choose. In addition, you can associate another column with the control to set its value. Typically, when the user chooses a value, an event action passes information to a parameter, which then alters the display of the world. For example, you can present a list of years and have the user choose one. Then the chart in the scene changes to display the data for that year.

*Table 5. QMF Visionary standard controls*

| Layout and Icon | Palette | Description |
|---|---|---|
| Combo  | Controls | Displays a list of values from which the user can choose one or more. |
| List  | Controls | Displays a list of values from which the user can choose one. |

## Displaying a single record

Not all your SQL queries will return multiple rows of data from a table. You might want to display just one record, such as a single customer profile or a single URL to a vendor site. This is particularly helpful when you use levels of detail (see "Creating levels of detail in a data template" on page 71) to drill down on a single data point in a layout.

For example, you might create a healthcare QMF Visionary world that allows the viewer to zoom to a single X-ray in one patient's health chart. QMF Visionary provides the simple form as a way to display a single record using SQL queries.

*Table 6. QMF Visionary single record layout*

| Layout and Icon | Palette | Description |
|---|---|---|
| Simple form  | Layout | Displays data for a single row returned from a query. |

## Displaying geographic locations

Most businesses have a geographic aspect, whether it is where the products are shipped, store locations, or television program ratings in various cities. QMF Visionary provides the linear map so that you can display your SQL data geographically, either in coordinates on the map or in polygons or polylines on the map.

*Table 7. QMF Visionary geographic layout*

| Layout and Icon | Palette | Description |
|---|---|---|
| Linear map  | Layouts | A map that displays spatial data as coordinates, polylines, or polygons. |

## Using primitive graphics

Use primitive objects to add simple graphics to your scenes. Primitive objects can be static or bound to data from an SQL query, a parameter, or other calculated value. For example, you can use a text object to display a string from a table column, or you can use a property function to make the visibility of a rectangle depend on the value of a parameter.

QMF Visionary supplies the following primitives on the Primitives palette of the Palette Manager:

- Text
- Line
- Arrow
- Double Arrow
- Polyline
- Multipolyline
- Polygon
- MultiPolygon
- Rectangle
- Round Rectangle
- Ellipse
- Picture

For more information on these objects, see the QMF Visionary Studio online help. For examples of using primitive objects, see the QMF Visionary tutorial.

## Using parameters

QMF Visionary enables you to capture user information (or other information, such as a query result) and pass it to other parts of your world with parameters. A *parameter* is a value given to a variable in an operation or an expression before it is executed by the world.

Parameters enable you to do the following tasks:

- Capture user input.
- Capture current context, such as query results, object property values, or locations.
- Pass user input or context information from one part of a world to another.
- Vary query results based on dynamic criteria.
- Vary navigation results based on dynamic criteria.
- Vary visual display or other design elements based on dynamic criteria.

You can use parameters in many ways. Event actions typically capture user input in a parameter. You can use global and scene parameters when you design interactive controls in your scenes; when a user makes a selection in the control, an underlying event action sets the parameter value. The parameter value determines what data is displayed or how it is displayed.

QMF Visionary provides three types of parameters:

- Global
- Scene
- Query

Each of these types are described in this section.

### Using global parameters

A *global parameter* is available to all scenes in a world. Users can use global parameters (such as viewer classes) to change the display of a world. Use global parameters if you think you might use the parameter in several scenes.

A global parameter is available to all scenes in a world and to the end user. You can set global parameters when you specify object properties or when you specify event actions. For example, after you publish your world and you deploy it in a Visual Basic application, the defined global parameters determine the initial level of detail and the viewer class.

Global parameters can be public or private. A public parameter can be modified by the end user in the Runtime Settings dialog box or via the container application.

QMF Visionary provides four built-in global parameters: ViewerX, ViewerY, ViewerZoom, and ViewerClass.

### Using scene parameters

A *scene parameter* is available only for the scene for which it is defined. Use scene parameters if you want the display of only one scene to change. You typically set a scene parameter by an event or wormhole in another scene. When you specify a destination scene for an event or wormhole, you have access to the scene parameters defined in that scene. For more information on using scene parameters, see “Making scenes dynamic with parameters” on page 44.

### Using query parameters

A *query parameter* is a variable used in a query that is set at runtime before the query is executed. Use to change query criteria. For more information about using query parameters, see “Using query parameters” on page 37.

## Using navigation

*Navigation* in a world determines how scenes are connected to one another, and how the user can change the data display or access more information.

Start your navigation plan by specifying one layout for each scene and design a single overview scene that acts like a menu for accessing individual scenes. Later, you can consolidate some of the individual scenes into multi-layout scenes or hidden levels of detail contained in a single scene. From this general structure, you can determine routes—which scene is connected to which.

### An example of scene navigation

The following illustration charts the scene connection that constitute navigation for this world.

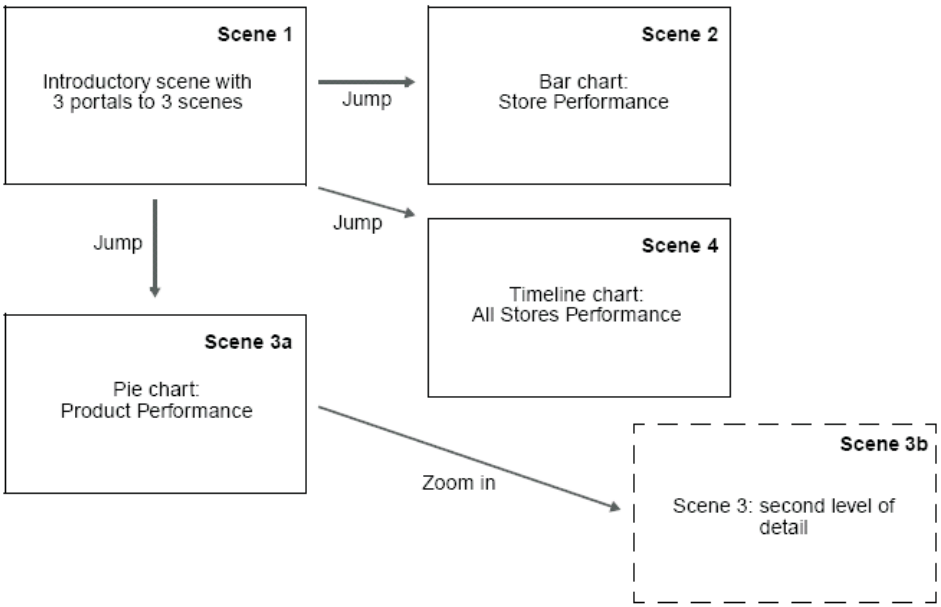



*Figure 2. Navigation diagram*

The two main navigation features that move you from scene to scene are *jumps* and *wormholes*.

Other navigation can change data display based on the user's interaction with an object. For example, you can create *levels of detail* so that as your users zoom in on data points, more detailed information is revealed. You can also create an *event action* that sets a parameter, so that the value of the *parameter* changes the type of data displayed (see "Using parameters" on page 19.

Your users can move around in your world using the pan tool, scroll bars, zoom in or zoom out tools, and activating navigation features you have designed: jumps, wormholes, levels of detail, or event actions.

### Guidelines for choosing navigation

You decide how you want the user to get to new information. Here are some guidelines for choosing navigation features:

- Is the new information in a different layout and different scene?

  Consider a jump or a wormhole.
- Why does the user want more information on this data?

  If they want to look closely at a single data point and see a single record, consider adding a scene level of detail that displays a simple form of that record.
- When would the user want more information on this data?

  Consider asking the user to make a choice using a RadioGroup control. The choice activates an event action that takes the user to a new layout or sets a new parameter that changes how the scene appears.
- Is the new information based on an existing query, but revealing more columns of data?

  Consider adding a level of detail to the data template so that the query can reveal general information in one level and focused or specific information in another.
- Does the user want to preview a scene before navigating to it: for example, via a menu showing what other types of data displays are in the world?

  Consider using wormholes to show the contents of other scenes and to provide a way to jump to those other scenes.
- Does the user want to get new data based on a condition, such as a particular type of customer?

  Consider setting a scene parameter using an event action. The click or other user interaction is the event, and setting the scene parameter is the action. In this case, you might specify a query parameter (such as customer type) as the scene parameter.

These guidelines are not exhaustive, but they give you a sense of the types of navigation features available and how to decide which you want to use and where in your world.

For a full description of QMF Visionary navigation features, see Chapter 5, “Creating navigation,” on page 63.

## Using controls

If you want your users to be able to perform actions in the QMF Visionary world, then you can use standard controls. *Standard controls* are objects that a user can manipulate to perform an action. *ActiveX controls* are small software components for displaying items like database data, documents, movies, Web pages, and text.

QMF Visionary provides the following standard controls (in addition to the List and Combo controls described in "Displaying data using standard controls" on page 17):

- Textbox
- Button
- Checkbox
- Radio Group
- Slider

Standard controls are on the Controls palette of the Palette Manager.

QMF Visionary provides the following ActiveX controls (in addition to the 3-D charts described in "Displaying data" on page 10):

- ActiveMovie
- WebBrowser
- RichTextBox
- Clock3D
- Text3D

ActiveX controls are on the ActiveX Controls palette of the Palette Manager.

The controls provided by QMF Visionary are useful in several ways:

- To capture user input and use it to set global, scene, or query parameters
- To allow users to view documents, movies, and Web sites
- To provide users with flexible viewing options

  For example, by selecting from a list of parameters for a scene, the user determines how a scene is displayed.
- To provide self-documenting navigation buttons

Controls are typically used with event actions and parameters.

For more information on using controls with event actions and parameters, see "Using controls" on page 77.

## Design tips

The following sections provide design tips that are helpful when planning your world.

### Using an overview scene

If you are planning a world with many scenes, consider creating an overview scene to orient users. An overview scene could contain a linked map of the world, simple instructions, controls providing navigation to other scenes, or wormholes displaying portions of other scenes.

For example, if you plan to provide three different charts showing quarterly, monthly, and daily sales performance, you might create an overview scene that lists these three options and provides jumps to each scene. Alternatively, you might create an overview scene that contains wormholes to the three charts. Each wormhole is like a window looking onto another scene; three wormholes provide a vivid menu of the charts provided by your world.

### Using wormholes for repeated elements

If you plan to use a title bar, menu bar, side bar, or other elements that will appear in many of your scenes, consider creating a scene containing the repeated elements and then insert a wormhole (without an associated jump) to that scene in all the other scenes. Make sure the wormholes do not have jump events (so that users do not go to the repeated elements scene), and turn off the visibility of the wormhole borders.

### Creating a printer-friendly scene

If you expect users to print a scene, consider creating a printer-friendly version of the scene that does not have elements the user does not want to print, such as menu bars, side bars, or titles. Then add a button named **Print Scene** to the original scene that takes the user to the printer-friendly scene.

### Using standard objects

If you are planning a world that uses specific objects, you can create the object define its specific properties and then it to use again. For example, you can define a graphic object for your company logo and define its height and width. This object can then be placed on every scene and provides consistency throughout your world.

## Storyboard example

You can use both simple and sophisticated tools to create storyboards. You can sketch one using pencil and paper. You can use a presentation tool like PowerPoint to create sample layouts, with supporting detail on notes pages. You can also put together Web pages to show actual jumps and navigation flow.

**Note:** Consider using a separate sheet of paper for each scene so that you can easily rearrange the order of scenes. If you are using pencil and paper to create your storyboard, consider using sticky notes so you can easily rearrange elements in each scene.

## Organization chart example

Suppose you want to create a world that displays each department in your organization. You want users to be able to choose the department they want to view. In addition, users should be able to zoom in on any employee in the chart to reveal a photo and more information about the individual, such as his or her phone extension.

You might create a storyboard like in the following figure.

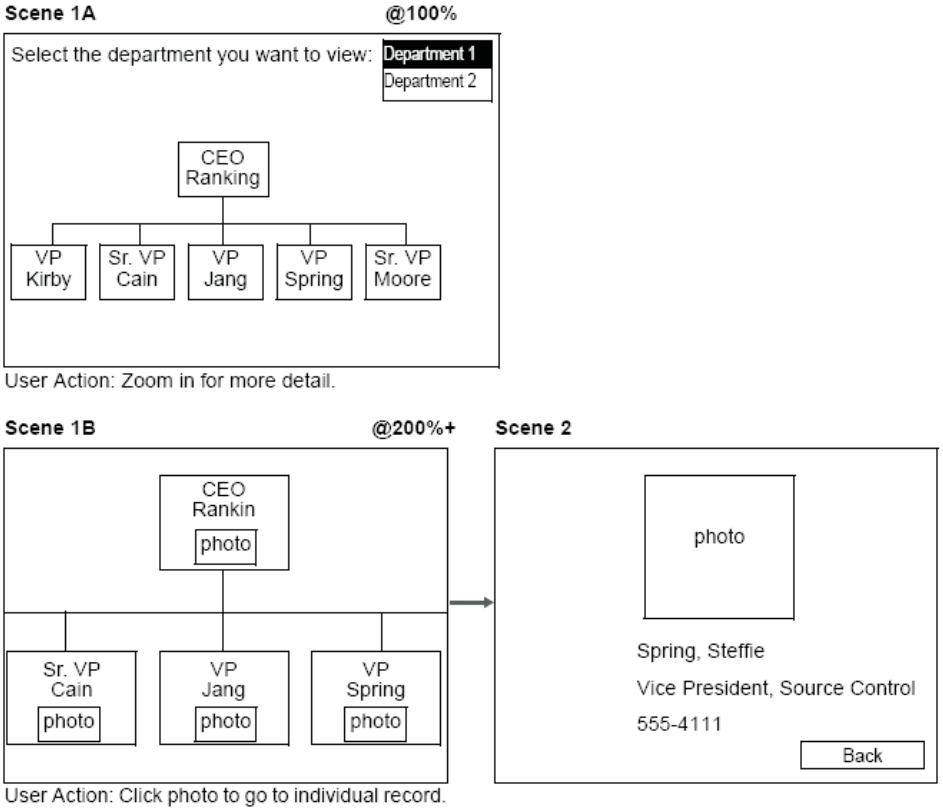



*Figure 3. Organization chart storyboard*

In this storyboard, there are two main scenes: the department organization and the individual record. Scene 1 consists of two levels of detail:

- At 100% zoom, you can see the names and titles of the employees in the department.
- At 200% zoom and higher, you can see the names, titles, and photos of employees in the department.

To isolate an individual employee record, you can click the individual's photo and go to Scene 2, which is dynamically generated using a scene parameter containing the employee name.

After you have designed the scenes, their graphical contents and levels of detail, and the navigation features for your world, you now have a detailed plan for creating the world in QMF Visionary Studio.

# Chapter 3. Preparing your data

This chapter tells you how to connect to data sources, how to create and use QMF Visionary workbooks for accessing and managing your relational data, and how to write SQL queries in QMF Visionary to retrieve data to display in your world.

In this chapter:

- Managing connections
- Using workbooks
- Creating queries
- Modifying queries

## Managing connections

QMF Visionary allows you to connect to various supported databases using Open Database Connectivity (ODBC), QMF, or both. You must have a data source defined for each database you intend to query from your QMF Visionary world. You can have queries to more than one database in the same world.

Refer to *ReadMe* file for a complete list of supported database servers and their compatible drivers.

### Configuring ODBC data sources

To configure an ODBC data source, click **Tools —> ODBC Data Source Administrator**. After you select the driver for your data source, the configuration options and steps are driver specific. If your configuration requires you to specify an authentication method for the data source, be sure that you specify the same method used by the database server.

You must configure your ODBC data sources as either system data sources (system DSNs) or user data sources (user DSNs). A system data source is defined for the computer and accessible to all users that log on to that computer. A user data source is defined only for a particular user on the computer.

ODBC data sources typically require the following types of information:

- Data source name and description
- Host name
- Server name

- Protocol
- Connection information (such as port number)
- User name and password

Additionally, for some data sources, you can select advanced options on data source configuration pages.

**Note:** Consider naming your data source to reflect your database name, unless you have a naming conflict among multiple databases you intend to use with QMF Visionary.

After you have configured one ODBC data source for a particular database server, you can autoconfigure additional data sources for other databases on that server, as described in the following section.

### Configuring QMF data sources

To configure a QMF data source, you use the QMF for Windows Administrator. See the QMF for Windows documentation for more information.

You cannot autocofigure QMF data sources.

### Autoconfiguring ODBC data sources

After at least one ODBC data source is configured for a particular database server, you can easily add data source definitions for other databases on that server. You must be connected to the host computer and the target databases must use the same driver as the pre-existing data source.

For example, you can autoconfigure Microsoft Access data sources based on an existing Microsoft Access data source on the same host computer. But you cannot autoconfigure a Microsoft Access data source based on an Informix data source on the same host computer.

After you select the driver for your data source, the configuration options and steps are driver specific. If your configuration requires you to specify an authentication method for the data source, be sure that you specify the same method used by the database server.

To autoconfigure data sources, connect to the data source on the machine that hosts the additional databases you want to configure and click **Tools —> Auto Configure Data Sources**.

QMF Visionary configures a data source for each compatible database it discovers. Each data source definition inherits the same user name and driver used in the initial data source that you defined manually.

QMF Visionary assigns each automatically configured data source a name based on the database name, as long as there are no naming conflicts. If there is a naming conflict, QMF Visionary appends a number to the data source name.

## Connecting to databases

To access data stored in databases and write queries to return data, you must connect to a database.

There are several ways to establish a connection from QMF Visionary. By default, when you launch QMF Visionary, the Select Data Source dialog box prompts you to connect to a data source.

**Note:** If you want to work offline or do not want to connect to a database each time you launch QMF Visionary, you can disable the automatic connection prompt at login. To disable the prompt, click **Tools —> QMF Visionary Options** and then clear the **Show the Select Data Source dialog when the application starts** option.

You can also manually open a connection in QMF Visionary (click **File —> Connect**). You might want to open another connection during the development of a world to connect to a second or third database and retrieve additional data.

## Managing connections

Database connections are associated with worlds. The connections associated with a particular world include:

- All currently open connections
- Any connection that is opened while the world is open in QMF Visionary Studio
- Any connection that has previously been open while the world was open

When you open a world, QMF Visionary Studio prompts you to connect to all of that world's connections. Typically, the connections for a world are those that the world actively uses with queries. You should delete unused connections.

## Using workbooks

A workbook provides a filtered view of your database, including only those database objects that you want to use in your QMF Visionary world. You can select tables, views, synonyms, functions, and procedures. The figure below illustrates an unfiltered view of a database versus a workbook.

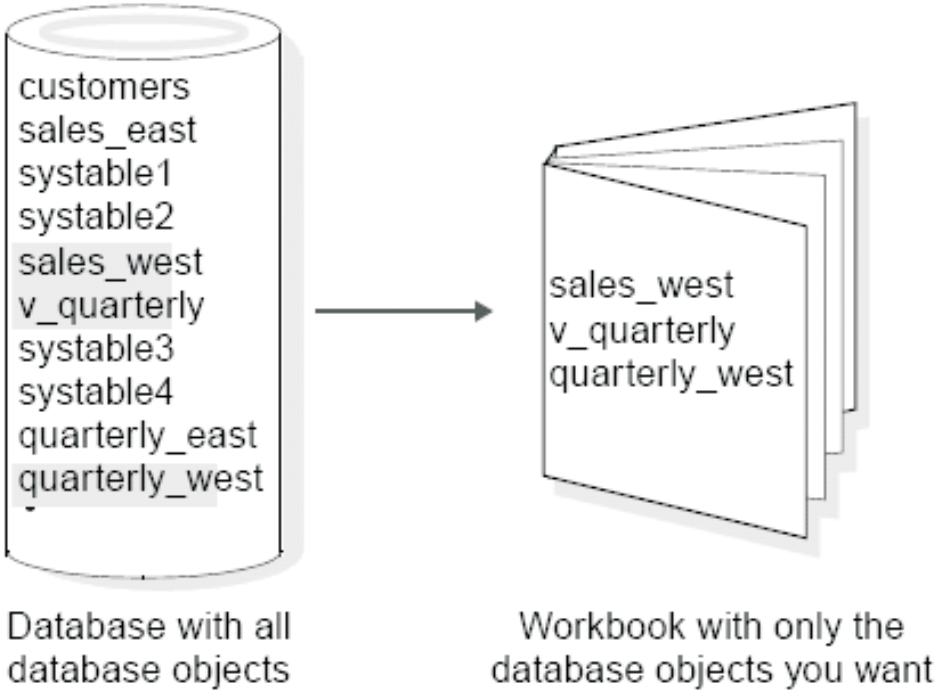



*Figure 4. Database vs. workbook*

Suppose you share a database with other departments of your company. The database probably contains some information that is not relevant to the business analysis you want to illustrate in your QMF Visionary world. Creating a QMF Visionary workbook allows you to tailor the view of the database to your focused needs.

There are three types of workbooks:

- Default
- Custom
- QMF

A default workbook provides an unfiltered view of the database. QMF Visionary automatically creates a default workbook when you connect to a

database. A default workbook is dynamic; QMF Visionary Studio re-creates it whenever you start QMF Visionary Studio instead of opening a saved workbook file. These workbooks can be quite large, with long lists of tables that make it difficult to find the tables you need. After you create a custom workbook, disable default workbooks by clicking **Tools —> QMF Visionary Options** and clearing the default workbook option on the Login page.

A custom workbook provides a filtered view of the database. You create a custom workbook using the Workbook wizard. You select only those tables, views, synonyms, procedures, and functions that are useful to you. For instructions on creating a custom workbook with the Workbook wizard, see the *DB2 QMF Visionary Getting Started Guide*.

A QMF workbook provides a list of the QMF queries for the selected QMF data source.

The figures below show two workbooks (on the Worlds page of the World Manager). The qademo workbook is a default workbook that shows the full contents of the database.

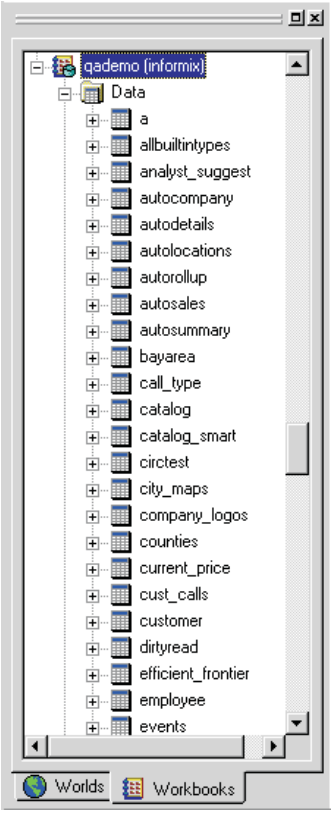


*Figure 5. Default workbook (qademo)*

The Auto_Summary workbook is a custom workbook that shows only those tables and views that are needed to create a QMF Visionary world that summarizes and analyzes auto sales.


*Figure 6. Custom workbook (Auto_Summary)*

## Managing workbooks

You can manage the way you view workbooks, what types of objects are included in workbooks, and the relationships between database tables.

Many workbook commands are available on the context menu for a workbook and its subfolders. See the *DB2 QMF Visionary Getting Started Guide* for more information.

Workbooks have the following types of options:

- General options include when to open and close the workbook, whether to prompt to save it, and whether to create the default workbook. These options are located on the Workspace page of the QMF Visionary Options dialog box (click **Tools —> QMF Visionary Options**).
- Display options sort objects and control the display of functions and data types in your workbook. Import options determine which database objects are selected by default in the Workbook wizard. You can set display and import options in the Workbook Options dialog box (right-click the workbook in the World Manager and click **Options**).

  See "Workbook display options" on page 33 and "Workbook import options" on page 34 for the lists of options.

### Workbook display options

Set display options on the Display page of the Workbook Options dialog box. Display options take effect immediately. The following table lists the display options you can modify for a workbook.

*Table 8. Display options for a workbook*

| Display options | Description |
|---|---|
| Sort data node by | Object type classifies data into subgroups of tables, views, and synonyms. Object name sorts tables, views, and synonyms alphabetically. |
| Show owners as tree nodes | Sorts data, functions, and data types according to the owner name and displays separate nodes for each owner. |
| Include owner name in tree | Displays the owner of the database object as part of the object name. |
| Show Data node in tree | Allows you to display the Data node in the workbook tree. If the check box is clear, the tables, views, and synonyms are displayed directly under the workbook name and do not reside in a Data folder. |
| Show types in tree | Displays the data types in the Workbooks page. |
| Show functions in tree | Displays the functions on the Workbooks page. |

**Note:** Object owner information can only be shown in individual workbooks if the corresponding general workbook option (**Show database object owner information when available**) is selected in the QMF Visionary Options dialog box. Click **Tools —> QMF Visionary Options** to verify or change this option.

### Workbook import options

Set import options on the Import page of the Workbook Options dialog box to modify the defaults used the next time the Workbook wizard is launched.

The following table lists the import options you can modify for a workbook.

*Table 9. Import options for a workbook*

| Display options | Description |
|---|---|
| Data objects | Select the object types to include. |
| Functions | Select the types of functions to include. |
| Additional options | Select the options to include: system objects, objects owned by a particular owner, primary and foreign key relationships, and inferred relationships. See "Managing table relationships in a workbook" on page 34 for a definition of any inferred relationship. |

### Managing table relationships in a workbook

You can use your workbook to define relationships between tables and use those relationships when creating query joins. Based on these table relationships in your workbook, the Query editor can automatically create visual links between columns when you insert related tables into the Query Diagram view.

When you import a workbook, you can include the primary and foreign key definitions in the tables. Columns defined as primary keys are identified in a workbook by a key icon, similar to the symbol primary key shown below.

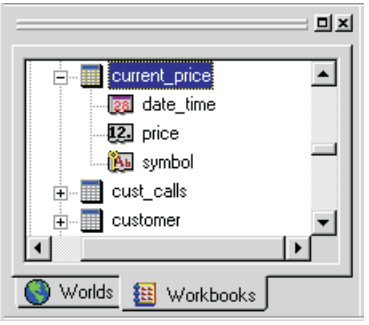



*Figure 7. Primary key column in a workbook*

**Note:** The table relationships you define in a workbook are stored in the workbook and are valid for creating queries in QMF Visionary; they are not created in the database.

You can define these types of table relationships:

- Inferred
- Explicit

An inferred relationship between tables occurs when one table contains a primary key definition and another table has a column of the same name and data type. You can create inferred relationships by selecting the **Infer relationships between tables** option when importing or creating a workbook.

Typically, tables in a database have explicit relationships based on the primary and foreign keys, which must be of the same data type. For example, a customer table might have the cust_id column as the primary key, and the orders table might have cust_id as the foreign key column. You can create explicit table relationships using the Relationships page of the Table Properties dialog box (right-click the table in the World Manager and click **Properties**).

## Creating queries

After creating a workbook, you can write queries to display data in your QMF Visionary world.

You must have a workbook open to use most of the query tools in QMF Visionary Studio. Query wizards and Query Diagram fields get their data from the selected workbook. You select the workbook you want in the New Query dialog box when you create a new query.

Workbooks are not required for creating queries in the SQL Text view of the Query Editor; you can type the query and run it with an open database connection.

### SQL query options

QMF Visionary Studio provides these query tools for creating SQL queries:

- **Simple Query wizard**. Creates simple queries based on a single table.
- **Advanced Query wizard**. Creates complex queries that can include joins, functions, filters, and other advanced elements.
- **Query editor**. Contains these views for creating and editing SQL queries:
  - **Query Diagram view**. Provides grid-based support for composing SQL statements (for developers familiar with SQL). QMF queries cannot be displayed in the Query Diagram view.
  - **Text view**. A text editor with color-coding for syntax elements. Use this view to modify queries if you are a skilled SQL programmer, to add SQL clauses that the Query Diagram view does not support, or to paste copied queries from other applications.

The Query editor also contains a Data Sheet view for viewing the results of your queries.

The following table lists SQL options and which QMF Visionary query tool you can use to include them in your queries.

*Table 10. QMF Visionary query tool support*

| SQL option | Simple Query wizard | Advanced Query wizard | Query diagram | Text view |
|---|---|---|---|---|
| Simple table | Yes | Yes | Yes | Yes |
| Multiple tables (joins) | No | Yes | Yes | Yes |
| Aggregates | Yes | Yes | Yes | Yes |
| Functions | Yes | Yes | Yes | Yes |
| Parameter | No | Yes | Yes | Yes |
| GROUP BY clause | No | Yes | Yes | Yes |
| Filters (WHERE clause) | No | Yes | Yes | Yes |
| HAVING clause | No | Yes | Yes | Yes |
| ORDER BY clause | No | Yes | Yes | Yes |
| UNION clause | No | No | No | Yes |
| INTO TEMP clause | No | No | No | Yes |

Aggregates include statistical functions such as SUM(), AVG(), and so on. Functions include operators, such as =, >, and <, as well as any functions you might have imported from the data source into your workbook.

QMF Visionary Studio online help and the *DB2 QMF Visionary Getting Started Guide* provide details about the individual pages of the query wizards.

## Query writing tips

Here are some tips for writing queries:

- **Examine all data in the table**. Use the Simple Query wizard and select all columns. After you execute the query and see all the data in the Data Sheet, you can decide which rows you want to display in your layout and how you want to narrow your queries.

  If your table is too large to view in the Data Sheet view, you might want to increase the number of rows in the World Settings dialog box, Data Control page (click **Tools —> World Settings**).
- **Use logical names for aliases**. You must specify aliases for calculated values, such as column values you used to determine an average or a

function result. Because you are likely to display these values in your QMF Visionary layouts, it is helpful to give them a logical name.

- **Name your query**. The default name for queries is Query*n*. It is helpful to give the query a meaningful name so you can identify its purpose. For example, a query that returns all profits from all stores in a retail chain might be named Profits_All_Stores.
- **Write filter expressions**. You can restrict the data returned from the database by writing filter expressions.

## Writing filter expressions

You can write filter expressions to modify data returned from a query using the Advanced Query wizard and the Query editor. You can filter row data (one specific record in your database), group data (all grouped records), and all data results from the query.

Filter expressions are equations that include one or more of the following elements:

- **Columns**. All columns selected for the query are available.
- **Operators**. Includes =, >, >=, <, <=, and <>.
- **Functions**. Includes all database server functions imported into the workbook.
  - General functions return a calculated value based on a column of the appropriate data type.
  - Filter functions return a Boolean value based on arguments of the appropriate data types.
- **Query parameters**. Values assigned to variables set at runtime, before the SQL query is executed.

The typical format for a filter expression is *variable=value*, where *variable* can be a function expression, column, or query parameter; *value* can be a constant or query parameter; and = (equals) can be any operator.

**Note:** If you use the Advanced Query wizard or the Query Diagram view, you should not include the WHERE keyword in your filter expression; QMF Visionary includes it automatically.

For example, you might want to see sales data for a particular year. To do so, you can create a query of all sales totals that filters the results so only sales above specific amounts are in the result set: sales > 50.

## Using query parameters

Use a *query parameter* to create a dynamic query. When you include a query parameter, your result set varies, depending on the value of the parameter at runtime.

Dynamic queries are useful when you want to perform one of the following tasks:

- Provide your users with display options

  You can capture the user's choice and pass it from a scene or global parameter to a query parameter, thereby changing the filter for the query results.
- Display data depending on some threshold or condition

  You can use a query parameter to specify the threshold for some object property, such as Visibility or Color. Then, at runtime, if the threshold or condition is met, the appearance of the object is changed.

Parameters can provide you with many ways to transform your analysis of data into an intuitive visual display for your users.

The most common way to use a query parameter is to have it supply the contents of a filter. For example, `WHERE columnname = :Q_filter`. Q_filter is then set to different values. Use the Advanced Query wizard or the Query Diagram view to insert query parameters as filters.

You can also use query parameters to supply:

- A predicate

  For example, `WHERE :Q_predicate`. Q_predicate is then set to different versions of column = value. Use the Query Diagram or Text view to insert these parameters.
- A column or table name

  For example, `FROM :Q_tablename aliasname`. Q_tablename is then set to different table names. Use the Text view to insert these parameters. Define the parameter as a literal type and include an alias for the column or table name.

### Creating query parameters

You can create a query parameter in the Advanced Query wizard (click the **Define Parameter** button) or in the Query Diagram view (right-click the Parameters block and click **Define Parameter**). To create a query parameter, you name it, specify a data type, and in most cases, specify a default value. You then use the parameter in a query.

**Note:** By using a consistent naming prefix, you can recognize which scene, global, and query parameters you are passing values from and to. For example, you can name query parameters with a Q_, scene parameters with an S_, and global parameters with a G_. Then, to pass a company name from a scene parameter to a query parameter, you might name them S_company and Q_company.

### Setting query parameters

To set a query parameter, select the DataTemplaten.QueryParameters object from the Object Inspector list and then edit the value cell for the parameter, as shown below.

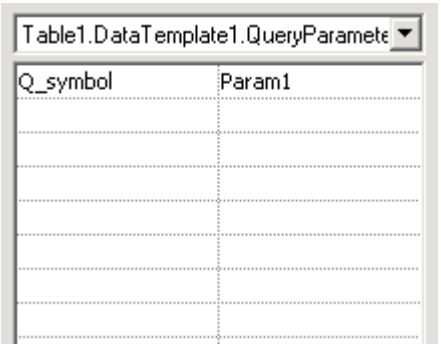



*Figure 8. Setting a query parameter*

You can set a query parameter to a constant, a functional expression, or the value of another parameter, such as a scene or global parameter. For example, to pass information from a scene parameter to a query parameter, you set the value of the query parameter to be equal to that of the scene parameter. For more information on modifying object properties and for information about functional expressions, see Chapter 7, "Object properties and events," on page 91.

## Using query groups

You can use query groups to ensure that related queries execute against the same database connection. Query groups allow you to use the results of one query in another query. For example, the first query creates a temporary table that the second query uses. If you do not use query groups, connections are pooled and might not be executed against the same connection.

Query groups apply to both worlds in QMF Visionary Studio and published worlds. A query can belong to only one query group. Query groups do not determine the execution order of queries; queries are executed when the results of the query are needed to display an object. For more information, see "Specifying drawing order" on page 50.

To create a query group, click **Edit —> Query Groups**.

## Modifying queries

The Query editor allows you to view and modify your queries. After you create a query in one of the wizards or in the Query Diagram view, a query diagram and SQL text are saved and you can later open the query to edit it. You can view and modify SQL text for any query in a QMF Visionary world. However, after you modify a query in SQL text, you might not be able to open a query diagram for it. All queries are saved in the Queries folder of the world.

**Note:** You cannot view or modify QMF queries in the Query Diagram view.

To view or modify a query, in the Worlds page of the World Manager, right-click the query and click **Open Query**. Click **Data Sheet** to run the query and check your results.

For instructions on using the Query Diagram or Text views, see the *DB2 QMF Visionary Getting Started Guide*.

### Setting query properties

You can right-click the background of the Query Diagram view and click **Properties** to:

- **Prevent duplicate records**. Some queries, such as joins, can return more instances of a record than you want to display in your layout. To make sure each record represents a unique case—for example, one unique sale to a customer—you can select the Unique Results check box.
- **Count records**. When creating a layout, you should know how many records (data points) will be displayed. To count the number of records returned from a query, select the Count records check box. The COUNT() aggregate function enables you to count the number of records for a particular field selected in the query.
- **Set maximum rows returned**. For each query, you can set a limit on the number of rows returned. Keep in mind that if you arbitrarily limit the rows returned, your query might not reflect the data accurately.

**Note:** You can set a row limit at the world level in the World Settings dialog box. The query row limit can make the result set smaller than the limit set for the world, but not larger. Click **Tools —> World Settings** and complete the Data Control page.

### Changing query associations

Each query is associated with the workbook used to create it and with the connection used by QMF Visionary Studio to connect to the data source. You can change a query's associations—either to its workbook or to its connection—with its context menu.

For example, you might develop a world using a test database, and then when the world is ready to be published, change the query associations to the data source for the production database.

# Chapter 4. Creating scenes

This chapter describes how to design, modify, and test scenes in a QMF Visionary world. This chapter introduces how to display data in a scene and provides information on how to use and refine layouts.

In this chapter:

- Displaying data
- Making scenes dynamic with parameters
- Refining scenes
- Using 2-D layouts
- Using list and combo controls
- Using 3-D layouts

You create a scene by inserting one into a world and adding layouts to display data, graphical objects, navigation, and events—any of the objects and object behavior that you want to present your data.

The principle tool for creating a scene is the Scene editor. You add objects to the scene displayed in the editor, then edit their properties with the Object Inspector and the Formula bar. For information on using the Scene Editor and other QMF Visionary Studio tools, see the *DB2 QMF Visionary Getting Started Guide*.

## Displaying data

Creating a scene is as much about designing data display as it is about arranging and modifying objects in a scene. To display data in a scene, you use a layout or certain controls. For descriptions of available layouts, see "Displaying data" on page 10.

Insert a layout by clicking **Insert —> Layout**. For most layouts, the Layout wizard helps you map data points to columns returned by your query. For more information see the QMF Visionary online help.

## Making scenes dynamic with parameters

A *dynamic scene* is one in which some aspect of the display changes based on a parameter value. You can use scene parameters or global parameters to make a scene dynamic.

Here are some examples of when would you want a scene display to change:

- **User interaction**. You might want to provide your user with specific options: for example, choosing the product for which they want to monitor sales performance.
- **Navigation events**. Users jump to the scene from a particular context in another scene and you want to carry that context into the new scene. For example, when the jump takes the user from a bar in a bar chart to a scene containing information specific to the bar data.
- **Viewer classes**. You want to provide your user with overall viewing options so that every scene in your world is based on viewer needs. A viewer class is a particular type of global parameter that can be used to make a scene dynamic.

After you define a scene or global parameter, you can use it to specify object properties or object behavior (event actions). See "Modifying object properties" on page 93 and "Creating event actions" on page 105 for more information.

### Using parameters

QMF Visionary enables you to capture user information (or other information, such as a query result) and pass it to other parts of your world with *parameters*.

Parameters enable you to do the following tasks:

- Capture user input.
- Capture current context, such as query results, object property values, or locations.
- Pass user input or context information from one part of a world to another.
- Vary query results based on dynamic criteria.
- Vary navigation results based on dynamic criteria.
- Vary visual display or other design elements based on dynamic criteria.

You can use parameters in many ways. Event actions typically capture user input in a parameter. You can use global and scene parameters when you design interactive controls in your scenes; when a user makes a selection in the control, an underlying event action sets the parameter value. The parameter value determines what data is displayed or how it is displayed.

The figure below shows the process by which information about a user event can be passed from the control to other actions in your world.

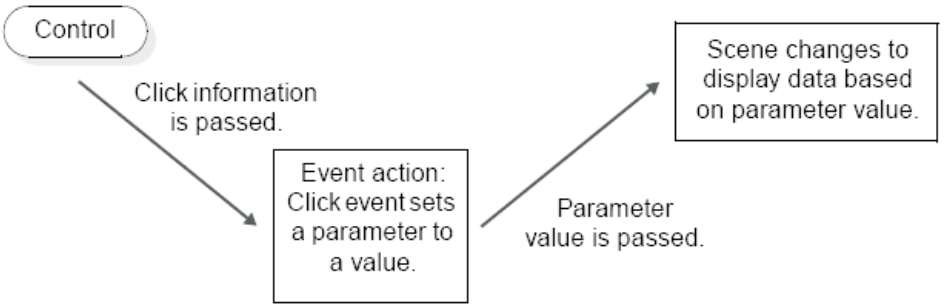



*Figure 9. Passing user input from controls to event actions*

In many scenarios, you might use a control to obtain information, but in general you must capture the user input either as a constant or into a parameter.

### Types of parameters

Parameters can be of the following types:

- **Global**. Available to all scenes in a world. Users can use global parameters (such as viewer classes) to change the display of a world. Use global parameters if you think you might use the parameter in several scenes.
- **Scene**. Available only for the scene for which it is defined. Passed from one scene to another by a wormhole or event action. Use scene parameters if you want only the display of one scene to change.
- **Query**. A variable used in a query that is set at runtime before the query is executed. Use to change query criteria. For more information, see "Using query parameters" on page 37.

### Defining parameters

Before you define a parameter, decide on a default value for it. You need to know your data to do this; you can check your query results in the Data Sheet view to see what records are returned and then select a value. For example, if you want to define a parameter that specifies a particular store ID, you can select all the store IDs from a table containing store information to determine a default value from among the possible query results.

To define a parameter, complete the Insert Parameter dialog box by providing the following information:

- Parameter name

- Use a naming strategy that helps you identify which type of parameter it is: for example, G_*name* for globals, S_*name* for scene parameters, and Q_*name* for query parameters.
- Data type
- Default value

The parameter name appears in the relevant parameters folder: Globals, Locals, or Queries.

### Using global parameters

A global parameter is available to all scenes in a world and to the end user. You can set global parameters when you specify object properties or when you specify event actions. For example, after you publish your world and you deploy it in a Visual Basic application, the defined global parameters determine the initial level of detail and the viewer class.

Global parameters can be public or private. A public parameter can be modified by the end user in the Runtime Settings dialog box or via the container application.

QMF Visionary provides four built-in global parameters: ViewerX, ViewerY, ViewerZoom, and ViewerClass.

To define a global parameter, click **Insert —> Global Parameter**.

For more information on modifying object properties and for information about functional expressions, see "Property expression examples" on page 102.

Use the ViewerClass global parameter to alter the appearance or behavior of a world, based on a viewer's identity. You can restrict access to certain scenes or data to those that belong to a specified viewer class. Viewer classes are not secure; users can change their viewer class in published worlds. You implement viewer classes with the IsViewer() function when you specify event actions or object properties, such as visibility, in a scene.

### Using scene parameters

A scene parameter is available in the scene in which it is defined. When you create wormholes and event actions, you pass scene parameters. You can also set the value of a scene parameter with an event action.

For example, when you create a wormhole, you connect two scenes and pass information from one scene to another. To set a wormhole parameter, first you insert a parameter in the source scene of the wormhole, and then you enter

the value of the parameter for the wormhole destination scene in the Wormhole.SceneParameter child object in the Object Inspector. The scene parameter name must be unique.

Scene parameters appear in the Locals folder under the scene in the World Manager.

## Refining scenes

In addition to layouts, a scene can contain a variety of graphic refinements, such as titles, buttons, color, and images. You create these graphics by adding objects to your scene and then modifying their properties.

The Palette Manager contains a selection of objects you might add to a scene. Explore your palettes to see the range of objects available.

The following sections describe some typical graphic refinements you might add to a scene.

### Adding titles

Like pages in a Web site, the scenes of your QMF Visionary world are easier to navigate and interpret if you add titles to them.

To create a title, add a Text object from the Primitives palette of the Palette Manager. For more information about the Palette Manager, see the *DB2 QMF Visionary Getting Started Guide*.

**Note:** Make sure you specify the value of the Text object with an equals sign (=) to signify a property expression rather than a constant. Put quotation marks
(″ ″) around the literal text and plus signs (+) between literal and variable portions of the expression.

If your world has several scenes that each need the same title or title bar, you can use a wormhole to a title scene. For more information about wormholes, see “Creating wormholes” on page 65.

### Adding instructions

When a scene requires action by your user, you need to provide instructions. You might create text, similar to the scene title created in the previous section, or you might create a label on a button. Creating instructions is part of designing the user interface for your published QMF Visionary world.

Suppose your scene includes a RadioGroup control that offers your users three different ways of viewing a layout. The options might seem self-explanatory to you, but users might need more explicit instructions.

The figure below shows a Text object containing instructions for how to use the Radio Group control.

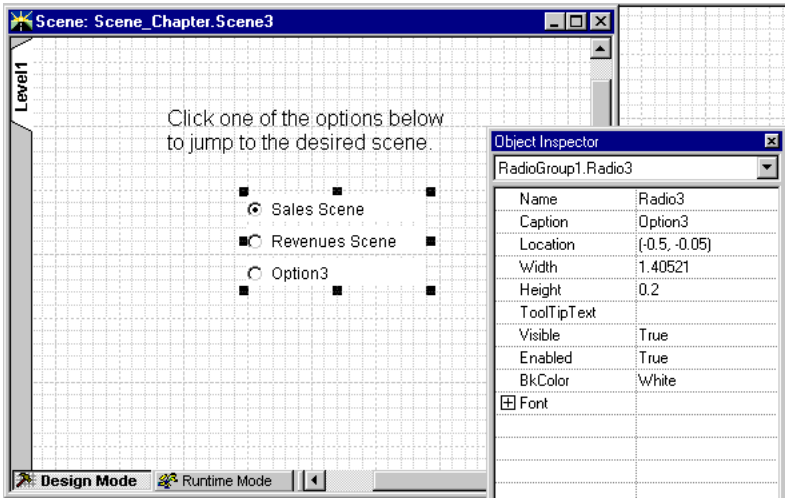



*Figure 10. Editing instructions for a Radio Group*

To make each option a jump, you must create an event action for each. See "Creating event actions" on page 105 for details.

## Providing online help

Depending on the complexity of your world and its audience, you might need to provide online help:

- Create a separate help scene that users access by clicking a link or button.
- Use mouse-over events to changes the visibility of help text. You can designate an area where help text appears when the user places the mouse over an object.

  For example, if you have a chart, you create a mouse-over event on the chart. Then add a text object containing your help text to the designated area. You can add many text objects on top of each other, each containing a different help topic and with visibility set on only when the user places the mouse over the corresponding chart.

## Adding tooltips

To specify tooltip text (displayed when a user keeps the cursor over an object), use the object's TooltipText property, as shown below.

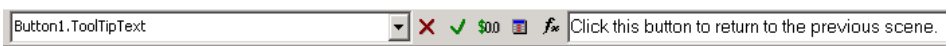


*Figure 11. Tooltip for a button*

All objects in QMF Visionary Studio provide a TooltipText property.

## Providing a legend

When you use symbols or colors to represent different types of data in a layout, it is helpful to provide a legend. You can insert data symbols into the scene that reflect the symbols used or to illustrate the colors in the layout and provide explanatory text next to them, as shown below.

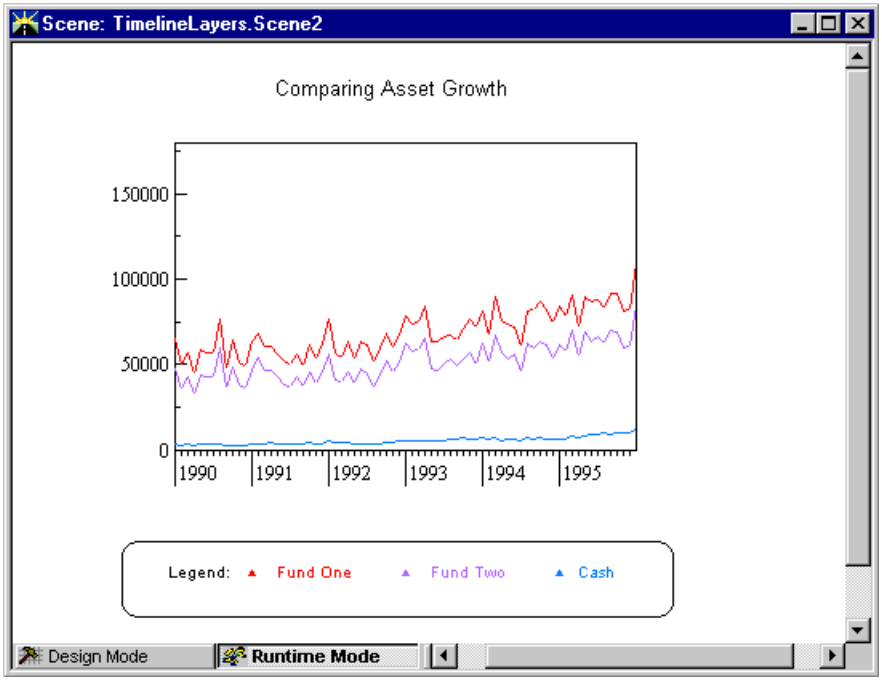



*Figure 12. Legend for a layout*

## Adding buttons

You can create buttons in your scenes by inserting the Button control and specifying the caption for the control and an event action.

For example, you might create a button to launch another application from QMF Visionary, such as Microsoft PowerPoint.

To create a button to launch another application, you insert a Button control, add a click event (use the Events page of the Object Inspector), specifying that the event launch the application you choose.

## Creating the overview scene last

If you create an overview scene to act as an entry point to your world, consider creating it after all other scenes are completed.

Some overview scenes contain wormholes that act as links to the other scenes in the world. In this case, the overview scene looks like a set of windows into the rest of your scenes.

**Note:** When you publish your world, by default the Publish wizard selects the first scene you created for your world as the first scene to display in the published world. Make sure when you publish that you select the first scene you want to display.

## Setting print defaults

Many of your users might want to print a scene. You can set printing defaults to print the scene accurately, including the default page size, orientation, and margins. Use the Page Setup dialog box (**File —> Page Setup**) to specify how a scene is printed. These settings are preserved when you publish the world.

By default, tables that span multiple pages are split so that rows remain whole; they also have complete borders on each page. Other objects that span multiple pages are moved to the next page, unless they are bigger than the page size. To allow tables or objects to be printed normally, clear the **Smart table breaks** or **Smart object breaks** check boxes on the Page Setup dialog box.

To see how your scene fits on a printed page, turn on the print guides (**View —> Print Guides**). They appear as dashed blue lines. The print area is offset from the upper leftmost object:

- If you move an object beyond the top or left of the print area, the print guides are adjusted.
- If you move an object beyond the bottom or right of the print area, a new page is added.

You can also see the print preview by clicking **File —> Print Preview**.

For more information on creating a printer-friendly version of the scene, see “Creating a printer-friendly scene” on page 24.

## Specifying drawing order

Objects lower down in the tree (within the same level) are drawn after objects higher in the tree. Sometimes you want objects to have a display based on the results of another query or temporary table, then you want the object to be drawn later.

A drawing order is necessary for the following:

- Objects that have properties set to another object's property (object.property) must be after the object.
- If a query is dependent on results of another query the object with dependent query must be after the other object.

To change drawing order, drag the object up or down in the world hierarchy in the World Manager.

Drawing order is not applicable to parameters.

## Using 2-D layouts

The 2-D layouts on the Layouts palette act like templates: they provide a basic format that you can enhance or modify to suit the needs of your SQL data display. QMF Visionary provides graphical objects and layout properties that you can use to customize your layouts, including the following items:

- **Layers**. Data templates that display data from different queries on a single layout. See "Using layers in a layout" on page 53.
- **Data symbols.** Icons that can be used to represent each data point in a layout. QMF Visionary provides default data symbols for most layouts, but you can substitute other data symbols for these defaults. You can also modify properties of a data symbol. See "Modifying data symbols" on page 55.
- **Connectors**. Lines that can be used to represent connections between data points. See "Using connectors" on page 56.
- **Axes**. Numerical or logarithmic scale lines used to chart data on graph layouts. See "Modifying axes in chart layouts" on page 57.
- **Scaling**. Property used in hierarchy layouts to determine the relative sizes of parent and child data points. See "Modifying scaling in hierarchy layouts" on page 59.

To group objects in a data template, you might need to use an alignment panel.

**Note:** When you click **Insert—>Layout**, the Layout wizard has additional choices besides those layouts on the Layouts palette, specifically, selected controls from the Controls palette and the ActiveX Controls palette. While these controls can display query data, they do not have data templates; consequently, you cannot modify them in the ways described in this section.

### Using the Data Template editor

When you insert a layout from the Layouts palette into a scene, QMF Visionary automatically inserts a *data template* for that layout. The data template provides the format for each data point shown in a layout.

Add graphics, text, data symbols, connectors, and events to data points in a layout using the Data Template editor pane of the Scene editor. The Data Template editor is the primary tool for modifying the appearance of each data point in the layout.

The Data Template selector (to the left of the Data Template editor) allows you to select a particular data template to display it in the Data Template editor. You can also to select individual objects within a data template to display their properties in the Object Inspector.

You can experiment with designs for your data templates. You might want to alter the color, visibility, or width of the objects in the data template. You might also want to add Primitives (such as Text or a Picture) and map column data to the primitives that reveals what each data point represents. Some data templates use a particular type of data symbol as default. You might want to replace it with another object.

The following figure shows the Data Template editor and selector at the bottom of the Scene editor after a pie chart was inserted into a scene. The Scene editor contains a placeholder object that represents the pie chart. The Data Template editor contains objects that are part of the data template for the pie chart. The Data Template selector displays the object hierarchy for the pie chart.

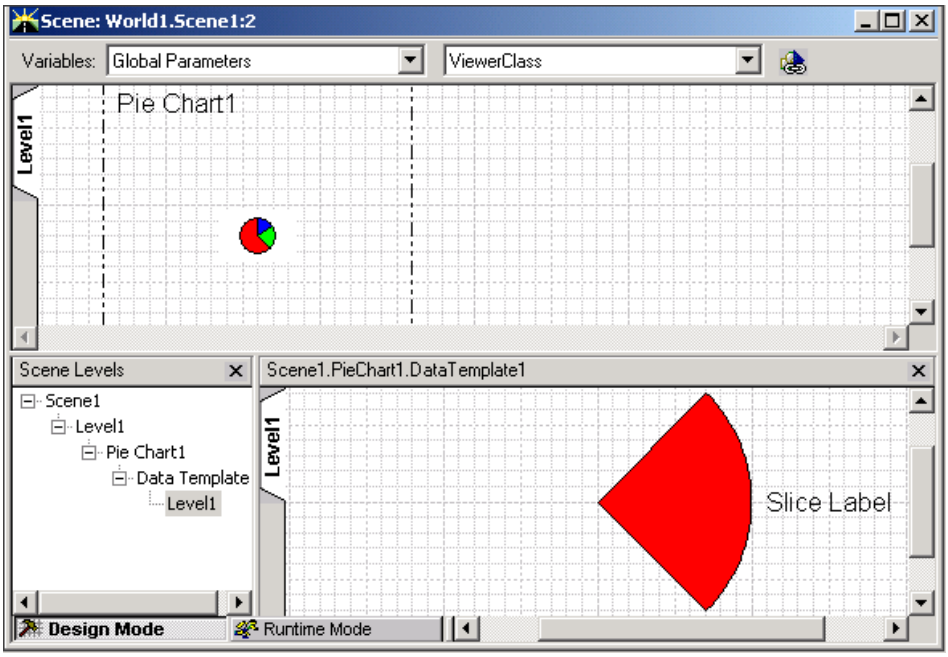


*Figure 13. Scene editor with the Data Template editor and selector*

## Using layers in a layout

*Layers* enable you to show data from several queries in one layout; each layer corresponds to more than one data template in the layout. The figure below illustrates the concept of layers.

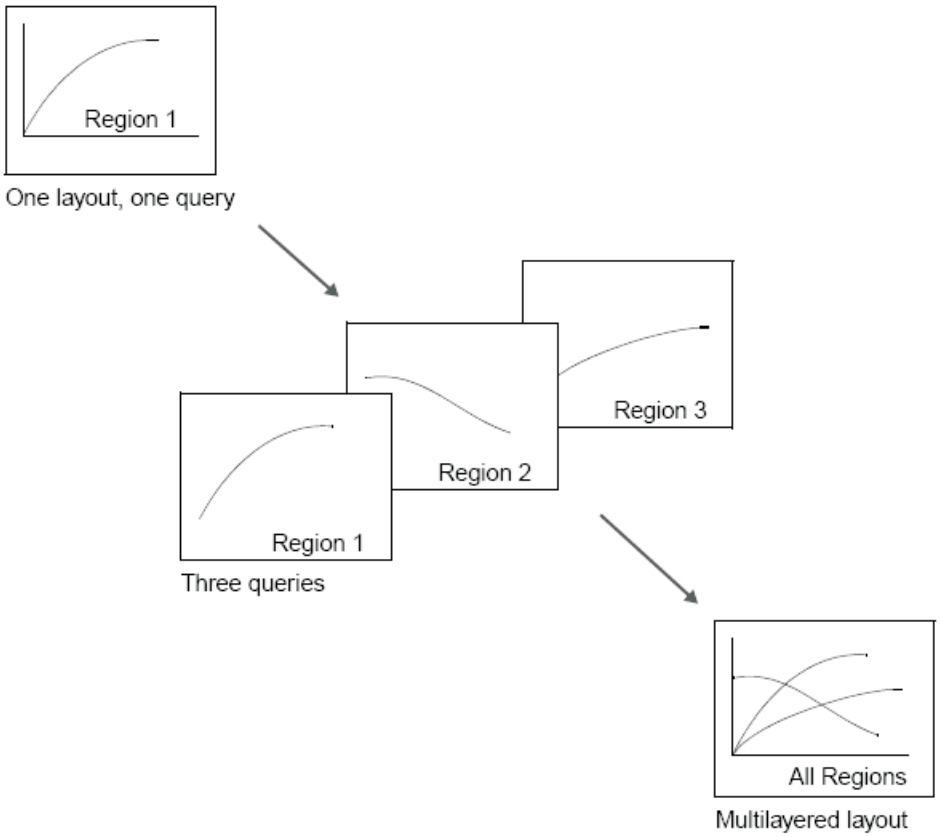



*Figure 14. Three layers in a layout*

**Note:** The term *layer* does not appear in QMF Visionary Studio, except in the documentation. The term is used to clarify what it means to have more than one data template in a layout and how to create this effect.

When you create a layout, the wizard prompts you to select a query and then helps you map the query to the layout. That initial query supplies the data for the initial data template.

You can add layers to the following layouts:

- Bar chart
- Column chart
- Candlestick chart
- Event band chart
- Linear map
- Scatter chart
- Stock chart
- Timeline chart
- XY chart

To create a layer on a layout, you add another data template—based on a different query—to the layout. There are several ways to create another data template. You can:

- Add a layer that looks similar but contains different data than the first layer:
  - Copy the initial data template and then change its query.
  - Copy the initial data template and then use a query parameter to differentiate the query results for each layer.
- Add a layer that looks different than the first layer. Insert a new data template into the layout. If you are creating a multiple-object data symbol, you might need an alignment panel. For more information, see "Using an alignment panel" on page 55.

To duplicate a data template and change the query:

1. Right-click the data template in the World Manager and click **Copy**.
2. Right-click the layout in the World Manager and click **Paste**.
3. Right-click the new data template in the World Manager and click **Change Query**.
4. Select a different query and click **OK**.

To duplicate a data template and use the same query with a query parameter:

1. Right-click the data template in the World Manager and click **Copy**.
2. Right-click the data template in the World Manager and click **Paste**.
3. Select the DataTemplate1.QueryParameters object from the Object Inspector list and set the query parameter to a value.
4. Select the DataTemplate2.QueryParameters object from the Object Inspector list and set the query parameter to a different value.

To add a layer with a different data template:

1. In the World Manager, right-click layout and click **Insert Data Template**.
2. Select the query and click **OK**.
3. In the Data Template Editor, insert an alignment panel, if necessary, a data symbol, and any other objects you need.
4. Specify the location of the alignment panel, if you are using one. See "Using an alignment panel" on page 55 for more information.

The second layer is added to the layout. You can modify the objects in each data template to distinguish them. For example, in a chart that uses connectors in the data template, you might make the Connector object for each layer a different color.

## Modifying data symbols

Typically, the default data symbol in a layout is the most appropriate one to use and can be customized by modifying its properties in the Object Inspector. A bar chart has horizontal value bars; a candlestick chart has candlesticks.

However, you might want to create a second data layer on a particular chart. For example, you might want to display particular event instances on a timeline chart that tracks a continuous trend in the market. By adding an event band data symbol in a data layer, you can add depth and significance to a timeline chart.

You might want to use multiple objects for each data point. For example, you might want to display a company name above each point in a scatter chart. To do so, you can add a text object above the point object in the data template. The data symbol is then a point and a text label.

### Using an alignment panel

If you create a multiple-object data symbol for a chart with axes that are measured data units instead of inches (or metric units), you should use an alignment panel to prevent the data symbol objects from overlapping.

Typically, when a layout requires an alignment panel to group objects at each data point, the data template includes the alignment panel for your convenience. For example, the X-Y chart has axes mapped to column values and includes an alignment panel so you can easily add other objects to represent data points.

When you use an alignment panel to group data symbol objects, a panel is centered at each data point. To determine to what you should set the Location property for an alignment panel to, look at the Location property of the default data symbol for that chart. For charts with axes that supply an alignment panel by default, the Location property for alignment panels is set to the Pt() function, which references one or more column names.

To insert an alignment panel:

1. In the World Manager, double-click the data template you want to modify to display it in the Data Template editor.
2. Click **Insert —> Alignment Panel**.
3. Specify a value for the Location property of the alignment panel.
4. Insert objects into the alignment panel:
   - In the World Manager, move existing objects under the alignment panel so that they become child objects of it.
   - With the alignment panel active in the Data Template Editor, insert new objects into it from the Palette Manager. The objects appear in the World Manager as children of the alignment panel.
5. If necessary, arrange the child objects on the alignment panel by dragging them or by modifying the values of their Location properties.

The following figure illustrates an alignment panel with child objects in the World Manager.

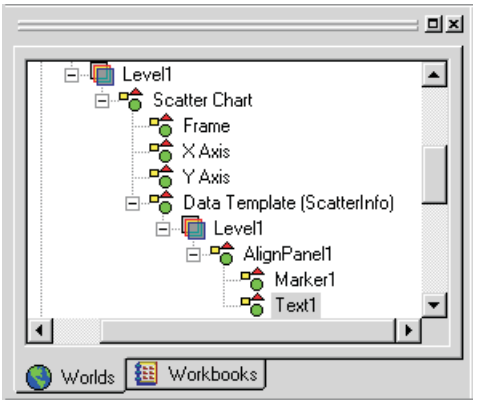



*Figure 15. An alignment panel displayed in the World Manager*

## Using connectors

*Connectors* can provide visual enhancement to both scenes and layouts. Typically, connectors are included when you create a hierarchy layout or a chart, such as an XY chart.

The Connectors palette of the Palette Manager provides several line connectors and a single Connection Point object.

The three line connectors, Straight Connector, Elbow Connector, and Spline Connector, provide distinct graphical differences in your layout or scene designs but do not affect the underlying mathematical or logical relationships between data points in the layout.

The Connection Point object specifies where each data point in a hierarchy connects to its parent element. To override the default connection point for a layout, you can insert a Connection Point object into a data template or change its location. For example, you can change the appearance of an organization chart by replacing the elbow connector with a spline connector.

**Note:** Hierarchy layouts and chart layouts with line connectors contain implicit connection points. No Connection Point object appears in the World Manager when these layouts are created; however, the Connector object has a Connection Point property defined for it. You can change the location of the connection point by altering the coordinates of the Connection Point property of the Connector object.

## Modifying axes in chart layouts

*Axes* provide the graphing capabilities for chart layouts. You can create standard charts with a single x-axis and a single y-axis or, as in stock charts, you can create more sophisticated graphs with multiple axes in one dimension.

Most chart layouts create the required axes by default. For example, when you create a scatter chart using the Layout wizard, the layout contains two axes: ScatterChart.XAxis and ScatterChart.YAxis, as shown below.

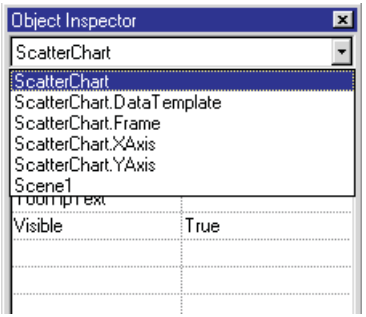



*Figure 16. Axes of a Scatter chart*

### Modifying the axis range

You can modify an axis range to change the way your layout appears.

Typically, you use the default AutoScaleUpper and AutoScaleLower properties to set your upper and lower axes limits. However, if you want to refine or limit the data shown in the chart, you can manually set the axis limits.

**Note:** In general, your data display will be most accurate and usable for other types of displays if you specify your ranges in the query supplying the layout, rather than in the chart's upper and lower limits.

### Modifying the axis interval labels

You can also modify the label intervals and minor ticks on the axis.

To manually specify the label and tick intervals, you must first specify a real value for the label interval based on the data returned by your query.

For example, if you have numerical data between 10,000 and 150,000, you must specify label intervals that correspond with this range of data. Thus, a label interval of 10,000 would be logical. After you specify an appropriate label interval, you can then change your IntervalSettings property to Fixed.

### Adding an axis

Adding an axis to a chart is similar to adding a layer to a layout; it enhances the analysis of your data. You can add axes to any chart that uses axes to graph data points: for example, timeline charts, stock charts, and scatter charts.

**Note:** Column charts and bar charts allow you to graph multiple values along the y-axis without adding any additional y-axes. They generate multiple bars to represent the multiple values.

Typically, when you have two axes representing two data ranges, you can display a single correlation. For example, you might want to correlate sales volume with time of year, location, advertising spending, or some customer demographic.

By adding another numerical axis to a graph that contains a single date/time axis and a single numerical axis, you can display a dual correlation against time. For example, you might want to create a scatter chart to show how total revenues per store compare to average revenues over a period of time. Total and average revenues can be charted on individual y-axes and time can be charted on a single shared x-axis.

You can insert an axis by right-clicking the layout in the Worlds page and selecting Insert Axis. After you insert an axis, you must select it in the Data Template editor as shown in the figure below. Then, you must map objects in

the data template to the columns of your query.

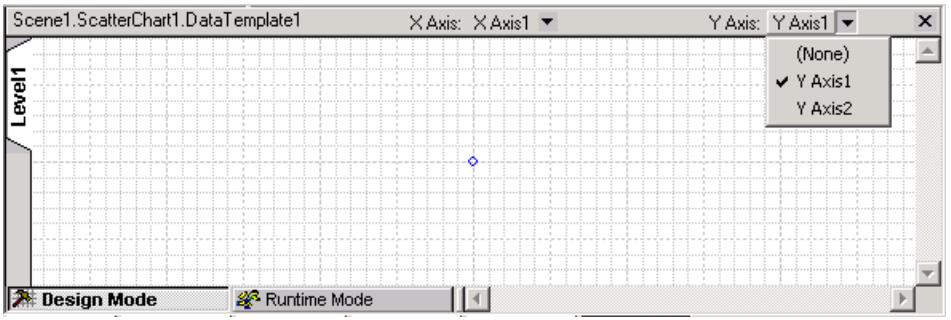



*Figure 17. Selecting another axis in the Data Template editor*

**Note:** In the Object Inspector, examine the data mapping specified for the initial x- and y-axis pair and match it in the new x- and y-axis pair.

## Modifying scaling in hierarchy layouts

*Scaling* allows you to specify two aspects of hierarchy layouts: the relative size of child and parent elements, and the number of child levels viewable at each zoom increment.

The following figure shows the Object Inspector displaying the default properties of the Organization chart layout.

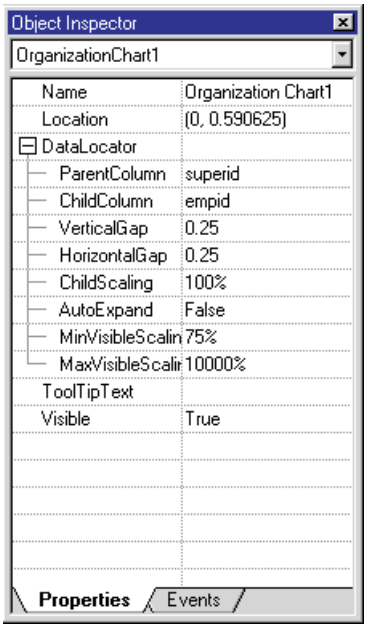


*Figure 18. Organization chart properties*

The ChildScaling property specifies the relative size of each child cell to its parent. In the default organization chart, the child cells are scaled at 75% of the size as the parent cells. For an enhanced effect, or to save screen space, you might change the ChildScaling to 50% so that child cells appear much smaller than parent cells.

MinVisibleScaling and MaxVisibleScaling specify the zoom levels at which each child level of the hierarchy is viewable when the scene is run.

If ChildScaling is 100%, all levels of the hierarchy are viewable at a MinVisibleScaling up to 100%, because all levels are the same size and the default zoom level for the scene is 100%.

However, perhaps you want your layout to reveal each child level as the viewer zooms in. You might want the second level of the hierarchy to become viewable only when the end-user zooms in to 200%.

One way to create this effect is to set the ChildScaling to 50% so that each child level becomes viewable when its the zoom level doubles the size of the child element. You must set the MinVisibleScaling to a value greater than the ChildScaling factor of 50% and less than or equal to the default zoom level of 100%.

The default MinVisibleScaling of 75% allows the first and second levels to be viewable between a zoom level of 75% and 100%. Then, the third level becomes viewable when the zoom level is greater than 100%.

By experimenting with these scaling factors, you can achieve a desired affect for viewers zooming in on your data.

## Using list and combo controls

The List and Combo controls, on the Controls palette, allow you to display data returned by a query as options in a list box.

You can display one set of values in the list box and specify another set of undisplayed values associated with the display values. Thus, you can provide text values, such as country names, in the control but pass less intuitive options, such as geospatial boundaries, to other QMF Visionary objects, such as a linear map.

List and Combo controls have three properties that determine the displayed list options, the value of the control to be passed, and the default value of the control:

- **OptionList**. The options shown in the list box.
- **ValueList**. An optional set of undisplayed values that set the Value property.
- **Value**. The value of the control that can be passed to parameters, events, and other QMF Visionary objects. Determines the default value of the list box.

When your user selects a value in the list, the selection sets the List.Value property for the control. The Value property is equal to the corresponding value of ValueList, if it is defined, otherwise OptionList. You can design an event action that occurs based on the option selected by the user. See "Creating event actions" on page 105 for information on creating event actions.

You can set the Value property to a parameter or function to define the default value of the list box.

If you want to base which option is selected on the action of the user from another object, use a global parameter. Set the Value property for the control equal to the parameter.

Set the ValueList property when you want the value sent to a parameter or event to be different than the value selected to the user (set by the OptionList property).

When a user selects an option from the list, the row from ValueList that corresponds to the same row number from the chosen option is passed to the event or parameter.

You can define the ValueList property in two ways:

- **Use the Layout wizard**. Select another column on step 3 of the Layout wizard. This sets ValueList to the Field() function. The Field() function references a column, which is different from the one used by the OptionList property, and a query, which is the same as the one used by the OptionList property.
- **Edit it manually**. You can type a value in the ValueList field. You can use the Field() function to reference a column and query, neither of which need to be the same as those used by the OptionList property. You can also set the ValueList property to a parameter, a function, or a static list.

To specify a static list, click the button at the end of the OptionList or ValueList value cells.

To specify a column returned by a query, use the Field() function for the OptionList and ValueList properties, and the FieldValue() function for the Value property:

- The Field() function takes two arguments: the name of a column and the name of a query.
- The FieldValue() function takes three arguments: the name of a column, the name of a query, and the row number (in the database) of the default value selected in the list.

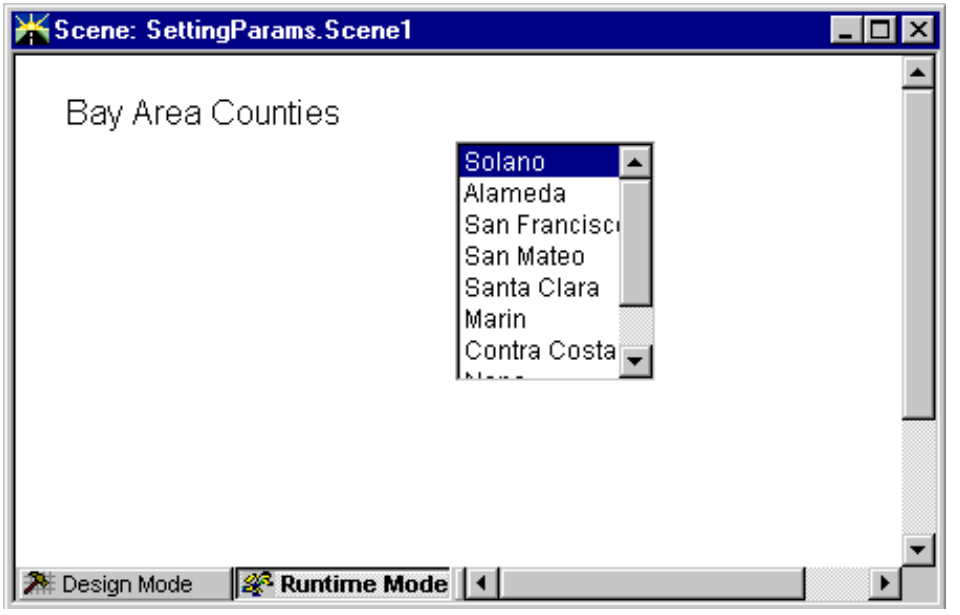



*Figure 19. List box example*

## Using 3-D layouts

The 3-D layouts (charts) on the ActiveX Controls palette act like templates: they provide a basic format that you can enhance for modify to suit the needs of your SQL data display. QMF Visionary provides graphical objects and layout properties that you can use to customize your layouts. For information on changing the layout properties or events associated with these 3-D charts, see “Modifying object properties” on page 93 and “Creating event actions” on page 105.

# Chapter 5. Creating navigation

This chapter describes the types of navigation you can provide to users of your world. QMF Visionary navigation allows you to design worlds not only with jumps between scenes and standard scrolling, panning, and zooming, but also with data-driven navigation such as drilling down to greater detail and changing data display based on user input.

In this chapter:

- Types of navigation features
- Standard navigation features
- Standard navigation features

These sections provide descriptions of features and a selection of examples for creating these features in your world. For more information about the specific dialog boxes and application commands you use to create these features, see the QMF Visionary Studio online help.

## Types of navigation features

QMF Visionary Studio provides both *standard* and *custom* navigation features.

*Standard* navigation features are provided for developers to navigate within editor windows in QMF Visionary Studio and for end users to navigate QMF Visionary worlds within QMF Visionary WorldView.

*Custom* navigation features include jumps, event actions, and other data- or user-driven features you can create for your QMF Visionary world.

Each category is described more fully in the following sections.

## Standard navigation features

QMF Visionary provides the following standard navigation features in both QMF Visionary Studio and for end users in QMF Visionary WorldView:

- **Panning**. Use the Grab Canvas tool to move the scene left, right, up, or down.
- **Scrolling**. Use the scroll bars on the editor windows to shift focus up or down, to the left or to the right.
- **Zooming**. Use the zoom tools to zoom in or zoom out.

For example, you can zoom in on, or magnify, a scene or a data template you are designing. You can also zoom in or zoom out from a runtime view of the world.

You can also create levels of detail in a scene or data template to add value to zooming actions. See "Creating levels of detail" on page 70 for details.

- **Jump Tos**. Right-click the background of any scene in runtime mode and select Jump To to jump instantly to another location.

## Custom navigation features

QMF Visionary Studio provides the following custom navigation features:

- **Jumps**. Allow end users to move immediately from one scene or viewpoint to another.
- **Wormholes**. Allow end users to see a portion of another scene before jumping to it. Wormholes look like windows into another scene.
- **Viewpoints**. Allow end users to examine a particular area of a scene at a particular zoom level.
- **Levels of Detail**. Allow end users to see more data as they zoom in.
- **Drilldowns**. Allow end users to click a data point and jump to a new scene containing more information.
- **Scene Tabs**. Allow end users to click tabs to jump from scene to scene.

Custom navigation features enable you to create dynamic worlds, in which users can move through many data displays and make choices about the kinds and amounts of data they want to view. They are described more fully in the following sections.

### Creating jumps

*Jumps* are hyperlinks between locations in a world. Most often, you provide a jump to a different scene, but you can also provide jumps to other viewpoints.

To create custom jumps, you assign an event, such as a user clicking a button, to the jump action.

Jumps can carry context information to the new location. For example, you can specify a scene parameter to be set at jump time and base the parameter on a calculated value or category, such as store location. Thus, a destination scene showing sales revenues can be based on store location and display different data depending on what context the user is coming from.

You must create both the source and the destination scenes or viewpoints before you can create a jump between them.

You can use the Object Behavior dialog box (double-click an event type on the Events page of the Object Inspector) to specify the jump event. For instructions, see the *DB2 QMF Visionary Getting Started Guide*.

You can specify an existing destination scene in a jump using an expression instead of a scene name. For example, you can specify the following expression:

```
=If(IsViewer("Plumber"), "Plumbing_Scene", "Electrical_Scene")
```

If the viewer class is Plumber, the jump goes to Plumbing_Scene. Otherwise, the jump goes to Electrical_Scene.

You can specify a parameter to be set when a jump action occurs. Global parameters appear in the Set Parameters page of the Object Behavior dialog box. Scene parameters appear in the Perform Action page after you select the jump destination.

**Note:** If you want the event action to set a parameter at the same time that it specifies the jump, you must create the parameter before creating the event action.

## Creating wormholes

*Wormholes* are portals through which you can view a portion of another scene and then jump to that scene. Wormholes can enhance end user's decision-making by providing a glimpse into the destination scene. You can create wormholes of various sizes, showing varying amounts of the destination scene or viewpoint. Alternatively, you can create a wormhole without an associated event action. This type of wormhole is useful for creating navigation bars or other repeating elements. For more information, see "Using wormholes for repeated elements" on page 24.

The following figure illustrates the concept of a wormhole.

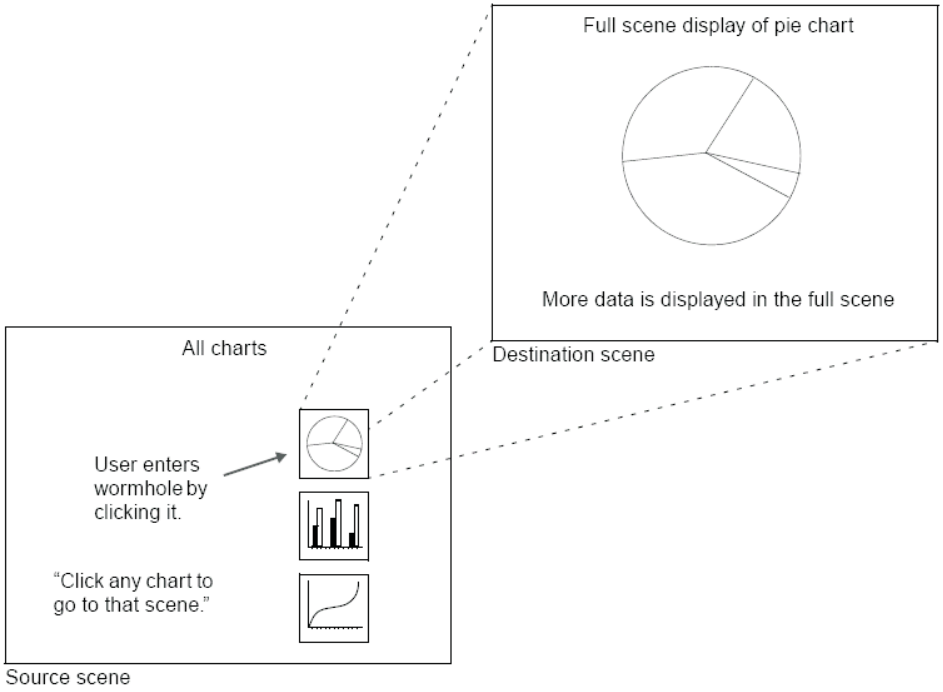



*Figure 20. An illustration of a wormhole*

You must create the source and destination scenes before you can create a wormhole between them.

When you create a wormhole, you can select how and whether the user enters the wormhole. Clear the **Enter wormhole when clicked with the mouse** check box in the Wormhole dialog box if:

- You want the user to select the Enter wormhole tool and then click the wormhole to go to the destination scene.
- You do not want the user to be able to jump to the destination scene (the wormhole only displays the destination scene).

To create a wormhole:

1. Open the source scene.
2. Double-click the Wormhole object on the Portals palette of the Palette Manager.
3. In the Wormhole dialog box, select a destination scene and whether the user can enter the wormhole by clicking the mouse.
4. Move the wormhole where you want it to appear and size it, as desired.

5. Modify the wormhole properties, such as the zoom properties and scroll bars, as desired. For instructions, see the following section.

### Wormhole properties

Wormholes have several properties that you might want to modify, including:

- **ZoomPct**. Sets the zoom percentage of the destination scene as viewed through the wormhole. You might want to reduce the zoom so that you see more of the destination scene.
- **SceneCenter**. Sets the position of the wormhole over the destination scene. By default, this property is set to the center of the scene. You can change this value to display another area of the scene. For example, you might have several charts on one scene, but want to display only one of them through the wormhole.
- **Border**. Sets the style of the border around the wormhole. You can select a border style, or select no border.
- **PassThroughEvents**. If set to TRUE, allows the end user to trigger events in the destination scene, but the wormhole itself cannot have an associated event. For example, you can use a wormhole to display a row of navigation buttons, which have associated events to take users to specific scenes. The user can click button to jump to a new scene, but cannot go through the wormhole to the navigation button scene.

### Using parameters with wormholes

When you create a wormhole, you might want to pass information captured in the source scene so that the display of the destination scene is dynamically changed. For example, you might want to set a global parameter based on a user action in the source scene—such as clicking an option button—and then pass that global parameter to the scene parameter.

See “Making scenes dynamic with parameters” on page 44 for information on creating a scene parameter.

When you create a wormhole, QMF Visionary Studio also creates a child object, Wormhole.Scene Parameters, which lists the scene parameters defined for the destination scene. To set a scene parameter when a user enters a wormhole, select Wormhole.SceneParameters in the Object Inspector list and set the appropriate parameter to the value you want.

**Note:** To verify the default value of a scene parameter, you can set the parameter property as its own name (for example, =S_parameter).

The QMF Visionary Tutorial has an example of setting a scene parameter with a wormhole.

## Creating viewpoints

A *viewpoint* is a location in a scene. A viewpoint has three properties: the x- and y-coordinates of the center of the viewpoint, and the zoom level.

Viewpoints enable you to isolate a particular area of a scene at a particular zoom level. You can use them as targets in jumps between scenes and associate them with a particular a level of detail.

For example, suppose a scene presents a world map indicating factory locations. You could define viewpoints for each region of the world that would enable users to move quickly to the location and magnification for the region they are interested in.

The following figures illustrate two viewpoints of the same scene.

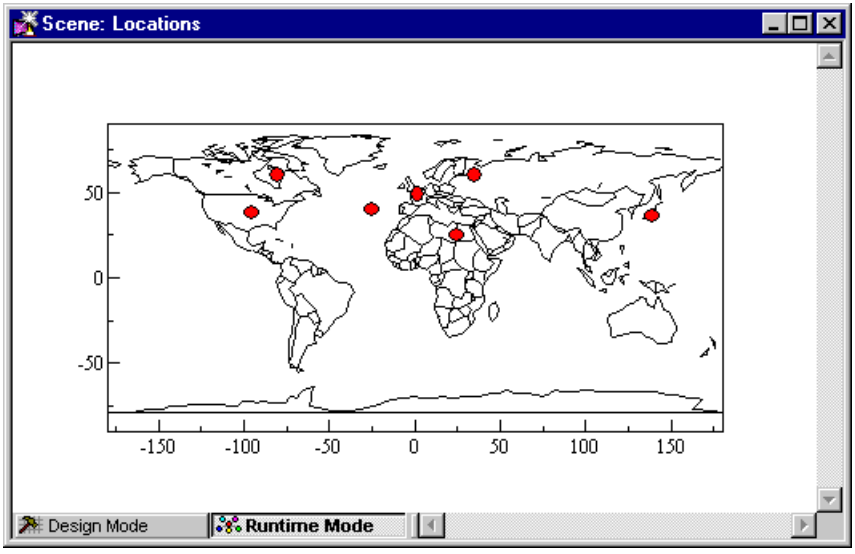


*Figure 21. A viewpoint named "Global"*

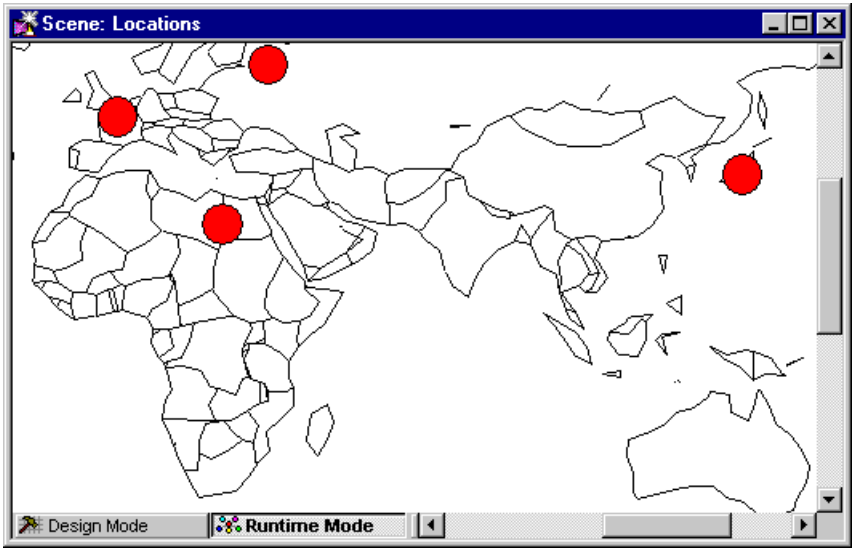



*Figure 22. A viewpoint named "Africa and Asia"*

To jump from one viewpoint to another, you can right-click the scene background, click **Jump To**, and select a named viewpoint.

In addition to providing a navigation tool, named viewpoints can be the target of an event action. For example, you can specify the display of a particular viewpoint if a user double-clicks an object.

**Note:** The default viewpoint (0,0, 100% zoom) is a helpful orientation point when you are developing a scene. In a large scene, when you want to return to the center, you can press the HOME key to take you back to the default viewpoint.

To create a viewpoint, with the scene open in the Scene editor, click **Insert —> Viewpoint** and then select values for the properties and type a viewpoint name.

**Note:** You can also save a viewpoint in runtime mode of the Scene editor. Click **Edit—> Save Viewpoint** to display the Insert Viewpoint dialog box.

You can now specify the named viewpoint when you create event actions that jump to a new location. Named viewpoints are a powerful navigation tool

when you combine them with *levels of detail*, which display more data as a viewer zooms in on a scene. See the following section for information about levels of detail.

## Creating levels of detail

A *level of detail* (identified as Level in the World Manager) is a view of data that is associated with a zoom level. Levels of detail enable you to present more information as you zoom in on a scene or data point. As you zoom in, fewer data points are displayed and more screen area is available for showing details.

Good design means just enough information just when it is needed. You can design layouts to display financial highlights, summary values, company logos, or other high-level information at first glance. You can then add levels of detail that offer more information as the user zooms in to reveal that information.

You can create levels of detail in:

- **A data template**. Allows you to provide more query information on each data point as the user zooms in on a 2-D layout. You must use the same query for every level of detail. For more information, see "Creating levels of detail in a data template" on page 71.
- **A scene**. Allows you to display an entirely different layout and query as the user zooms in on the scene. For more information, see "Creating levels of detail in a scene" on page 73.

After you create a level of detail, consider some of the properties you might want to modify, including:

- **Level of detail transitions**. Use the Level of Detail Transition editor to adjust the zoom level for the transition from one level to the next.
- **Visibility**. A property of most objects in QMF Visionary. Use the Object Inspector to specify the conditions in which the objects in the data template (or the data template itself) are visible. You can limit visibility by specifying a particular threshold or logical expression, such as an If() statement.
- **Scaling**. A set of properties in hierarchy layouts. Scaling properties determine how labels, data, and child cells of the layout appear when you zoom in. By adjusting scaling, you can control how large objects will appear when a user zooms in, and at what zoom level each node becomes visible or invisible.

See the reference section of the QMF Visionary Studio online help for details about object properties and functions available. The QMF Visionary Tutorial has an example of creating a level of detail.

### Creating levels of detail in a data template

You can create levels of detail in a data template for 2-D layouts only; you cannot create a level of detail for a 3-D layout.

By adding a level of detail to a data template, you can reveal additional or different data in a scene as the end user zooms in. The following figure illustrates the concept of levels of detail.

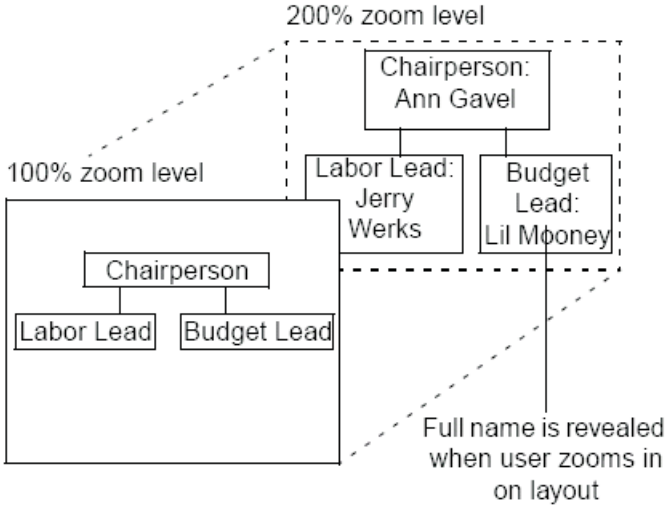



*Figure 23. Data template levels of detail*

For example, you might want an organization chart to show only titles when the scene is viewed at 100% (the default zoom level). However, when the end user zooms in to 200%, you might want to reveal names, phone numbers, and even photos of the employees.

Here is an example of two levels of detail in an organization chart. The following figure shows the first level.

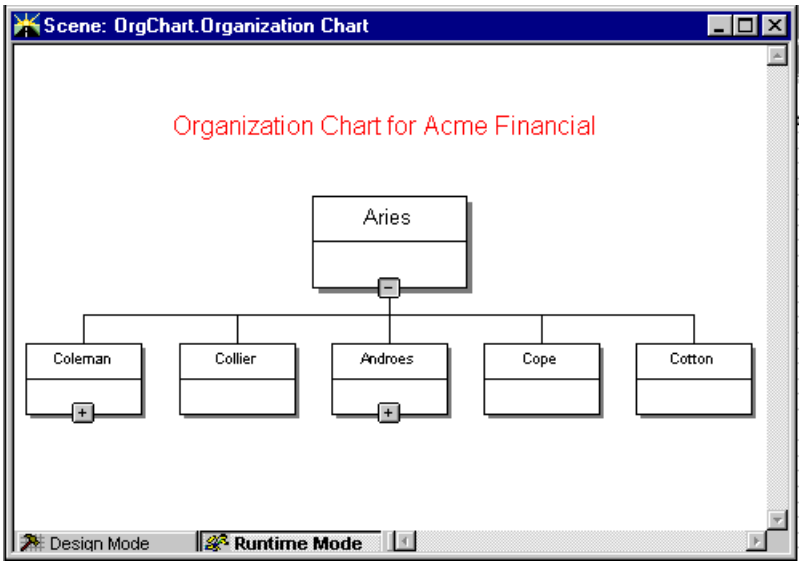



*Figure 24. Organization chart, level 1*

When the user clicks the zoom tool and then clicks the scene, QMF Visionary zooms in to 200% magnification and reveals the second level of detail.

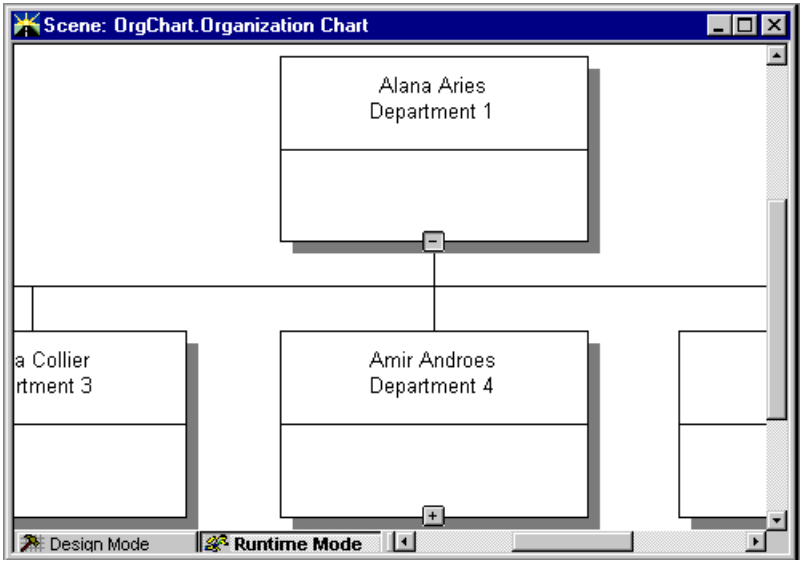


*Figure 25. Organization chart, level 2*

To create a second level of detail in a data template, right-click the data template in the Data Template editor and click **Insert level of detail**. You can then edit the objects to display more data. In this example, the Value property of the label has two additional columns in the second level of detail: firstname and department.

**Note:** When you compose queries for layouts, include as many columns as you think might be valuable to end users. You can display only the most important or highest-level column information in the default level of detail, and then add a level of detail to reveal more column information.

### Creating levels of detail in a scene

Levels of detail in a scene allow end users to see more or different information by zooming in on the scene. When you create a level of detail, you can specify the transition point (zoom level) at which a user shifts from one set of detail to another, and you specify what objects make up the level of detail.

You might want to create a scene level of detail if you cannot provide the detail you want using the same query that is mapped to the layout. In this case, you might create a second level of detail in the scene and include all the same objects as the first level of detail, but with a different query used for the layout.

In another situation, you might create a completely different layout in a second level of detail. For example, when the user zooms in to the finest detail on a particular data point, you might want to change the entire scene to display a single Web page that shows information about a company.

To create a level of detail for a scene, right-click the scene in the World Manager and click **Insert —> Level of Detail**. You can select to initiate the second level of detail with the objects from the first, or you can add a new layout to the second level of detail.

**Note:** Create viewpoints to direct user attention to a particular level of detail. Using a jump to a viewpoint, you can allow end users to go directly to the area in the scene and the level of the detail that contains desired information.

## Creating drill-down scenes

You might want users to click a data point to jump to another scene containing more information about that data point.

To create a drill-down scene, you need to create two scenes. The source scene should contain the following objects:

- An SQL query.
- A layout that supports a data template.
- Click event. Must have actions to jump to the destination scene set the scene parameter in the destination scene equal to a column displayed in the source scene.

The destination scene should contain the following objects:

- Layout.
- Scene parameter. Set to a column displayed in the source scene.
- SQL query. Must reference the column in the scene parameter and use a query parameter.
- Query parameter. Set to the scene parameter.

You can use the Drilldown wizard to automate creating the click event, the destination scene, and the scene parameter. The Drilldown wizard is available for all layouts that support data templates. For more information about the Drilldown wizard, see the *DB2 QMF Visionary Getting Started Guide*.

## Creating scene tabs

*Scene tabs* are navigation aids that you can provide in a published world. They allow end users to jump from scene to scene by clicking the tab associated with the scene they want to see. The following figure shows scene tabs in a published world.

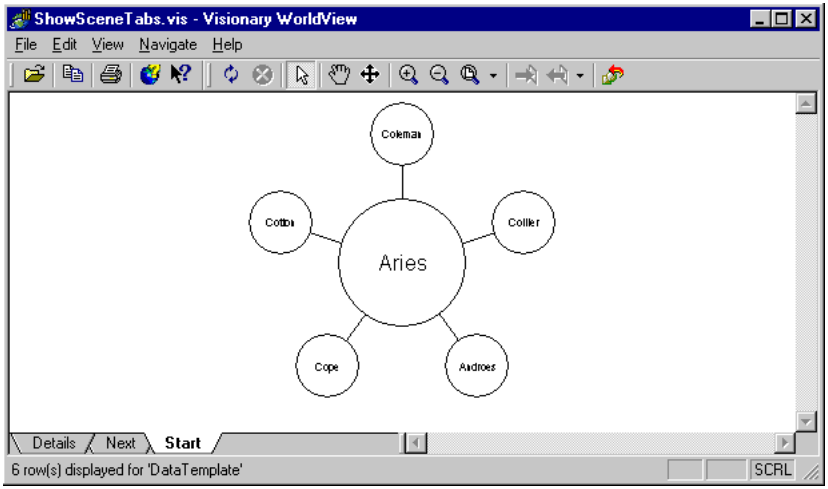



*Figure 26. Scene tabs in a published world*

To create scene tabs, the AllowAdHocJumps property for the scene must be set to True.

However, in some cases, you might want to restrict how end users navigate from one scene to another and thus prevent them from using scene tabs. For example, if you have a scene that displays a single record for a particular customer, you might want users to view that scene based on an earlier choice. Viewing the default customer record might not provide end users with helpful information. In this case, you might want to set the AllowAdHocJumps property of the scene to False.

# Chapter 6. Custom graphic features

This chapter describes a selection of the custom graphic features you customize for special use to enhance a QMF Visionary world. QMF Visionary Studio provides many other objects in the Palette Manager, including basic primitives and specialized objects, such as layouts, connectors, and data symbols. These other objects are described in other chapters.

In this chapter:

- Using controls
- Custom objects and the Custom palette
- Storing and using images in Image folders
- Creating custom color maps and sequences
- Creating and using viewer classes

Also, see the QMF Visionary Studio online help for details on the properties and events of all objects provided by QMF Visionary.

## Using controls

QMF Visionary supplies two types of controls:

- **Standard controls**. These edit controls allow the user to perform an action.
- **ActiveX controls**. These display controls are small software programs embedded in a QMF Visionary world to present information.

### Using standard controls

*Standard controls* (on the Controls palette) are active features; that is, you use them to encourage your end user to perform an action, and then you can specify what happens in the world as a result of that action.

Typically, you use these controls with event actions. For example, you might insert a Button control into a scene, with a caption that reads Back. You assign an event to the control and specify an action to take your user to the previous scene.

As another example, you might insert a RadioGroup control into a scene, with three options: StoreA, StoreB, and StoreC. You assign an event to each option and specify an action to have the user's selection to change which store's data is displayed in the scene.

The following table lists the controls provided in QMF Visionary Studio, along with brief descriptions of how to use them and some of the properties and custom events available with each control.

*Table 11. Standard controls and how to use them*

| Control/Uses | Selected properties | Custom events |
|---|---|---|
| **TextBox**<br>Allows user to input text. | Text can be justified, scrolled, and also displayed as asterisks (*) for password protection. | Actions can be triggered by user events such as selecting the object or changing the text in the box. |
| **Button**<br>Displays text or an image to prompt users to perform an action. For example, a **Back** button can assist user navigation. | A button can include a caption, an image, or both. An alternate graphic can be set if an image is disabled. | Actions can be triggered by user events, including selecting the button, as well as by standard mouse events, such as clicking. |
| **CheckBox**<br>Displays a box that users can select or clear to provide a Yes or No response. | The control can include a caption. | Actions can be triggered by standard mouse events, such as clicking. |
| **RadioGroup**<br>Provides users with multiple-choice options. Each option in the group is a child object with its own properties and events. | The control can include a caption. | Actions can be triggered by standard mouse events, such as clicking. You can create an event for the group or for each individual option. |
| **Slider (Horizontal or Vertical)**<br>Allows users to specify an integer within a range of values. | The minimum and maximum values in the range can be specified. Select properties, such as frequency, can be specified. | Actions can be triggered by a user beginning to change the value on the slider or ending the change; also by pressing or releasing a specified key. |
| **Combo**<br>Allows users to type an entry or select from a list. The list can be retrieved from a query or supplied as static text. | Option values and a sort order can be specified for options returned by a query. | Actions can be triggered by standard user events, such as clicking. Also, an action can be triggered when the value in the control is changed. |

*Table 11. Standard controls and how to use them (continued)*

| Control/Uses | Selected properties | Custom events |
|---|---|---|
| **List**<br>Allows users to select from a list. The list can be retrieved from a query or supplied as static text. | Option values and a sort order can be specified for options returned by a query. You can also specify if you want the user to be able to select multiple options. | Actions can be triggered by standard user events, such as clicking. Also, an action can be triggered when the value in the control is changed. |
| **Animated GIF**<br>Allows users to display animated GIF graphics. | The graphic can be a stored image or come from a file on a computer. | Actions can be triggered by standard mouse events, such as clicking. |

**CAUTION:**
**Captions for controls do not allow multiple lines of text. If you type the delimiter newline into a caption expression, the delimiter and any text following it are ignored.**

Most events are common to all objects in QMF Visionary. For full details on the properties and events associated with each control object, see the reference section of the QMF Visionary Studio online help.

## Using ActiveX controls

The following table lists the ActiveX controls provided with QMF Visionary Studio, along with brief descriptions on how to use them and some of the properties and custom events available with each control.

*Table 12. ActiveX controls and how to use them*

| Control/uses | Selected properties | Custom events |
|---|---|---|
| **WebBrowser**<br>Displays a Web page specified by a URL based on a query or a static value. Can display the contents for any OLE-compliant applications such as Microsoft Excel, Word, and PowerPoint. Can display HTML-based database entry forms and the contents of a directory. | URL address, tab, and caption can be specified. | Many custom events are available, including events related to downloading and navigation. |

*Table 12. ActiveX controls and how to use them (continued)*

| Control/uses | Selected properties | Custom events |
|---|---|---|
| **RichTextBox**<br>Displays text in Windows RTF or ASCII format. Text displayed can be retrieved from a column containing a TEXT data type or from a file, or it can be specified in the property cell. | A tabbed border can be created and can contain a caption. | Actions can be triggered when a user changes the text, selects text, or presses a key. |
| **ActiveMovie**<br>Displays AVI or MPEG video files accessible from the client computer or stored as a smart large object (BLOB data type) in the database. | The movie can begin automatically and can display VCR controls. | Custom events include timer events, display completion, and state changes. |
| **Pie3D**<br>Displays data in a 3D circular chart that is cut into wedges. Each wedge represents one data point. | The chart can have wedge labels and legends. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |
| **Pie3DSmooth**<br>Displays data in a 3D circular chart with a smooth edge. The chart is cut into wedges and each wedge represents on data point. | The size and the colors in the chart can be specified. The chart can have wedge labels and legends. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |
| **Torus3D**<br>Displays data in a 3D chart in the shape of a doughnut. Each piece of the torus represents a data point. | The size and the colors in the chart can be specified. The chart can have wedge labels and legends. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |
| **Bar3D**<br>Displays one or more horizontal bars along a labeled axis. | The size and the colors in the chart can be specified. The chart can have a legend. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |
| **Column3D**<br>Displays one or more vertical bars along a labeled axis. | The size and the colors in the chart can be specified. The chart can have a legend. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |

*Table 12. ActiveX controls and how to use them (continued)*

| Control/uses | Selected properties | Custom events |
|---|---|---|
| **Oblique3D**<br>Displays one or more vertical bars along a labeled axis with one or more planes. | The size and the colors in the chart can be specified. The chart can have a legend. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |
| **Area3D**<br>Displays areas that begin from the bottom of the chart with drops in the area. | The size and the colors in the chart can be specified. The chart can have a legend. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |
| **Ribbon3D**<br>Displays one or more horizontal ribbons along a labeled axis with drops in the ribbons. | The size and the colors in the chart can be specified. The chart can have a legend. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |
| **Surface3D**<br>Displays a surface that contains adjacent data values. | The size and the colors in the chart can be specified. The chart can have a legend. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |
| **Point3D**<br>Displays data values as points in the chart. | The size and the colors in the chart can be specified. The chart can have a legend. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |
| **Bubble3D**<br>Displays data as bubbles, where the size of the bubbles are determined by the data value. | The size and the colors in the chart can be specified. The chart can have a legend. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |
| **StackBar3D**<br>Displays one or more horizontal bars that consist of stacked data values. | The size and the colors in the chart can be specified. The chart can have a legend. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |

*Table 12. ActiveX controls and how to use them (continued)*

| Control/uses | Selected properties | Custom events |
|---|---|---|
| **StackColumn3D**<br>Displays one or more vertical bars that consist of stacked data values. | The size and the colors in the chart can be specified. The chart can have a legend. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |
| **StackLine3D**<br>Displays one or more horizontal ribbons that consist of stacked data values. | The size and the colors in the chart can be specified. The chart can have a legend. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |
| **StackArea3D**<br>Displays an area that consists of stacked data values. | The size and the colors in the chart can be specified. The chart can have a legend. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |
| **Stock3D**<br>Displays one or more vertical bars on a labeled axis. | High, low, close, trading volume, and trading date for a stock can be displayed. The chart can have a legend. | Actions can be triggered by events once the chart is created or filled with data. |
| **FloatBar3D**<br>Displays floating horizontal bars showing date ranges. | Start and end dates can be specified. The chart can have a legend. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |
| **Radar**<br>Displays a series of data points in a radar screen format. | The size and the colors in the chart can be specified. The chart can have a legend. | Actions can be triggered by standard mouse events, such as clicking or once the chart is created or filled with data. |
| **Polar**<br>Displays data points in a 360-degree radar screen format. | The size and the colors in the chart can be specified. The chart can have a legend. | Actions can be triggered by events once the chart is created or filled with data. |

*Table 12. ActiveX controls and how to use them (continued)*

| Control/uses | Selected properties | Custom events |
|---|---|---|
| **Shmoo3D**<br>Displays multi-series data in a cube-like format. Each cell of the cube is tested to determine if it passes a given condition. | Pass and fail values can be specified. The chart can have a legend. | Actions can be triggered by events once the chart is created or filled with data. |
| **Text3D**<br>Displays text in 3-D format. | Appearance, such as color and angle of plane, can be customized. | Actions can be triggered by standard mouse events, such as clicking or once the text is displayed. |
| **Clock3D**<br>Displays a clock in 3-D format. Time is set by the Windows clock on the end user's computer. | Can be analog or digital. Appearance can be customized. | Actions can be triggered by standard mouse events, such as clicking or once the clock is displayed. |
| **Legend3D**<br>Displays a legend in a 3D format. | Can display multiple rows and columns with different mark shapes. | Actions can be triggered by standard mouse events, such as clicking or once the legend is created or filled with data. |

## Custom objects and the Custom palette

QMF Visionary provides a Custom palette in the Palette Manager where you can save graphic objects you create.

For example, if you use a Text object often in a world but want the size to be 9 point rather than 12 point, and the font to be Palatino rather than Arial, you can save your customized Text object to the Custom palette. Perhaps you want to reuse a Marker object that is defined as a solid triangle shape, red color, and 9 point size; you can save the customized Marker object to the Custom palette.

The Custom palette is particularly useful for storing objects that are used as template graphics in every scene in your world. For example, reusing a title object that has the font, size, frame, color, and fill you want in every scene provides consistency throughout your project.

The following figure shows these custom objects saved to the Custom palette.

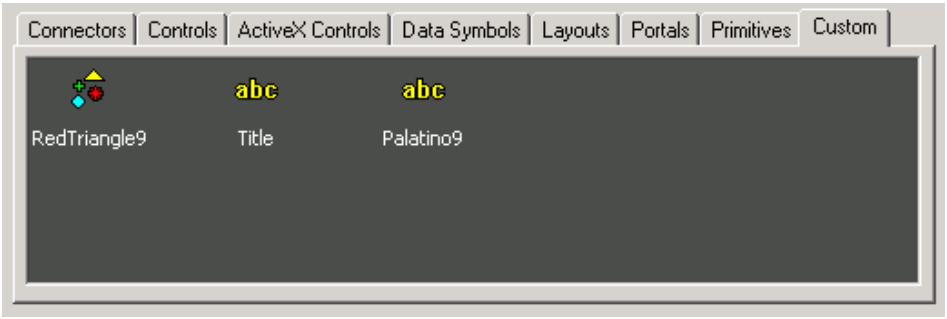



*Figure 27. Custom objects on the Custom palette*

You can also rename your custom objects to identify their significant properties, as illustrated in the figure above.

Similarly, if you create a picture that you want to use in several places in the world, you can specify a particular stock image as the ImageSource property for the Picture object and then store the customized Picture object to the Custom palette.

**CAUTION:**
**You can only store object properties with a custom object. Events and parameters associated with an object are not stored and cannot be reused from instance to instance. You must manually assign event actions and parameterized properties to objects if you want to duplicate those attributes.**

To add a modified object to the Custom palette, drag it from the editor window to the Custom palette. An object named Objectn appears in the palette, where Object is the standard object name and n is an integer. To rename the object, right-click in the Custom palette and click **Rename**.

**Note:** You can also right-click an object in an editor and select Add to Custom palette to save custom objects.

Your custom objects are saved in QMF Visionary Studio and appear in the Custom palette the next time you launch QMF Visionary Studio. Custom objects are not saved with the world, but with QMF Visionary Studio.

To delete a custom object from the Custom Palette, right-click the object you want to delete and click **Delete**.

You can modify a custom object as often as you want by inserting it into an editor window, modifying it, and saving it again with the new properties.

## Storing and using images in Image folders

You can create stock images to use in your QMF Visionary world by storing images in the Globals folder of a QMF Visionary world. When you add Picture objects to scenes or data templates or Animated GIF objects to scenes in your world, you can specify one of two sources: an image stored in the world or an image file from your computer's file system.

**Note:** To understand the distinction between a picture/animated GIF and an image, think of a Picture/Animated GIF as a class of object, while an image is a particular instance of the Picture/Animated object.

Stored images enable you to store your image resources with your world and thereby can make your scenes run faster. They also provide some performance enhancement when they do not require any processing at runtime.

To store an image in the Images folder, click **Insert —> Image** to display the Insert Image dialog box and then enter the file system location of the image and click **OK**. The image appears in the Images folder in the World Manager Worlds page.

To use a stored picture image in a scene:

1. Insert a Picture object into the scene by double-clicking the Picture object on the Primitives palette.
2. Set the ImageSource property of the Picture object to the StockImage() function and specify the name of the stock image you want, as follows:

   `=StockImage(`*`image_name`*`)`

To use a stored animated GIF image in a scene:

1. In the Scenes folder, double-click the scene into which you want to insert an animated GIF.
2. Drag the animated GIF object into the scene from the Images folder of the World Manager Worlds page. The ImageSource property of the animated GIF object is automatically updated with the name of the stock image you want.

**Note:** When you want to reuse images, store them as named pictures in the Custom palette.

## Creating custom color maps and sequences

You can vary the colors of an object depending on its value with the following color functions:

- **Color map**. Sets specific colors for defined value ranges. You can name color map functions and then use them to specify object properties.
- **Color sequence**. Sets each data point in query results set to a different color; colors in a color sequence do not map to specific values.

Color folders are located in the Globals folder of each QMF Visionary world, as shown below.

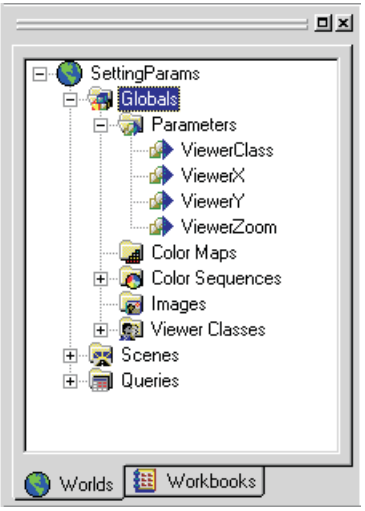



*Figure 28. Globals folder of the World Manager*

### Creating a color map

When you want to specify a function that sets specific colors for defined value ranges, create a color map. Creating a color map using the Insert Color Map dialog box is iterative: as you define each color value, you specify a breakpoint in the sequence, select its color, and view the color map.

For example, you might want a bar chart to display bar colors depending on the values returned by the query you mapped to that layout. If you want the color to change from red to black when your product shows a profit, you can specify a named color map function.

When you define a color map, you specify its breakpoints and its colors. A *breakpoint* is a value at which the color changes.

There are two types of color maps:

- **Discrete**. Displays discrete colors based on the value specified. Discrete maps have one more color than breakpoints.
- **Continuous**.Displays blended colors for values that fall on the breakpoints and discrete colors for non-breakpoint values. Continuous color maps have two more colors than breakpoints.

The number of breakpoints is typically one less than the number of color values. The highest color value represents the range of values from the last breakpoint to the maximum value in the data returned. Similarly, the lowest color value represents the range of values from the minimum value in the data returned to the first breakpoint.

To create a color map, click **Insert —> Color Map** to display the Color Map dialog box.

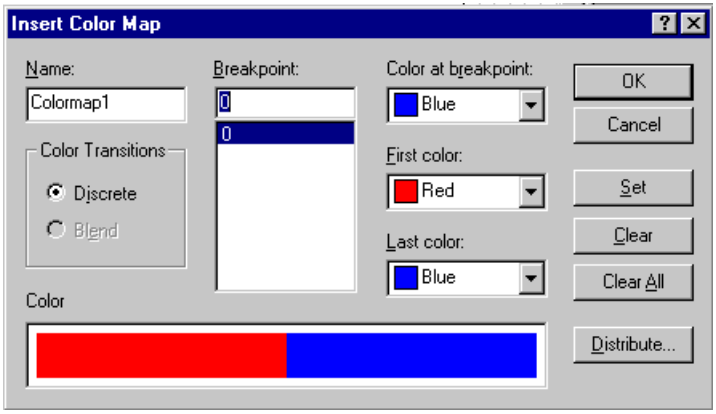



*Figure 29. Insert Color Map dialog box*

Color maps appear in the Globals folder of each world so that you can use them anywhere in your world.

To create a color map with an automatic selection of colors and an automatic even distribution of the colors over the range of values displayed, click **Distribute** in the Insert Color Map dialog box. In the Distribute option, you can select a color pattern such as colors of the rainbow or shades of gray. You can also set the number of breakpoints and a minimum and maximum value and let QMF Visionary create the individual breakpoints for you.

## Creating a color sequence

Color sequences are useful for showing a distinct color for each data point displayed in a layout.

For example, to display each wedge in a pie chart as a different color, you can specify a default or a user-defined ColorSeq1() function. When you create a pie chart, QMF Visionary Studio provides the default color sequence ColorSeq1().

When you create a color sequence, QMF Visionary provides a default set of colors. You can change all or some of the default colors, and you can rearrange their order.

To create a color sequence, click **Insert —> Color Sequence** to display the Color Sequence Properties dialog box.

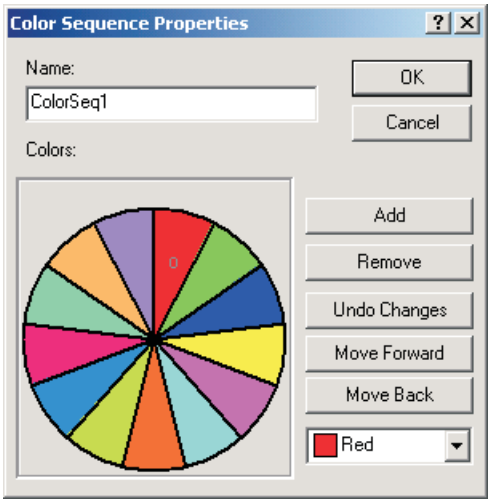



*Figure 30. Color Sequence Properties dialog box*

The default color sequence is shown. You can select other colors from each of the color lists, add, rearrange, or remove color lists, or undo changes made to the color lists and sequence.

When you close the Color Sequence Properties dialog box, the new color sequence function appears in the Color Sequences folder of the world.

## Creating and using viewer classes

*Viewer classes* allow you to filter the data displayed in a world according to the identity of the user. You can expose a viewer class when you publish a world by making it public at runtime. As a result, your users can select the way they want to view your world.

For example, you might want to create a world that shows a series of blueprints for a building. However, the electrician, the plumber, and the city inspector might have different viewing needs. By creating three viewer classes to represent your three types of users and their data needs, you can include all the data in the world but filter out distracting or obscuring information for each user.

**Note:** A viewer class is not a security feature. Viewer classes cannot prevent users with particular login identities from viewing sensitive data.

To create a viewer class, click **Insert —> ViewerClass**, type a name for the viewer class in the Viewer Class dialog box, and click **OK**.

The named viewer class is added to the Globals folder for the world. Every world has a default viewer class called Anonymous.

You can now use the new viewer class when writing an object property expression. For example, you might want to limit the visibility of an object to only one viewer class. In this case, you can use the IsViewer() function, which returns a Boolean result, and specify the named viewer class.

To use a viewer class to specify visibility, set the Visible property of the object to the following function expression, where viewer_class_name is the named viewer class you created previously:

```
=IsViewer(viewer_class_name)
```

**Note:** Test global parameters, such as viewer classes, after you publish your world to make sure they are working as you intended. Viewer classes are always available to end users of a published world, and every end user can access any viewer class.

# Chapter 7. Object properties and events

This chapter introduces the QMF Visionary object hierarchy and explains how to modify and manipulate objects. It also describes how to use columns, parameters, and functions in object property expressions, and discusses some syntax rules for writing property expressions.

In this chapter:

- About objects in QMF Visionary
- QMF Visionary object hierarchy
- Modifying object properties
- Property expression examples
- Creating event actions

For a full reference to objects and their properties, see QMF Visionary Studio online help.

## About objects in QMF Visionary

QMF Visionary Studio provides many objects you can use to develop your worlds, from simple text, lines, and symbols; to controls such as radio buttons and Web browsers; to complex world components, such as layouts and wormholes.

As you develop your world, you insert and modify objects. You can modify object properties and behavior. You can also duplicate, copy, and save objects.

Object properties and behavior vary depending on the object. You can see object properties in the Object Inspector. A full reference to objects and their associated properties and events can be found in the QMF Visionary Studio online help.

You edit an object by modifying its properties, which might include:

- Color
- Position
- Size
- Scaling
- Font
- Borders
- Visibility

- Tooltip text

Many properties supply options for you to select in a drop-down list, such as Maroon or Blue for color values. You can also use functions, column values, parameters, and other resources to modify a property.

You can also assign an event to an object to specify a particular behavior. Object events typically include mouse actions, such as Click and Double-click, and other specialized events for particular objects.

## QMF Visionary object hierarchy

Objects in QMF Visionary Studio are part of an object hierarchy that is reflected in the tree you see in the Worlds page of the World Manager. Objects have a parent-child relationship, with child objects sharing some characteristics of their parents.

Here is an example of a world displayed on the Worlds page.

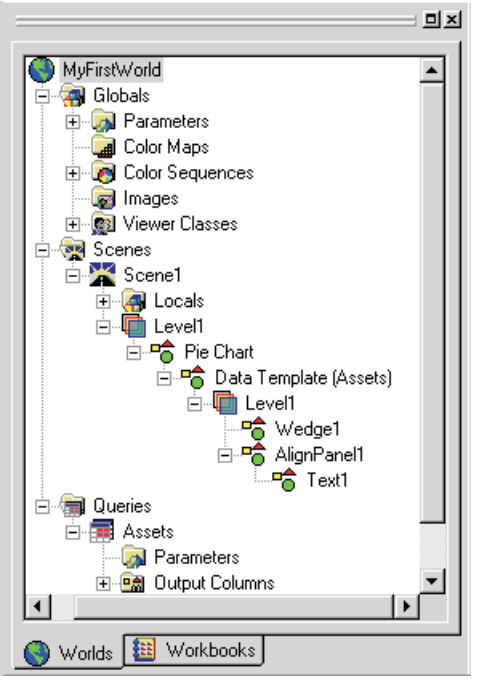



*Figure 31. Objects in a world*

Each object in the world fits into this overall hierarchy. Each world and scene has its own parameters and other resources that can be used within the scope of that world or scene.

As a result of the hierarchical context, you cannot delete an object from the middle of the structure without deleting all the child objects underneath it. In addition, some child objects cannot be moved or copied without the context of their parent objects.

For example, a data template for a particular 2-D layout can be duplicated within the same layout to create the effect called *layering*. However, the same data template cannot be copied to a different type of layout elsewhere in the world.

The layout context dictates the format for the data template; a pie chart template would not work in a linear map layout.

In addition to layouts, each level of detail in a scene might contain child objects, such as text or images. Within each level of detail in a 2-D layout data template also are child objects, such as text, images, data symbols and so on.

## Modifying object properties

This section talks in general about objects. Both 2-D and 3-D layouts are discussed in detail in Chapter 4, "Creating scenes," on page 43.

Each object has a unique set of properties you can modify. You can specify a value for an object property and thus change its appearance. For example, you can change several Font properties for a Text object.

In general, there are three ways to modify a property:

- Map an SQL column or a parameter to the property.

  When you map an object property, you create a direct link from the value of the property to the value of a variable, such as a parameter or a column from a query result. In most cases, when you create a layout, the Layout wizard helps you to map data points to columns returned by your query.

  For more information, see "Mapping object properties to column or parameter values" on page 94.
- Write a property expression, using built-in functions, parameters, columns, constants, or a combination of these.

  QMF Visionary provides built-in functions you can use in expressions, such as the logical If() function; Max(), Min(), and other mathematical functions; geometric functions; and functions that count data points in a set or

numbers of child cells in a hierarchy. For a full list of functions you can use to write property expressions, see the Function Reference in the QMF Visionary Studio online help.

Writing property expressions is described in "Writing property expressions" on page 95.

- Select a value from those provided in the list box, if one is provided.

  QMF Visionary supplies default values and, in many cases, a list box of valid values for a property. For example, a color picker is displayed when you select the Color property. To see a list of optional values for a property, in the Object Inspector, click in the entry cell adjacent to the property name.

## Mapping object properties to column or parameter values

To manually map an object property to a column or parameter value, select a variable type and a variable on the Variable bar and then click a property name in the Object Inspector.

You can also type the column or parameter name preceded by an equals sign (=):

```
=column_name
```

The following table shows situations where you might want to map a column from an SQL query to a property value.

*Table 13. Examples of mapping a column from an SQL query to a property value*

| Mapping task | Procedure |
|---|---|
| Change the data symbol used in a 2-D layout | 1. Delete the default data symbol. 2. Insert the new symbol. 3. Map the appropriate column to the Value property of the inserted data symbol. |
| Label each data point in a 2-D layout | 1. Insert a Text object into the data template. 2. Map the column you want to be displayed in the label (for example, customer_name) to the Value property of the Text object. |
| Map an object property when the Layout wizard does not provide this step. | 1. Open the data template of the layout. 2. Insert the object you want to be mapped to the column value. For example, insert a Text object if you want to display the column value as text. 3. Map the column to the Value property of the inserted objects. |

*Table 13. Examples of mapping a column from an SQL query to a property value (continued)*

| Mapping task | Procedure |
| --- | --- |
| Modify the mapping after you change the query for a data template. For example, you might want to duplicate a layout and data template but display different data. | 1. Open the data template. 2. Right-click and click **Change query** and then select a new query in the dialog box. 3. In the revised data template, map the column you want to the Value property of the appropriate object in the data template. |

## Writing property expressions

You can write *property expressions* to define object properties based on functions, columns, or parameters that supply values at runtime. Property expressions can be simple, such as a column returned by a query. Property expressions can also be complex, such as an If () statement combined with a parameter and a concatenated string of literal values.

Property expressions enable you to create dynamic values for your object properties. Many business decisions depend on factors such as market conditions, sales, operational expenses, new locations, profit and loss margins, and customer satisfaction. These factors are subject to change.

Your QMF Visionary world can be responsive to these changes and show their significance in the appearance of the world. Profit or loss, for example, can be displayed visually using color: black for profit, red for loss.

Here is a property expression that you might use to distinguish profit and loss in the Color property of the object:

```
If(sum(sales/expenses)>=0,Black,Red)
```

The following sections provide syntax guidelines and examples to help you write property expressions in QMF Visionary.

### Supplied property values

Many QMF Visionary objects have a list of constant values you can use to define a constant property.

The following figure shows an example of the supplied values for the LineStyle.Pattern property.

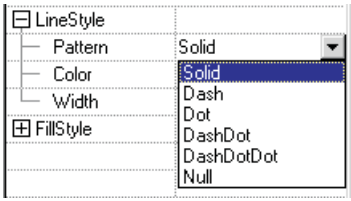



*Figure 32. LineStyle.Pattern supplied property values*

By looking at the supplied property values in the Object Inspector, you can also determine the data type that must be returned when you write a property expression. For example, the Visible property (a property of most QMF Visionary objects) supplies True or False as options in the property list box. You can also write a property expression for the Visible property that returns a true or false result.

For a full reference to QMF Visionary objects, object properties, and property data types, see the QMF Visionary Studio online help.

### Syntax rules for property expressions

When you write property expressions, you must use the syntax required by QMF Visionary. The following table describes some basic rules for typing property expressions in the Object Inspector or the Formula bar.

*Table 14. Basic rules for property expressions*

| Task | Rule | Example |
|---|---|---|
| Display a calculated value, such as a parameter or column name | Use equals sign (=) for calculated expressions. | `=cust_num` |
| Display a literal value | Type the literal exactly as you want it to appear in text. | `Sales Organization` |
| Display a text string in an expression | If the string is part of a combined expression that also includes calculated values, columns or parameters, it must be surrounded by quotation marks (" "). | `="My name is: "+fname` |

*Table 14. Basic rules for property expressions (continued)*

| Task | Rule | Example |
|---|---|---|
| Concatenate two or more elements of ambiguous or differing data types into one expression | Use the Concat() function to concatenate elements in an expression. | `=Concat(fname,lname)` |
| Concatenate two or more text strings | Use the plus sign (+) to concatenate text strings. | `="Click here to go to"+newline+ "Scene2"` |
| Add two or more numeric values | Use the plus sign (+) to add constants or parameters of numeric data types. | `=Param1+5` |
| Insert a carriage return in an expression | Type `newline`. If the expression elements are the same data type, use a plus sign (+) before and after. Otherwise, include newline in the Concat() function.**Warning:** ToolTipText and Caption properties do not allow multiple lines of text. | `=city+newline+state` |
| Display the value of another property | Use the equals sign (=) plus the object name and property name with dot notation. | `=Text1.value` |

This following sections describe these syntax rules in more detail and describe formatting, units of measure, and casting rules in QMF Visionary Studio:



In addition to these syntax rules, the QMF Visionary Studio online help contains detailed reference entries for all functions provided by the application.

### Calculated expressions versus literals

QMF Visionary treats all object properties as literals, unless the value entered begins with an equals sign (=). The following figures show two property

expressions for the ToolTipText property of a Button object: the first defines a literal value, and the second defines a calculated value.


*Figure 33. Literal value for the Tooltip property of a Button object*


*Figure 34. Calculated value for the Tooltip property of a Button object*

Try it. Use the = sign in an expression and then try the same expression without the sign. View the results in runtime. For example, type a column name from the query for your layout. View the object in runtime. Then precede the column name with the = sign.

### Concatenation

You can concatenate parts of a property expression in two ways:

- **Plus sign ( + )**. Used for mathematical calculations and for concatenating text strings.

  When the parts of the property expression are of unambiguous data types, you can use the plus sign. Thus, two text strings can be concatenated explicitly using the plus sign, and two integers can be added using a plus sign.

  However, if the parts of the expression have different data types, you can use the plus sign.
- **Concat() function**. Used to concatenate values that have different data types.

  To concatenate more than two elements, you must embed more Concat() expressions in the outer Concat() expression.

Therefore, you can use the plus sign to add parameters or columns that all have a numeric data type. You can also use the plus sign to concatenate two columns that contain text or character data types. However, to combine a literal text string with a column that contains a numeric data type, you must use the Concat() function as shown.


*Figure 35. Using the Concat() function*

To combine a parameter that has a text data type, a literal text string, and a column that has a numeric data type, you must use embedded Concat() functions, as shown.


*Figure 36. Embedded Concat() functions*

### Custom formatting for property values

QMF Visionary provides a custom formatting function and templates for displaying property values in special formats. The FormatNum() function transforms the value you supply into the display format you specify.

The syntax for FormatNum() is as follows:

```
FormatNum(format,value)
```

In this expression, *format* is one of the templates provided by QMF Visionary, and *value* is the value you want formatted.

The function FormatNum() is similar to custom number formats in Microsoft Excel. It allows you to specify how you want your object properties to be displayed. You can either type the syntax for the format template or select the format template from a list of available templates. To select a format template click **Edit —> Format Number**.

FormatNum() can be used to format the following types of values:

- **Numeric**. Values can be expressed as either positive or negative; both templates are provided.
- **Exponents**. Values can be expressed in mantissa-exponent formats.
- **Dates and Times**. Values can be expressed in days of the week, quarters, hours, A.M. and P.M., and so on.
- **Currencies (money)**. Numeric values can be displayed as currency using the currency symbol stored in the Windows registry, such as $ or DM.

**Note:** You can embed a literal character in a FormatNum() argument by preceding it with the backslash (\) character.

For a description of the syntax and templates you can use with the FormatNum() function, see the reference section of the QMF Visionary Studio online help.

### Unit abbreviations for dimensions and location

QMF Visionary Studio allows you to enter the following unit specifications in a property value.

*Table 15. Unit abbreviations*

| Units | Abbreviation Formats |
|---|---|
| Inches | in, inch, inches |
| Centimeters | cm, centimeter, centimeters |
| Millimeters | mm, millimeter, millimeters |
| Points | pt, point, points |
| Twips | tw, twip, twips |

If no units are specified, the default units are inches or the default setting in your Windows registry.

**Note:** Axes in chart layouts display the units provided by the data values mapped to the axes.

### Explicit and implicit casts

QMF Visionary Studio automatically casts some data types when a scene is executed. For example, in any property that requires a text value, you can write a property expression that returns any data type (Boolean, Color, PointList, Integer, and so on). Thus, the property displays these values as text strings: True, Blue, 1, and so on. However, you cannot cast an expression that returns a PointList value into a property value that accepts only a MultiPolygon type.

If you want to explicitly cast an object property value to a different data type, QMF Visionary provides functions for explicit casting.

See the reference section of the QMF Visionary Studio online help for a list of implicit casts and a list of the explicit casting functions.

### Built-in function libraries

QMF Visionary provides many built-in functions and a standard function library (StdFunc.vlb) that you can use to write property expressions. In addition, you can include your own function libraries in QMF Visionary Studio by placing the library .dll file in the same directory as the QMF Visionary Studio .exe file. When QMF Visionary Studio is launched, it recognizes and loads your function library.

The following table describes the types of built-in and standard functions provided.

*Table 16. Built-in and standard functions*

| Function category | Description | Example |
|---|---|---|
| Color | A set of functions that provide information about and perform calculations on colors | **Brighten(*color*, *amount*)** returns a color brightened by the specified amount. |
| Conversion | A set of functions that convert data types | **NumToBool(*number*)** returns a Boolean value. |
| Data Formatting | A set of functions that convert data between database format and Windows regional format | **DbNumToWin(*number*)** converts a number from database format to Windows format. |
| Date and Time | A set of functions that perform date and time calculations | **Date(*year*, *month*, *day*)** returns a numeric value that represents the specified date. |
| Hierarchy | A set of functions that are valid for hierarchy layouts | **SiblingCount()** returns the number of sibling data instances in the current data template. |
| Information | A set of functions that return information | **StockImage(*imageName*)** retrieves the specified image from the Images folder of the world. |
| Logical | A set of If() functions that accept various argument types and return various value types The If() function returns Boolean values if the supplied values for valueIfTrue and valueIfFalse are Boolean. Similarly, it returns other data types based on the argument data types. | **If(*condition*, *valueIfTrue*, *valueIfFalse*)** |
| Math and Trigonometry | A set of functions useful for calculating angles, degrees, exponents, and other mathematical formulas | **Abs(*number_value*)** returns the absolute value of a numerical argument. |
| Spatial | A set of functions that scale, convert, rotate, and compute spatial values | **CenterPoint(*pointList*)** returns the center point for a list of points. |
| Statistical | A set of functions that apply statistical calculations | **StdDev(*number_value*)** returns the standard deviation in a series. |

*Table 16. Built-in and standard functions (continued)*

| Function category | Description | Example |
|---|---|---|
| Text | A set of functions that examine text values | **Trim(*text_string*)** returns the text string with any blank spaces to the left and right of the string removed. |

For a complete list of built-in functions and the standard function library, see the reference section of the QMF Visionary Studio online help.

## Property expression examples

You can use either the Object Inspector or the Formula bar to write property expressions. The following sections provide some examples.

### Writing an If() statement

You can use an If() function to create a binary property value. For example, if you want to display the object as blue if the condition specified is met and yellow if it is not met, you might write the following expression:

```
Color   =If(S_terrain="water",Blue,Yellow)
```

In this example, the condition of the object is based on a scene parameter, S_terrain. The object might be a polygon in a map layout or a data symbol shown on a chart. If the value of the scene parameter is water, the color of the object is blue. Otherwise, the color is yellow.

You can specify different conditions for the If() function, such as a column value or a numerical threshold.

**Note:** There are several types of If() functions, depending on the data type of the conditions specified. When you review function entries in the Function Reference in QMF Visionary Studio online help, look for the If() functions under data type, such as BooleanIf().

### Displaying column data in a text string

When you modify a data template, you can enhance the data points for your layout by adding a Text object that reveals information about the record returned.

For example, you might want each item in a tree chart to show the name of a department in a store, preceded by the word Department. You could insert a Text object into the data template and then specify a property expression which combines the column name and a literal string.

```
Value     ="Department:"+newline+dept_desc
```

In this expression, *dept_desc* is a column returned by the query to your data template.

## Passing data from user input

You might want to create user options for display purposes. For example, you might provide two ways to view a column chart: in terms of sales or in terms of profits. While you could show both in the same chart, you might want to allow your user to isolate the data and look at only one of these options.

To create user options that change a display, you must capture the user's desired choice using an event action. For example, you might create two data symbols: one is a green circle with the word Sales next to it, and the other is a black diamond with the word Profits next to it. In a Text object, you can instruct your user to "Click either Sales or Profits."

Create the global parameter Gshow to store information about the user's choice. Then create the event actions associated with clicking the green circle or the black diamond. The following figure shows how to define one of these event actions in the Set Parameters page of the Object Behavior dialog box.

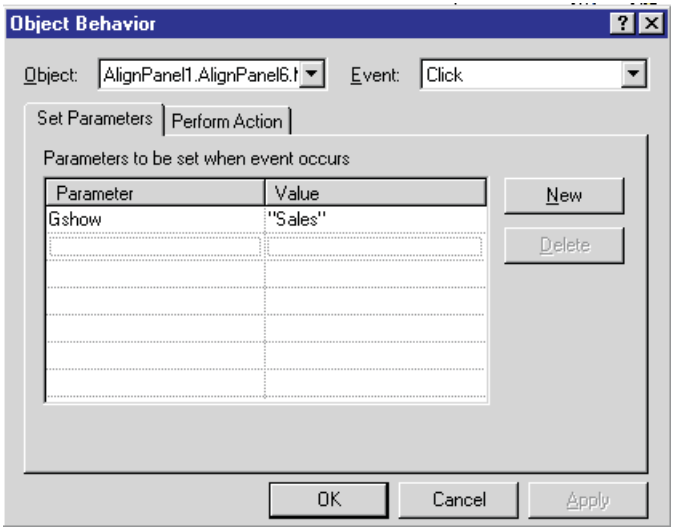



*Figure 37. Setting a global parameter in an event action*

Finally, use the global parameter Gshow to specify which version of the column chart is displayed—sales or profit.

To pass the user input stored in Gshow, write an If() expression for the Value property of the VerticalValueBar object in the data template for the chart.

Here is what that property expression might look like:

```
Value     =If(Gshow="Sales",total_sales,total_profit)
```

In this expression, total_sales and total_profit are columns returned by the query.

You can also write embedded If() statements to specify multiple conditions beyond the two possibilities in the outer If() statement.

## Passing a value to a query

To pass values, you use parameters. Parameters are typically used in object property expressions as placeholders for calculated values to be set at runtime.

Suppose you want to create a query that returns customer buying information for each department in a store. However, you want your users to specify the department. You must first create a query that uses a query parameter. The value of the query parameter is based on information passed from the user's choice.

You can provide user choices in various ways: option buttons, a list control, or text instructions that tell the user to click the relevant data point in a related layout. Then, as described in "Passing data from user input" on page 103, you assign an event action to any object the user might choose.

To pass information about the user's choice to a query, you can use a global or scene parameter to supply the value of a QueryParameters object.

Here is an example of how you can specify the value of the QueryParameters object:

```
Q_dept=G_department
```

In this expression, G_department is the global parameter for the corporate department, and Q_dept is the query parameter.

This is similar to the example in the section "Passing data from user input" on page 103. In that example, the global parameter was used to create a dynamic chart. The query in that example returned one set of values, and the user input specified which column was used in the data template. In this example, the global parameter changes the query results.

## Formatting a time value

To specify a particular format for a time value, you use the FormatNum() function.

For example, to display the value of the column transaction_time in minutes, you might enter the following functional expression:

```
Value     =FormatNum(mm,transaction_time)
```

The format argument mm represents minutes when a row returns minutes data and represents months when a row returns months data.

## Creating event actions

You can assign event actions to objects to provide interactivity and navigation. *Event actions* automatically execute an operation, function, or calculation as a result of a triggering event, such as clicking the mouse. You can assign event actions to any object you create, from controls to primitives to ActiveX movies.

You create an event action by selecting an event for the object and then assigning it an action in the Object Behavior dialog box. You can also set parameters when an event occurs or combine results so that both an action is performed and a parameter is set.

Object events and actions are also listed in QMF Visionary online help reference sections.

You can specify the following actions:

- **Jump to a new location**. The location can be either another scene or another viewpoint. If the destination scene has any scene parameters defined, you can set them in the Jump Destination field in the Object Behavior dialog box.

  You can additionally set global parameters using the Set Parameters page of the Object Behavior dialog box.
- **Launch another application**. Allows the user to open another application that resides on the client computer or to execute a batch file. You must specify a full pathname to the program.

  You can specify that QMF Visionary pause until the user exits the other application.
- **Enter wormhole**. Allows you to assign this action to a particular object, such as a Button object or a Text object. You must specify a destination scene or viewpoint, and can optionally set a scene parameter on the Perform Action page.

  Users can enter a wormhole using the Enter Wormhole tool or by clicking the wormhole if you selected the appropriate option when you created the wormhole. However, you might want to assign this action to an object other than the wormhole.
- **Exit wormhole**. Allows you to assign this action to a particular object.

Users can exit a wormhole using the Exit Wormhole tool. However, you might want to create a more explicit instruction and assign an event action to the object displaying the instruction, such as a Button object.

- **Execute SQL statement**. Allows you to execute a database operation, such as an UPDATE statement or stored procedure. You must specify an open data source or click Connect to open a connection, and you must specify the SQL statement you want to execute.

  You can also save the query result in either a global or local parameter.
- **Execute Script**. Allows you to execute a script written in VBScript or JavaScript.
- **Print a Scene**. Allows you to assign this action to a particular object, such as a button object.

  Users can click a button on a particular scene and have the scene printed to their printer.
- **Execute a Shell Command**. Allows you to execute a shell command like that performed using the Windows Run command from the Windows Start menu.

You can access the Object Behavior dialog box either by double-clicking an event name in the Events page of the Object Inspector or by selecting an object in the Worlds page and clicking **Edit —> Behavior**. For instructions on creating event actions, see the *DB2 QMF Visionary Getting Started Guide* and the QMF Visionary online help.

# Chapter 8. Managing and publishing worlds

This chapter describes how to manage worlds with the World Structure Editor and to publish worlds.

In this chapter:

- Viewing and modifying world structure
- Publishing worlds
- Deploying worlds
- Testing worlds

## Viewing and modifying world structure

At any time during world design in QMF Visionary Studio, you can view the overall world structure in the World Structure editor. The World Structure editor has two views:

- **Pseudocode view**. Displays read-only text of the codelike elements in a world.
- **Structure view**. Displays an editable hierarchy of a world that can be expanded and collapsed and also shows parameter connections.

### Using the Pseudocode view

The Pseudocode view allows you to search your world for particular codelike elements and parameter settings. It includes the following elements:

- World name
- Initial scene
- Global parameters
- Viewer classes
- Color maps
- Color sequences
- Stock images
- Database connections
- Queries
- Scenes

Because all of its contents are text, you can click **Edit —> Find** to find text in the Pseudocode view. To find the next instance of the text, click **Find Next**.

The following figure shows part of the pseudocode for a world.

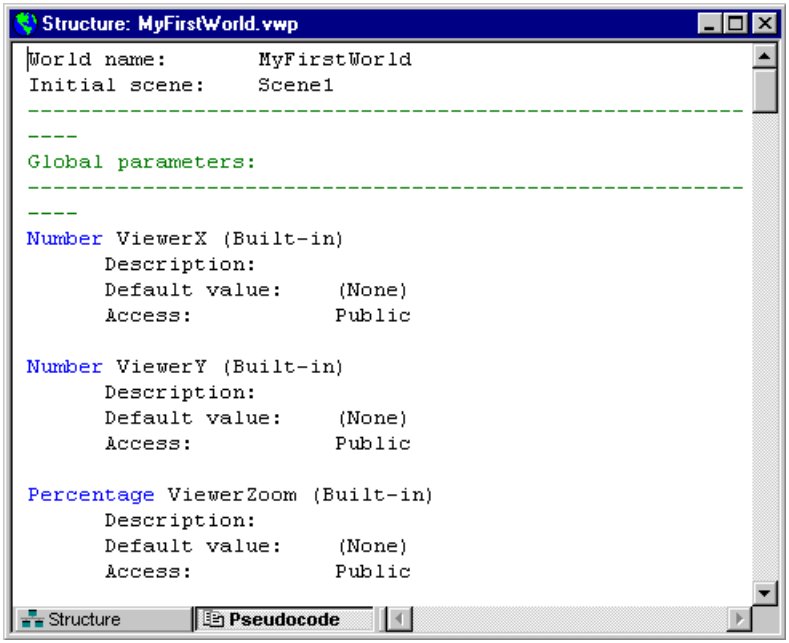



*Figure 38. Pseudocode view*

The pseudocode elements are listed in sequence according to their place in the world object hierarchy. For more information about the object hierarchy, see "QMF Visionary object hierarchy" on page 92.

Data types appear in blue text. For example, in the figure below, the first two global parameters are Number data types. Objects are listed with their

associated properties and events, as shown below.

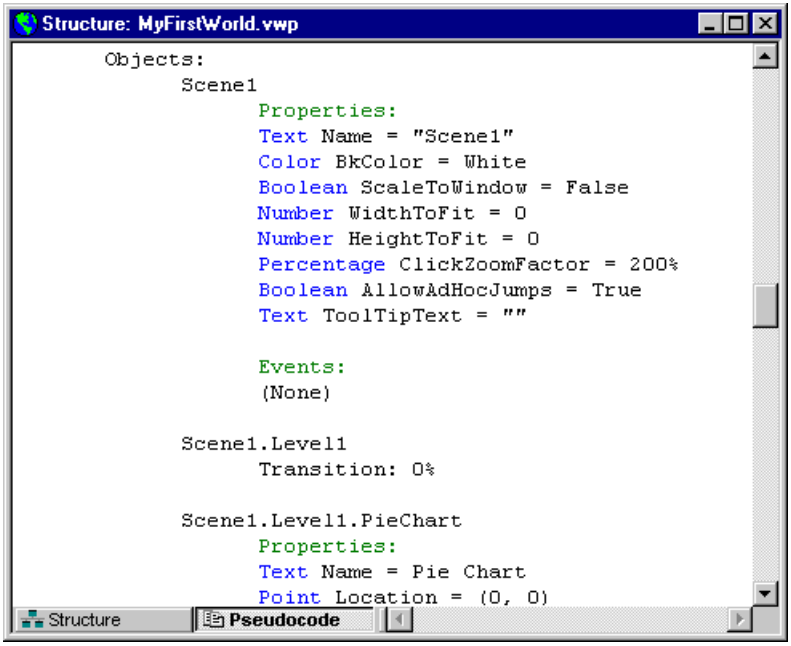



*Figure 39. Objects in Pseudocode view*

**Note:** Pseudocode is read-only; you cannot edit the contents to modify the world.

## Using the Structure view

The Structure view provides a hierarchical view of your world, similar to that provided by the Worlds page of the World Manager. It contains the following elements:

- World name
- Scenes
- Wormholes
- Parameters (global, scene, and query)

  Output parameters can pass information to other structure elements and are shown on the right of a structure element. Input parameters can receive information and are shown on the left of an element.

The Structure view displays only scene, query, and parameter relationships. This condensed and collapsible view allows you to focus on parameter relationships and to edit those relationships.

*Wiring a parameter* means to pass information from the top level of the world or from a particular scene to another scene or query lower in the hierarchy. The following figure shows a scene parameter wired to a query parameter.

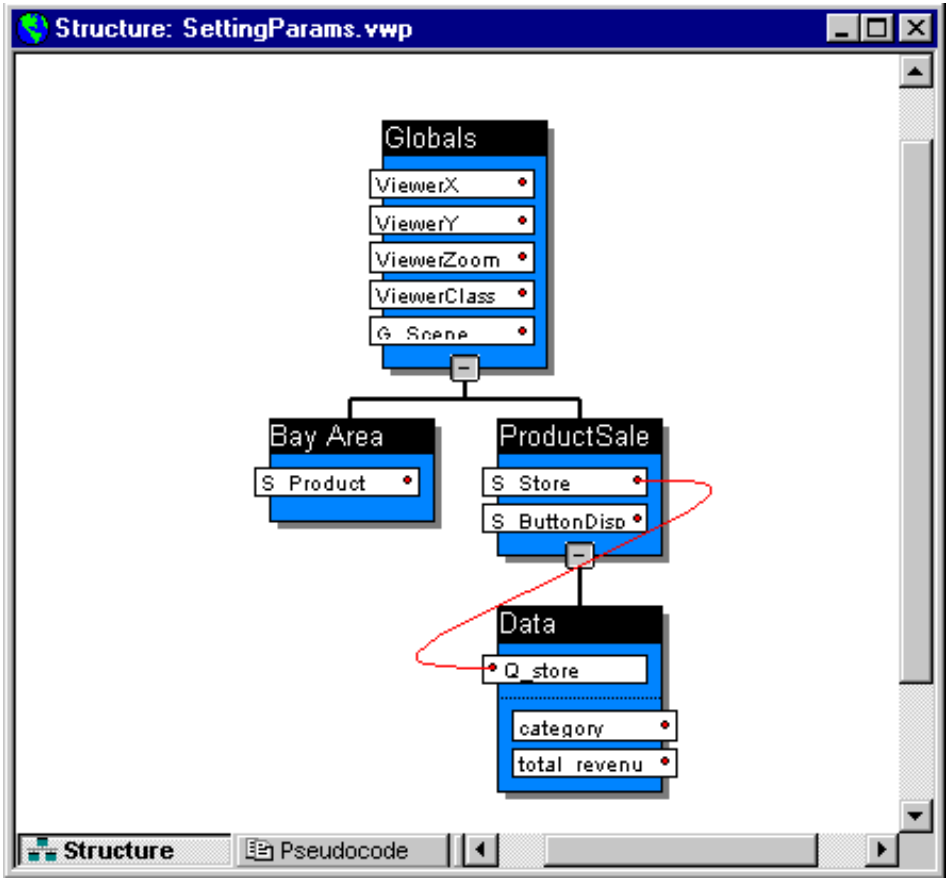



*Figure 40. Wiring parameters*

The scene parameter S_Store passes the store name to the query parameter Q_store. The data display changes according to the value of the scene parameter, thus making the scene dynamic.

You can wire a parameter directly in the World Structure editor. Your world must contain the parameters before you can link them in the editor.

To wire a parameter, click the red dot on the right side of the output parameter, move your cursor to the red dot on the left side of the input parameter and then click again. A red line now links the two parameters together.

**CAUTION:**
**The World Structure editor allows you to wire any input and output parameters together and does not prevent you from wiring incompatible data types. When you execute a scene in the world containing a wired parameter, be sure to check the Output window for error messages, such as data type conflicts.**

To remove a parameter wire, click **Edit —> Undo Wire Parameter**. If you have wired several parameters, you can click **Edit —> Undo Wire Parameter** for each parameter wired during the current session in the World Structure editor.

**CAUTION:**
**You cannot use the World Structure editor to remove parameter wiring that was created using any other feature of Visionary Studio. Also, you cannot remove parameter wiring that was created in previous saved sessions of Visionary Studio.**

To modify previously saved parameter wiring, you must use the Object Inspector or Formula bar to edit the object property that includes the parameter name. See Chapter 7, "Object properties and events," on page 91 for information about modifying object properties.

## Publishing worlds

A published world is a .vis file that contains the byte code defining the world. The file is compiled when you complete the steps in the QMF Visionary Publish wizard.

To publish a world, click **Tools —> Publish World** to launch the Publish wizard.

When you use the Publish wizard, you have the following options:

- Deployment
- Name and location
- User access control
- Database connection
- Row return limits

Publishing a QMF Visionary world should be a collaborative effort in which you as the application developer work together with your system and database administrators. Although you perform the publishing tasks with the Publish wizard in QMF Visionary Studio, publishing options require decisions from system administrators.

## Selecting deployment options

Deploying a world to a client/server architecture on a local or networked drive results in a two-tier architecture, as shown in the following diagram.

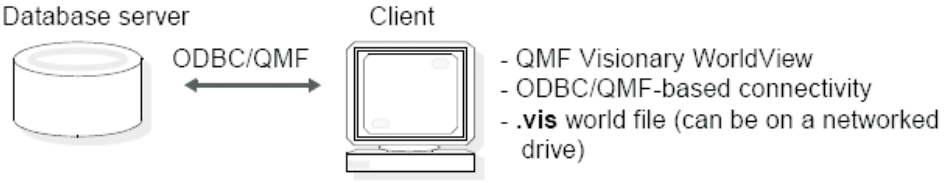



*Figure 41. Client/Server deployment*

An end user accesses a QMF Visionary world by opening its world file (.vis). The end user views a world with QMF Visionary WorldView. QMF Visionary WorldView communicates with the database server through ODBC/QMF-based connectivity.

**Note:** QMF Visionary WorldView is installed along with QMF Visionary Studio and QMF for Windows. Therefore, the end user must have either of these applications installed to view QMF Visionary worlds with QMF Visionary WorldView.
**CAUTION:**
**Client computers must have a data source configured for every data source used by the QMF Visionary world.**

The world file (.vis) contains the published world. It can either be located on the client computers or be accessible from a networked drive; every client computer can have a copy of the same world file, or all clients can access the same world file from a networked drive.

The advantage of keeping the world file on a networked drive is that the world file is automatically updated when it is republished. You need to update the world files on client computers one by one.

## Selecting world name and location

When you specify a name for a world, QMF Visionary uses that name for the world file (.vis).

When you publish to a local or networked drive, you select a location on the local computer or a networked drive. You can then give your users the location of the .vis file, move it to another location, or give each user a copy of it.

## Selecting user authentication options

You can specify one of two authentication options for users of your world: login authentication or anonymous login.

### Login authentication

Each user is prompted by the first database accessed for login information; often this is the database where the world is stored.

**Note:** If the world accesses more than one database, individual logins might not be valid for all databases, so consider using anonymous authentication. You can then supply security with an application.

### Anonymous login

All users can access a world and all data in the world from a shared user name and password. Anonymous access login information is encrypted and stored in the world file (.vis).

**Note:** If the world has connections to one or more databases, you can use different user name and password combinations for each connection.

## Selecting database connection options

Your database connection options vary according to how the user is authenticated. If the world uses login authentication, you can specify the number of connections each user can have for each database. Connections are not pooled; each user gets a separate connection.

Typically, the number of connections for each data source used by a world is equal to the number of queries to any one data source that are executed simultaneously. Specifically, the number of connections you need for a particular data source is proportional to the following factors:

- The number of packets per the maximum result set of any one query
- The maximum number of queries to a single data source that are executed simultaneously

For example, if your packet size is the same as the maximum result set, and the most queries to a data source in a scene is five, you should have at least five connections. You might need more connections if your concurrent usage is high.

## Limiting returned rows

You can specify a limit on the number of rows returned for queries as a performance safeguard. In a production environment, queries might return an unexpectedly large number of rows, which can place an unacceptable load on the database server and the network, as well as slowing response time for the end user.

If you do not know what a safe row limit is for your world, accept the default data set limit of 1000 rows.

## Deploying worlds

Perform the tasks described in the following sections to deploy QMF Visionary worlds. For a description of client/server architecture, see "Selecting deployment options" on page 112.

### Configuring the client

Complete the following tasks to enable client computers to access QMF Visionary worlds:

- Install QMF for Windows or QMF Visionary Studio on each client computer.
- Configure data sources on each client computer. Configure data sources for each data source the world accesses. For instructions, see "Managing connections" on page 29.
- Install the world file (.vis) on each client computer, or notify the users of the location of the world file and give them permission to access that location. You can allow users to download the world file from a Web page.

### Modifying and removing deployed worlds

To modify a published world, the developer republishes it using QMF Visionary Studio. If you have only one copy of the world file (.vis) on a networked drive, the world file is automatically updated when it is republished to that location. However, if you have distributed the world file to client computers, you need to overwrite the old files with the new files.

To remove a published world, delete the world file.

## Testing worlds

You can test most of the functionality of your world using QMF Visionary Studio in runtime mode. If you have specified default values for your parameters, you should be able to run your scenes, view data displays, and navigate throughout the world.

If you encounter problems with any of the features you have created, such as event actions or layouts, the first place to look for errors is the Output window. Typically, query syntax, connection, data source definition, network, and data type errors appear in the Output window.

You might want to test your published worlds, as well. Consider the following areas for troubleshooting, depending on the publishing options chosen for the world:

- **Connections**. You must have working, defined data sources on either the client computer.
- **Navigation instructions**. Consider having an associate test your published world. Your users might overlook what seem like intuitive navigation actions if these actions are not clearly documented with instructions in your scenes.

Many problems you might encounter with a published world can be solved by modifying the world in QMF Visionary Studio and then republishing it.

# Chapter 9. Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10594-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:**
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will

be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
J46A/G4
555 Bailey Avenue
San Jose, CA 95141-1003
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement, or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

## Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX
C/370
CICS
COBOL/370
DataJoiner
DB2
DB2 Information Integrator
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
GDDM
IBM
IBMLink
IMS
iSeries
MVS
OS/390
Parallel Sysplex
PL/I
QMF
RACF
S/390
SQL/DS
VM/ESA
VSE/ESA
VTAM
WebSphere
z/OS
zSeries

Java or all Java-based trademarks and logos, and Solaris are trademarks of Sun Microsystems, Inc. in the United States, other counrtries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other company, product, and service names may be trademarks or service marks of others.

# Glossary

## A

**alignment panel.** An invisible reference object you can use to position child objects relative to a specified point. Objects on an alignment panel are not scaled by layouts or axes.

## B

**breakpoint.** A value at which a color in a colormap changes from the previous color to the next color.

## C

**chile depth.** The number of child levels in a hierarchical layout. A child depth value of 1 defines a hierarchy with one parent object and its children. A child depth value of 2 defines a hierarchy with a top parent object, and its children, and its children's children.

**child scaling factor.** Factor used to determine the size of each child element relative to its parent in a hierarchical layout. For example, in an organizational chart, if the supervisor box is one square inch and the child scaling factor is .5, the employee box is displayed as half the size of the supervisor box, or one square half-inch.

**color sequence.** A type of resource that contains a set of predefined colors that can be accessed by an integer index. Color sequences are user-defined functions that can be used when defining the attributes of an object.

**connection point.** A type of resource that can be used to vary the color of an object depending on a value associated with the object. Colormaps are user-defined functions that can be used when defining the attributes of an object.

**connection point.** An invisible point whose coordinates are used as a reference for a link that extends from one data point to another data point.

**connector.** A graphical object that provides links between data points in a layout.

## D

**data locator.** A data template child object that defines the location and scaling of data points in a layout.

**data symbol.** A graphical element used to represent a data point in a layout. QMF Visionary provides default data symbols for each layout, but users can substitute the data symbols they want and modify their properties. Examples of data symbols are markers, candlesticks, event bands, and price interval bars.

**data template.** The portion of a layout that is based on the query and displayed according to the layout style. The properties and child objects of a data template determine the format of each data point in a layout.

**destination scene.** A scene to which a wormhole connects. See also source scene, wormhole.

**destination scene offset.** A set of x- and y-coordinates that specify the center point of the destination scene that appears when it is displayed through a wormhole in the source scene.

## E

**event action.** An action that executes a function in response to an event. Events occur through user actions such as a mouse click.

## G

**global parameter.** A value assigned to a variable that is available to all scenes in a world and to the end user.

## L

**layer.** One of a set of data templates in a single layout. For example, an XY chart that displays auto sales over a calendar year and advertising spending over a calendar year contains two layers.

**layout.** A display format that determines how data appears in a scene. A layout that displays data from an SQL query contains data points representing each row returned by the query. Examples of layouts include pie chart, stock chart, and linear map.

**level of detail.** A view of data that is associated with a zoom level. There are two types of levels of detail: scene level and data template level. Increasing the level of detail provides more screen space to present detailed information for each data point.

## M

**maximum child depth.** The maximum number of child levels that can be displayed for a hierarchical layout. As you zoom into a hierarchical layout, the maximum child depth property controls the viewable number of levels at each zoom level.

## P

**primitive object.** A graphical object such as a line, rectangle, or polygon that is used in the graphic design of a scene or data template.

## Q

**query parameter.** A value assigned to a variable that is set at runtime before an SQL query is executed.

## S

**scene.** The graphical representation of information retrieved from a database. A scene can also include other graphic elements, such as static text and images, and navigational tools, such as wormholes and jumps to other scenes.

**scene parameter.** A value assigned to a variable that is passed from one scene to another. In QMF Visionary, wormholes and event actions pass scene parameters.

**source scene.** A scene with a wormhole object in it that links it to a destination scene. See also destination scene, wormhole.

**storyboard.** A functional specification for a QMF Visionary application. It includes diagrams of the scenes, how they are linked, and the queries to be used.

## V

**viewer class.** A type of resource that enables a user to restrict the appearance or behavior of a world based on a viewer's identity. Viewer classes are hierarchical.

**viewpoint.** A three-dimensional location for viewing a scene. A viewpoint has three properties: x-coordinate location, y-coordinate location, and zoom level. The x- and y-coordinate locations are set from the center of the scene.

**visibility.** A property that determines whether a data template or other object can be viewed in the scene. Visibility is a Boolean data type; it can be specified with a property expression that returns a value of true or false. In a hierarchical layout, visibility of child data fields is determined by the combination of child scaling factor and zoom factor. See also child scaling factor.

## W

**workbook.** A filtered view of database objects, including tables, views, synonyms, procedures, functions, and data types.

**world.** A display of multiple graphical representations of related data driven by dynamic SQL queries. A QMF Visionary world consists of scenes, rather than forms or reports. See also scene.

**wormhole.** A special QMF Visionary object that links two scenes and can carry context information (parameters) from the source scene through to the destination scene.

# Index

## D



## E

## N



## O



## P



## Q

## R



## S



## T



## U



## V

## W



## X



## Z

Program Number: 5724-E86, 5625-DB2

Printed in USA

SC18-9093-01