INFORMIX®-Data Director™ for Visual Basic

User Guide

Version 3.5
July 1998
Part No. 000-5160

Published by INFORMIX® Press

Informix Software, Inc.
4100 Bohannon Drive
Menlo Park, CA 94025-1032

ACKNOWLEDGMENTS

Documentation Team: Mary Leigh Burke, Andrea Ames, Caroline Arakelian, Adam Barnett, Lisa Braz, Mary Kraemer, Elaina Von Haas

Contributors: Farzad Farahbod, David Katcher, Yonghong Pan, Hardeep Singh, Terence Soh, Soumitro Tagore

Table of Contents

Introduction

Chapter 1 Quick Feature Tour of Data Director

Chapter 4 Modeling Your Database

Chapter 5 Customizing Your Model

Chapter 6 Linking Your Data to Visual Controls

Introduction

In This Introduction

This Introduction provides an overview of the information in this manual and describes the conventions it uses.

Why Use Data Director?

Data Director makes developing database applications easier in many ways:

- **You can avoid writing code.** Data Director extends the Visual Basic drag-and-drop paradigm to allow rapid application prototyping. This includes letting you create master-detail relationships without writing code.
- **You can develop applications without knowing complex SQL.** Data Director dynamically generates optimized SQL statements for you, to access and manage your data, so that no knowledge of SQL or the underlying database is required.
- **You can use a GUI or an API.** The entire Data Director interface is accessible to you both graphically and through a programmatic API. You can use either one, or both in combination.
- **You do not need a database connection.** At design time, you do not need an active connection to your database server; Data Director stores the structure of your database schema.
- **You can use ActiveX controls.** If you are using Visual Basic 5.0, Data Director lets you link database columns to COM objects.
- **You can use the Web.** You can create a multitier database application, with a GUI control for navigation, and deploy it on the Web.
- **You can use complex data types.** If your Informix database includes the Universal Data Option, you can create your own data types and routines to extend the database server.

About This Manual

This manual provides you with conceptual and task information to help you maximize your productivity as you use Data Director. Read this manual for a comprehensive overview of how to use the product.

Organization of This Manual

This guide includes the following chapters:

- Chapter 1, "Quick Feature Tour of Data Director," introduces the Data Director components, provides an overview of the Data Director development process, and describes Data Director runtime features.
- Chapter 2, "Designing and Building an Application," gives background and recommendations for designing a Data Director application and an overview of how to use Data Director to implement the design.
- Chapter 3, "Running and Customizing Data Director," describes how to run Data Director and how to set various Data Director options to customize your development environment.
- Chapter 4, "Modeling Your Database," introduces the concept of a Model and describes how to capture your database structure. This chapter also describes how to work with Model elements.
- Chapter 5, "Customizing Your Model," introduces the concept of a DataGroup and describes how to create and manage DataGroups.
- Chapter 6, "Linking Your Data to Visual Controls," introduces the concepts of DataLinks and data paths and describes how to link database columns to Visual Basic controls.
- Chapter 7, "Navigating Through Data," introduces the Data Director Navigation Control and describes how to customize it. This chapter also describes how to work with the properties, objects, and events available to the Navigation Control.
- Chapter 8, "Data Management at Runtime," describes some common strategies for your application behavior at runtime, such as querying data and using data lookup.

- Chapter 9, "Using Data Director Programmatically," introduces and describes the Data Director programmatic interface, which you can use to build your applications either instead of or in addition to the Data Director graphical user interface.
- Chapter 10, "Shared Development," describes how to work with Data Director in a multiple-developer environment.
- Chapter 11, "Troubleshooting Your Application," describes the runtime features of a Data Director application. This chapter also describes error handling and using the trace file.
- Chapter 12, "Deploying Your Application," describes how to deploy a finished Data Director application and includes a list of Data Director files required for distribution.
- Appendix A, "ODBC DataDriver Support," describes the Data Director ODBC DataDriver and includes information about compliance levels and additional software required to use the ODBC DataDriver.
- Appendix B, "Data Director Utilities," describes how to use the Data Director Connect utility to verify your database connection, and how to use the Query Builder utility to create SQL statements.
- The Glossary includes terms specific to Data Director and also to database applications and Web applications.

Types of Users

This manual is written for application developers who want to use Data Director to write client/server applications in Visual Basic. Before using Data Director, you should have the following information about your software environment:

- A basic understanding of Windows 95 or Windows NT, Version 4.0 or higher
- A working knowledge of Visual Basic 4.0 or 5.0
- A basic understanding of the database against which your application runs

Software Dependencies

To use Data Director as described in this manual, you must:

- be running Windows 95 or Windows NT, Version 4.0 or higher.
- have Microsoft Visual Basic 5.0 or 4.0a.
- have access to a database server that supports ODBC.

Contact your ODBC or database vendor for ODBC drivers to connect to non-Informix databases.

Demonstration Database

Data Director includes some demonstration applications, along with a small sample database named **sports**, which contains information about a fictitious sporting-goods distributor. The examples mentioned in this manual are based on the **sports** database.

For information on setting up the demonstration database, see Appendix A of the *Data Director Tutorial*.

Documentation Conventions

This section describes the conventions that this manual uses. These conventions make it easier to gather information from this and other volumes in the documentation set.

The following conventions are covered:

- Typographical conventions
- Icon conventions

Typographical Conventions

This manual uses the following conventions to introduce new terms, illustrate screen displays, describe command syntax, and so forth.

Convention	Meaning
KEYWORD	All primary elements in a programming language statement (keywords) appear in uppercase letters in a serif font.
italics ***italics*** *`italics`*	Within text, new terms and emphasized words appear in italics. Within syntax and code examples, variable values that you are to specify appear in italics.
boldface ***boldface***	Names of program entities (such as classes, events, and tables), environment variables, file and pathnames, and interface elements (such as icons, menu items, and buttons) appear in boldface.
`monospace` *`monospace`*	Information that the product displays and information that you enter appear in a monospace typeface.
KEYSTROKE	Keys that you are to press appear in uppercase letters in a sans serif font.
♦	This symbol indicates the end of product- or platform-specific information.
→	This symbol indicates a menu item. For example, "Choose **Tools→Options**" means choose the **Options** item from the **Tools** menu.

Tip: *When you are instructed to "enter" characters or to "execute" a command, immediately press* RETURN *after the entry. When you are instructed to "type" the text or to "press" other keys, no* RETURN *is required.*

Icon Conventions

The following table describes the product-version icons that appear in this manual.

Icon	Description
VB 4	Identifies information that is specific to Visual Basic 4.0
VB 5	Identifies information that is specific to Visual Basic 5.0

Additional Documentation

Data Director documentation is provided in a variety of formats:

- **Printed documentation.** In addition to this guide, the printed documentation set includes:
  - CD insert, which contains installation and configuration instructions.
  - *INFORMIX-Data Director for Visual Basic Tutorial*, which takes you through the key steps of developing a client/server application using Data Director.
  - *Informix Client Products Installation Guide for Microsoft Windows Environments*, which contains information about Client SDK and **Setnet32**.
- **On-line manuals.** The Informix Answers OnLine CD includes all Informix product documentation, and allows you to print chapters or entire books and perform full-text searches for information in specific books or throughout the documentation set.

  You can also access on-line versions of *Data Director User Guide* and the *Data Director Tutorial* from within Data Director. Choose **Help→Data Director Books Online** from the Visual Basic menu bar, and then select the book that you want to see.

- **On-line reference.** This facility includes reference information about the Data Director Objects programmatic interface, the Data Director controls, the association interfaces for linking to COM objects, and the Java classes. To access this on-line reference information, choose **Help→Data Director API Reference** from the Visual Basic menu bar.

What's This? help button

- **Context-sensitive on-line help.** You can obtain help on user interface elements by using the **What's This?** help button at the top of a dialog box, or by right-clicking an interface element, or by pressing F1 in a dialog box or window.
- **On-line examples.** The on-line examples illustrate using Data Director Objects in an application. The examples are located in the **Examples** directory within the directory where you installed Data Director. There is a subdirectory within the **Examples** directory for each example. See the accompanying text file in each of the **Examples** subdirectories for information about how to use that example.
- **Documentation notes and release notes.** Documentation notes, which contain additions and corrections to the manuals, and release notes are located in the directory where the product is installed.

  Please examine these files because they contain vital information about application and performance issues.

Informix Welcomes Your Comments

Let us know what you like or dislike about our manuals. To help us with future versions of our manuals, we want to know about any corrections or clarifications that you would find useful. Include the following information:

- The name and version of the manual that you are using
- Any comments that you have about the manual
- Your name, address, and phone number

Write to us at the following address:

Informix Corporation
Tools Technical Publications Department
4100 Bohannon Drive
Menlo Park, CA 94025

If you prefer to send electronic mail, our address is:

doc@informix.com

We appreciate your suggestions.

Chapter

1

Quick Feature Tour of Data Director

In This Chapter

This chapter introduces the Data Director database application development software.

Data Director is a powerful tool that adds sophisticated database functionality to your application development environment. With Data Director, you can create high-performance, open database applications with a minimum of coding.

What Does Data Director Add?

Among many other features, Data Director gives you:

- an easy way to connect your database schema to the forms you create in your application development environment.
- automatic visual access to data-management functionality, such as master-detail coordination, which saves you programming time and effort.
- access to the extensibility and power of Informix Dynamic Server with Universal Data Option.

Data Director automates *data access* by querying for data according to query conditions, managing database connections and cursors, providing logon management and logon security, and sorting data.

Data Director facilitates *data management* by automating standard data operations like saving, updating, and data navigation. Data Director also provides master-detail synchronization and data value synchronization across multiple application windows, manages concurrency and transactions, automates data caching, and supports bookmarks.

Data Director automates data management without increasing the code overhead of your application and without causing any degradation of application speed or performance. Any application that you develop with Data Director can include all the data management features mentioned in the preceding list, and more—with no additional code.

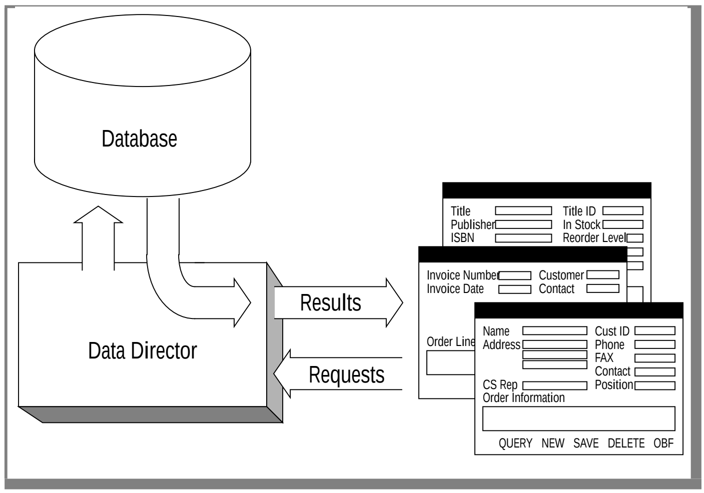


Figure 1-1
Data Director Automates Data Access and Data Management

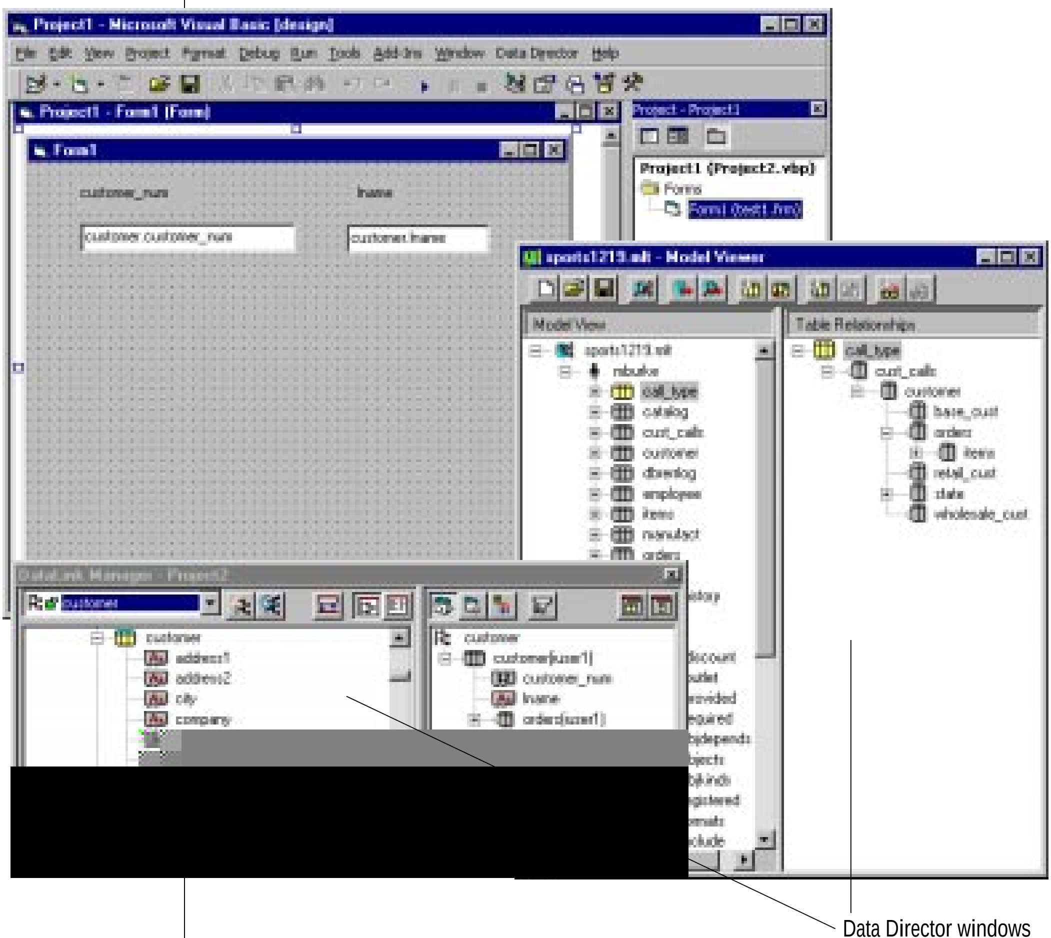
What Does Data Director Look Like?

Data Director is fully integrated with your application development environment:

- Data Director components appear as menu items and toolbar icons within your application development environment.
- Data Director lets you build your application's user interface the same way you would build it using the development environment alone.
- Data Director generates SQL statements for you to access and manage your data. You do not need to understand the complexities of SQL—Data Director transparently manages all communication with your back-end server.

Figure 1-2 shows a Visual Basic application being built with Data Director.

Figure 1-2
Data Director Integrates with Your Visual Basic Environment



To execute a query for the master-detail data in this figure, Data Director first queries the **customer** table. Based on the value of the customer number (the primary key column), Data Director then queries records in the **orders** table for the current customer. Finally, based on the order number (the primary key column for the **orders** table, Data Director queries records in the **items** table for the current order.

Based on the project files in use, Data Director automatically knows that **customer** is the master query table and that a one-to-many relationship exists from the master **customer** table to the detail **orders** table. Data Director recognizes this relationship because the primary key column in the **customer** table is a foreign key in the **orders** table. The same relationship exists between the master **orders** and detail **items** tables: the order number is a foreign key in the **items** table.

Data Director automatically performs master-detail coordination in a running application. You need not write any additional code to implement it.

Data Validation

When your application runs, Data Director checks that data values have been entered for all required columns and that a primary key is present when the user is updating or deleting a record.

You can also supply custom data validation by programming the events for the Navigation Control. For more information, see Chapter 7, "Navigating Through Data."

Support for Complex and Opaque Data Types

If your Informix server includes the Universal Data Option, you can create your own data types and routines to extend the database server and provide greater flexibility in the kinds of data you can store and manipulate. Complex data types are also available in all Informix Dynamic Servers beginning with Version 9.2.

More information about these data types is available in the following manuals:

- For general information about these data types, see the manual *Getting Started with Informix Dynamic Server*.
- For more detailed information about row, distinct, and opaque types, and user-defined routines, see *Extending Informix Dynamic Server*.

Data Director supports the following additional data types and routines:

- **Row types.** A sequence of fields that are all defined under a single name.

  For example, a row type named **address** might contain fields representing a street, city, state, and zip code.

  Row types are supported through virtual tables rather than through the GUI. For more information, see "Creating Virtual Tables" on page 4-29.
- **Distinct types.** A data type that has the same internal structure as an existing data type, but has a distinct name and includes functions that make it different from its source type.

  For example, distinct types named **franc**, **pound**, and **lira** might all be based on the existing MONEY data type.
- **Opaque types.** A fundamental extended data type that stores a single value and cannot be divided into components by the server.

  Opaque types are often used to represent unconventional data, such as geographic locations or colors.
- **User-defined routines.** A function or procedure that you define and register in the system catalog tables, and that you can invoke within an SQL statement or within another routine.

You can use Data Director to access this data in your Visual Basic application. For example, you can query for row types using a virtual table, and you can drag and drop opaque data to standard controls like edit fields and combo boxes and have the text representation appear in the control.

To display opaque data that is best represented in other than text format, you must write or obtain an ActiveX control that is capable of visualizing the data, and you must use the **IDDAssociation** COM interface to link your data to the ActiveX. For more information about this interface, see "Linking Data to COM Objects" on page 9-64.

Data Director supports complex and opaque data types by providing you with the following abilities:

- You can import user-defined routines and opaque data types from your database into your Model.

  For more information, see "Modeling Routines" on page 4-35.
- You can access elements in a row type using SQL statements in a virtual table.

  For an example, see "Querying a Row Type Column" on page 9-41.
- You can use a distinct data type the same way you would use the data type on which it is based.

  For more information about distinct data types, see *Extending Informix Dynamic Server*.
- You can link opaque data types to controls in your application using the DataLink Manager.

  For more information, see "Creating DataLinks for Opaque Data Types" on page 6-11.
- You can create a column with an opaque data type in your Model.

  For more information, see "Working with Opaque Data Types" on page 4-54.
- You can access user-defined routines through the **Routine** object in the DDO.

  For more information, see "Working with Routines" on page 9-48.
- You can set properties for user-defined routines in the Routine Properties dialog box.

  For more information, see "Using Routines" on page 4-33.
- You can access opaque data types through the **Opaque** object in the DDO.

  For more information, see "Using the Opaque Data Type" on page 9-76.

Chapter

2

Designing and Building an Application

In This Chapter

This chapter offers considerations for designing and building an application with Data Director, provides an overview of the steps you must follow to implement the application design, and offers some suggestions for improving application performance through design. Subsequent chapters in this manual go into more depth about each stage of building and deploying your application.

Design Issues to Consider

Before you build your database application with Data Director, you must make some decisions about how the application will be structured:

- Will the application be two-tier or multitier?
- Will the application be entirely visual or will it contain code-based functionality?
- Which databases will the application access?

The rest of this section addresses each of these questions.

Will the Application Be Two-Tier or Multitier?

A typical two-tier Data Director application has the components shown in Figure 2-1 at runtime. (The two tiers are the client computer and the database server.)

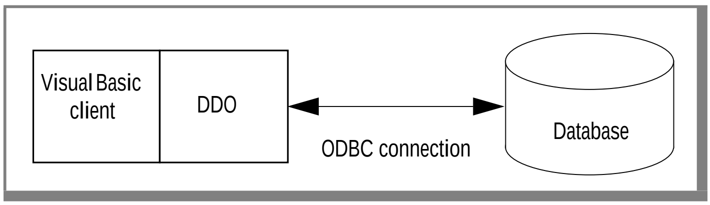


Figure 2-1
Two-Tier Data Director Application

In this figure, the client and the Data Director Objects (DDO) are usually on the same machine, with the database on a separate Informix server. (The DDO interface was introduced in "Using the Data Director Objects" on page 1-10.)

A multitier Data Director application looks like Figure 2-2. (The tiers are the client machine, the middle-tier server, and the database server.)

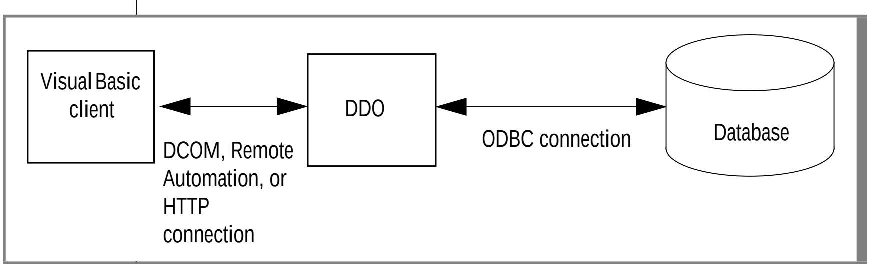


Figure 2-2
Multitier Data Director Application

In this case, the forms and presentation logic are on the client and are written using Visual Basic or an HTML editor, while the Data Director runtime environment and the database logic (written using the DDO) is in the middle-tier machine.

Tip: *The DDO is not itself multithreaded; however, it is thread-safe, which means that the DDO will support multithreaded applications.*

You can also use Data Director in a multitier configuration that includes a Web server. You write the application in VBScript or JavaScript using a simple connectivity API provided by Data Director. Figure 2-3 shows how this works.

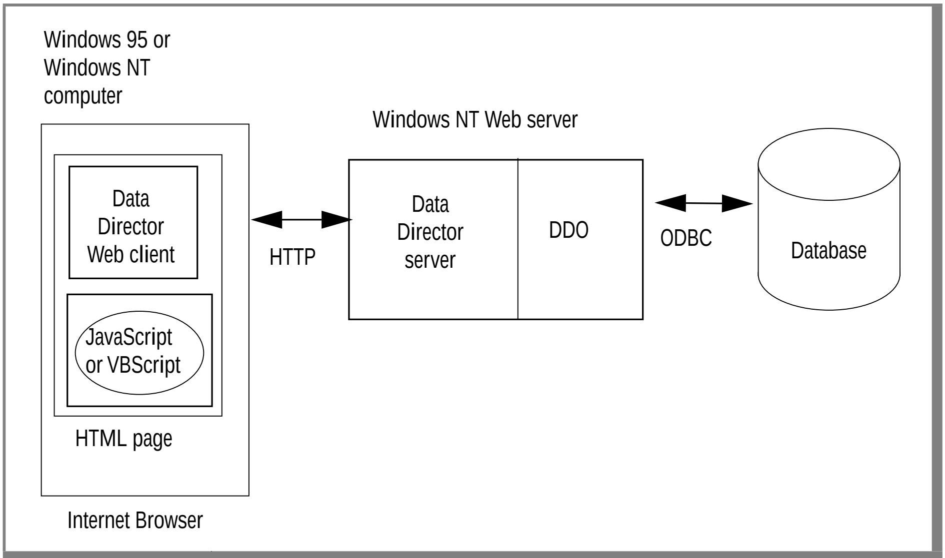


Figure 2-3
Multitier Data Director Application Using a Web Server

See "Deploying Your Application on the Web" on page 12-8 for more information.

Advantages of Multitier Configurations

Data Director lets you create multitier applications. Characteristics that support a multithreaded solution include the following:

- Many clients can call the same middle-tier application.
- If you write the application to create a new DDO **Project** object for each client, each client call causes the server to instantiate a **Project** object, and each new **Project** object can run in its own thread.
- If you use read-only DataGroups, all your clients can share the same database connection.

  Multiple clients can use the same physical database connection to access the database.
- If you use the Web configuration, only the original HTML page is downloaded, while the data retrieved and manipulated changes within the page.

  This reduces network throughput and increases your application's efficiency.

Implications for Deployment

The configuration you choose has implications for the eventual deployment of your finished application, as follows:

- If you use the two-tier configuration, you must create an installation program and package your files into a CD or diskettes. You must then run the installation on each of your client machines.
- If you use the multitier configuration, you only need to install the application once, on the middle-tier machine.
- If you use the Web configuration, you must copy your application's **.ddx**, **.mlt**, and project files to the Web server and set parameters that point to the location of the files from your HTML page. (For details, see "Steps for Deploying on the Web" on page 12-9.)

Will the Application Be Entirely Visual?

Most database access applications are based on data-entry forms. However, sometimes there is functionality that is better achieved outside a form. For example, your application might need to perform batch data entry from an input file by executing an SQL INSERT operation in a loop.

For programmatic database operations that require no user interaction, Data Director provides the *virtual table* interface.

You can use virtual tables to pass custom SQL statements and routines directly through Data Director to your database. For example, you could execute a batch INSERT by creating a virtual table as follows:

```
ddoDataGroup.createVirtualTable(DDOVirtualTable)
ddoVirtualTable.insertCommand = "INSERT INTO employee (
                last_name, first_name, salary) VALUES
                (lname, fname, salary);"
enter loop
    ddoVirtualTable.executeCommand (insertCommand)
exit loop
```

For more information about using virtual tables, see Chapter 9, "Using Data Director Programmatically."

Which Databases Will the Application Access?

For each database your application accesses, you must install and configure an ODBC driver. For a two-tier application, you must install a driver for every client machine; for a multitier application, you must install a driver only for the middle-tier server.

For more information about ODBC drivers, see Appendix A, "ODBC DataDriver Support."

Implementing Your Application Design

Designing an application for implementation with Data Director involves following these steps:

1. Determine what business processes your application will encapsulate.
2. Determine what, if any, of your application functionality should be accomplished without user interaction and what functionality will require forms.
3. Create a Model of your database for your application.
4. Create DataGroups for the business processes you determined in step 1.
5. Design the user interface.
6. Link the data to the forms.
7. Decide what portions of the application to code directly using the DDO.

Determining Business Processes

Business processes represent the way your business works. For example, your business might have one department that handles customer support calls, another group that handles taking and processing orders, and an internal human resources group that tracks employee information.

Your application could represent this system by creating one DataGroup to access data in a **customer** table, another to access information in an **orders** table, and a third to access information in an **employee** table.

Each DataGroup provides you with a different view into your application's data. DataGroups also coordinate data across multiple forms.

If your application contains many business processes, you might consider breaking up the application into smaller, more manageable pieces. It is unlikely that a user will need to access all the application functionality at the same time, and each smaller application is less resource-intensive. Additionally, if one of the business processes changes, you do not have to propagate the change as extensively, making maintenance easier.

Determining Form-Based Functionality

Most of your application will probably be based on forms and involve user interaction. You should plan up front for any functionality that will not require user interaction.

For example, your application might need to perform some batch processing between interactions with a user.

Creating a Model

Once you understand your business processes, you can create a Model to represent your data. Data Director provides you with a Model Import Wizard that lets you model each set of tables that your application will access. You can import any existing database to a Model. You can also create a Model and design your application without a database, and connect to a database at runtime.

Although it is possible to create a Model that depicts the entire database, it is more efficient to include only the tables your application will need to use. This reduces Model load time, data path resolution time, and application memory use.

Models are the starting point for any Data Director application, containing all the database structure information used by Data Director to implement data access and data management. Model concepts are discussed in greater detail in Chapter 1, "Quick Feature Tour of Data Director." Model usage is discussed in Chapter 4, "Modeling Your Database."

Creating DataGroups for Business Processes

Once you create Models, you can create DataGroups to organize the data in your Models and represent your business processes.

You create DataGroups using the DataLink Manager. When you create a DataGroup, you assign a Model and a master table from the Model to the DataGroup. The master table tells Data Director which table in your database to start queries from when querying data.

DataGroups also contain additional information about your application, such as table-sorting and query conditions, concurrency control settings, and error-handling options.

Because each DataGroup uses a separate database connection (unless it is a read-only DataGroup), you should create the minimum number of DataGroups your application needs for maximum efficiency. DataGroups are described in greater detail in Chapter 5, "Customizing Your Model."

Designing the User Interface

When you design your user interface, you represent some portion of your Model and implement the main business process that your application will manage. The following windows are typical examples:

- Order entry window
- Customer information window
- Employee information window

In addition to creating the user interface, you must also pay attention to any functionality in the application that will need to occur behind the scenes (for example, intermediate calculations using a virtual table), and decide at what points during the user's interaction with the application these operations will need to take place.

Linking Data to Forms

After you create DataGroups, you create DataLinks to connect your database columns to visual controls in your forms. A DataLink coordinates data between a database and your application and tells Data Director where data should be displayed in your application. More information about DataLinks is in Chapter 6, "Linking Your Data to Visual Controls."

Information about testing and deploying your finished application is in Chapter 11, "Troubleshooting Your Application."

Deciding Where to Use the Data Director Objects

The drag-and-drop feature of Data Director makes creating an application easy. However, using the DDO allows you more flexibility and greater control over your application's behavior at runtime.

Sometimes, using the DDO can also reduce the size of application files. For example, if a form within your application is used only rarely, you might choose to create DataLinks to the form using the DDO instead of the DataLink Manager and call the DDO code only when the form is loaded. This keeps the size of the DataLink (**.ddx**) file small and increases application efficiency.

For more information about the DDO and reasons to use it, see Chapter 9, "Using Data Director Programmatically."

Designing for Performance

This section covers some design steps you can take to maximize the performance of Data Director when your database is very large or when your application is highly interactive.

Using Parameterized Detail Fetch

When two tables in your application are in a master-detail relationship, Data Director generates a new query and fetches the detail table data whenever the user navigates within the master table.

You can avoid this situation and reduce the time spent querying the database by enabling *parameterized detail fetch*, a DataGroup property you can set at development time. When you enable parameterized detail fetch, the query that populates the detail table is generated once, with parameter place-holders in the WHERE clause, and then the query is re-executed with different values for each qualifying foreign key value.

Tip: *You can enable this property only if your ODBC database driver supports parameters. For more information about parameters and how to determine if you can use them, see “Parameters in the DDO” on page 9-17.*

Minimizing Form Load Time

A *reference link* is a type of DataLink that Data Director uses to query *lookup data*. Lookup data refers to data that appears in a combo box—such as lists of employee names. The data being looked up is called a *reference group*.

For example, suppose you are building an application window for entering new order information. To display a comprehensive list of employee names that you want to assign as a customer service representative for each new order, you would query and display all values from the **name** column of the **employee** table in a **CS Rep** combo box. Users can then choose an employee name for their customer service representative from the list.

Data Director queries all reference links on a form at startup. To avoid delays in loading forms that contain reference groups with many records (for example, if there are hundreds of employees in the **employee** table), you can do the following:

- Delay the reference group query until the point at which the user invokes the **DropDown** event for the combo box.
- Set the **ddoTable.MaxRows** property to limit the size of the result set in the combo box. For example:

  ```
  ddoTable.MaxRows = 15
  ```

Chapter

Running and Customizing Data Director

3

In This Chapter

This chapter discusses starting Data Director, and it covers the various options you can set to customize your environment. For information about installation and configuration, see the installation instructions that come with your Data Director CD.

Starting Data Director

Data Director integrates into your Visual Basic environment seamlessly. When you first start Visual Basic, you see all the standard elements, with additional windows and menu items for Data Director.

Launching the Startup Wizard

Whenever you start a new Visual Basic project, Data Director automatically invokes the Startup Wizard. The Startup Wizard guides you through the following basic startup tasks:

- Importing a Model
- Creating a DataGroup

You can specify skipping the Startup Wizard for future projects, either by checking the **Skip this wizard in future** check box in the Startup Wizard, or by unchecking the **Show the Startup Wizard when a new project is created** check box on **Environment** page of the DataLink Manager Options dialog box.

If you choose, you can run the Form Wizard immediately on terminating the Startup Wizard (see the next section).

Launching the Form Wizard

You can start the Form Wizard from the last page of the Startup Wizard, or by choosing **View→Form Wizard** from the Visual Basic menu bar. The Form Wizard guides you through the following tasks:

- Selecting tables and columns from the Model to appear on the form
- Configuring a Navigation Control for the form
- Naming the form

You can specify skipping the Form Wizard for future projects, either by checking the **Skip this wizard in future** check box, or by unchecking the **Show the Form Wizard start page** check box on the **Environment** page of the DataLink Manager Options dialog box.

Displaying the DataLink Manager

The DataLink Manager lets you link columns from your database tables to controls on your form, while the Model Viewer lets you create and use a Model of your database schema.

If you selected **Launch Data Director with Visual Basic** during Data Director installation, launching Visual Basic automatically starts the DataLink Manager. If the DataLink Manager does not start, open the Visual Basic Add-In Manager, and ensure that the **Data Director for Visual Basic** check box is checked.

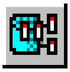
The **Data Director** menu appears in the Visual Basic menu bar. You will also find additional Data Director menu items in many of the Visual Basic menus. In addition to all the familiar Visual Basic components, the DataLink Manager appears below the Visual Basic controls, as was pictured in Figure 1-2 on page 1-5.

Displaying the Model Viewer

You can launch the Model Viewer from the menu bar in your development environment or from the DataLink Manager.

To display the Model Viewer from the menu bar

1. Choose **View→Model Viewer** from the menu bar.



Model Viewer button

To display the Model Viewer from the DataLink Manager

1. Click the **Model Viewer** button in the lower-left corner of the DataLink Manager.

Customizing Data Director

Before going further, you probably want to customize your environment. You can customize your Data Director environment by setting DataLink Manager options, Model options, filtering options, and runtime options.

Customizing the DataLink Manager

There are several characteristics of the DataLink Manager environment that you can customize. For example, you can specify how you want to display items in the DataLink Manager.

If you modify the default settings for the DataLink Manager, your changes take effect when you close the DataLink Manager Options dialog box, and they persist until the next time you modify them.

To customize the DataLink Manager

1. Choose **Data Director→Options→DataLink Manager** from the Visual Basic menu bar.
2. Use the tabbed pages in the DataLink Manager Options dialog box to customize your environment.

   The settings that you can modify are described in the sections that follow.

Setting Autosize Options

Autosize options let you specify when Data Director automatically resizes the DataLink Manager. Autosize options are useful when you need to maximize your available screen space when designing an application.

The **Autosize** page is shown in Figure 3-1.

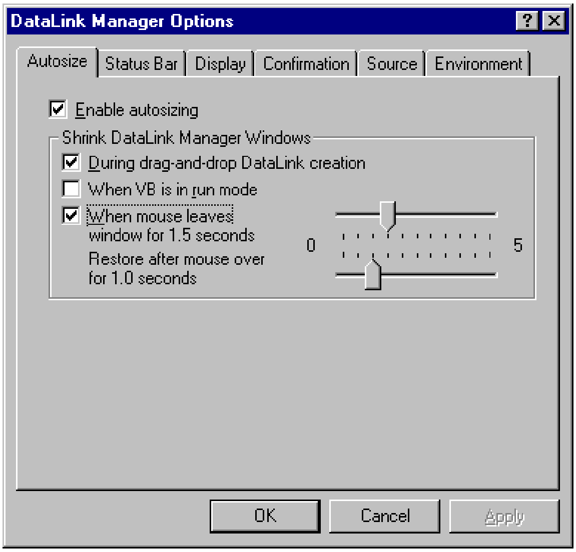


Figure 3-1
Autosize Page

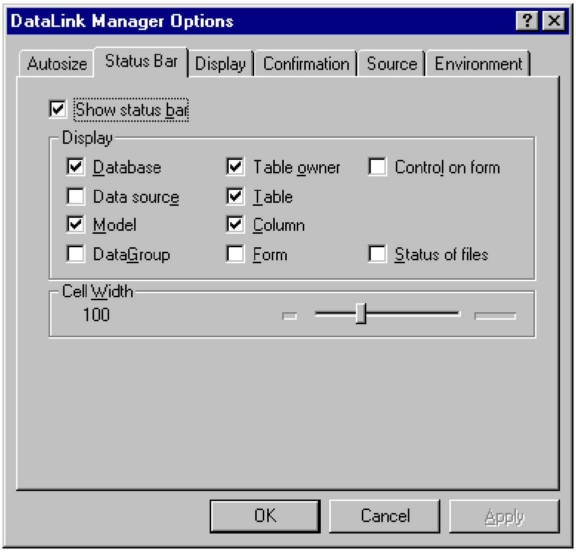
For a description of each option, use Microsoft's What's This? help.

Setting Status Bar Options

The status bar includes information about the items in the DataLink Manager window, as well as additional buttons that let you launch the Model Viewer, open the trace file for a project, or launch on-line help. Status bar options let you display the status bar in the DataLink Manager window and specify which items should appear.

The **Status Bar** page is shown in Figure 3-2.

Figure 3-2
Status Bar Page



For a description of each option, use Microsoft's What's This? help.

Setting Display Options

Display options let you configure the information that Data Director displays in the DataLink Manager.

In the **Sort Model Tree By** group, you can choose to sort the items that appear by object type. For example, you can sort all the items by tables, then views, and then routines.

The **Display** page is shown in Figure 3-3.

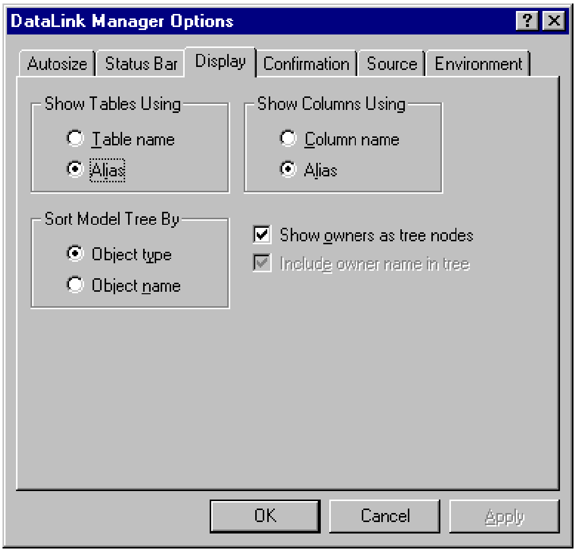


Figure 3-3
Display Page

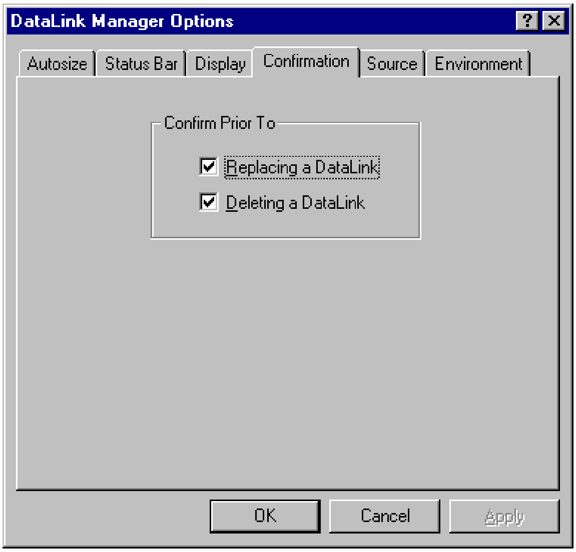
For a description of each option, use Microsoft's What's This? help.

Setting Confirmation Options

Confirmation options let you confirm the replacement or deletion of a DataLink at design time.

The **Confirmation** page is shown in Figure 3-4.

Figure 3-4
Confirmation Page



For a description of each option, use Microsoft's What's This? help.

Setting Source Options

Source options enable you to configure how Data Director works with your source code control system.

The **Source** page is shown in Figure 3-5.

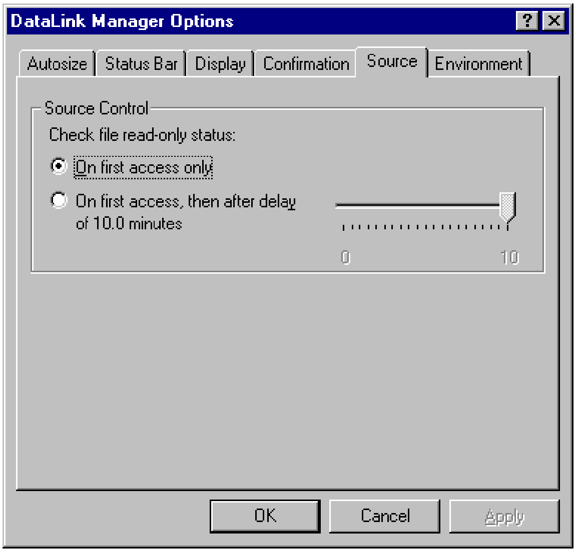


Figure 3-5
Source Page

For a description of each option, use Microsoft's What's This? help.

Setting DataGroup Filtering Options

DataGroup filtering options let you customize your view of the information that appears in the DataGroup pane of the DataLink Manager. For example, you can display only tables that are linked to controls on the current form.

To set DataGroup filtering options

Filtering Options button

1. Click the **Filtering Options** button in the DataGroup pane.
2. Click the filtering options you want to use.

The DataGroup Tree Filter Options dialog box is shown in Figure 3-6.

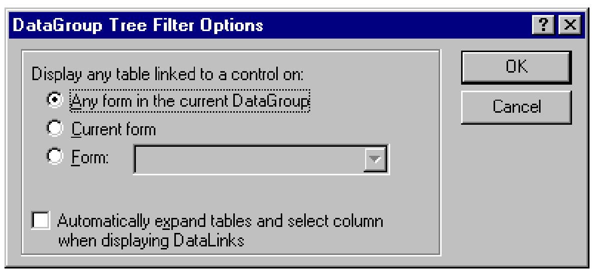


Figure 3-6
DataGroup Tree Filter Options Dialog Box

To quickly sift through large amounts of information in the DataGroup pane, check **Automatically expand tables and select column when displaying DataLinks**.

For a description of each option, use Microsoft's What's This? help.

Setting DataLink Filtering Options

DataLink filtering options let you customize your view of the information that appears in the DataLink pane of the DataLink Manager. For example, you can display only DataLinks that are linked from a specific table to the current form.

To set DataLink filtering options

Filtering button

1. Click the **Filtering** button in the DataLink pane.
2. Click the filtering options you want to use.

The DataLink List Filter Options dialog box is shown in Figure 3-7.

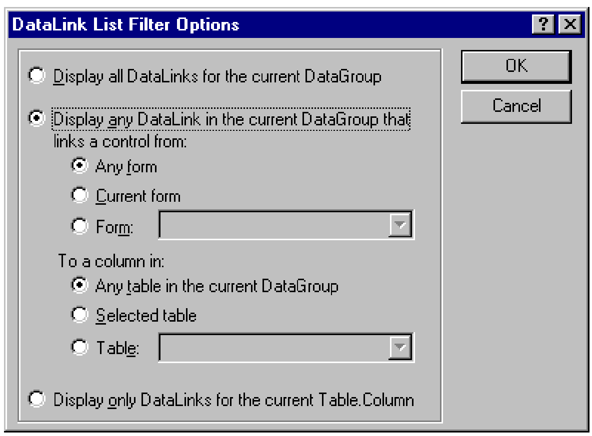


Figure 3-7
DataLink List Filter Options Dialog Box

The **Display only DataLinks for the current Table.Column** option is useful if two DataLinks to the same control have paths to two different tables (for example, a grid control can have DataLinks to more than one table). Using this option, you can quickly determine the different paths associated with the two DataLinks. The currently selected table and column are the table and column selected in the DataGroup pane.

For a description of each option, use Microsoft's What's This? help.

Setting Environment Options

The **Environment** page (shown in Figure 3-8) lets you set some additional DataLink Manager options.

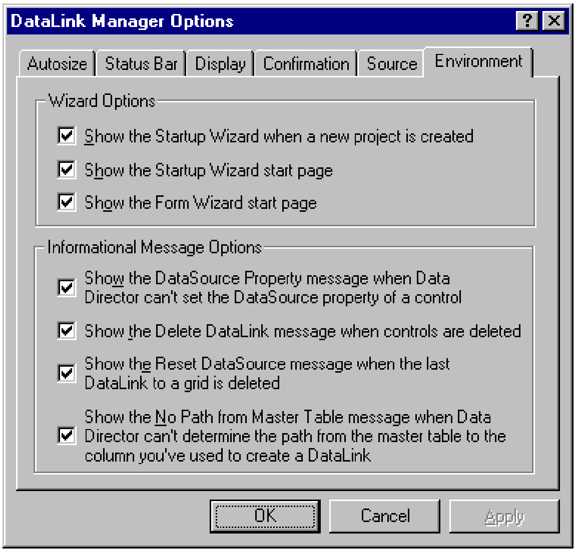


Figure 3-8
Environment Page

For a description of each option, use Microsoft's What's This? help.

Customizing the Model Viewer

There are several characteristics of Models that you can customize. For example, you can specify whether you want to be prompted to confirm the deletion of Model elements.

If you modify the default settings for a Model, your changes take effect on closing the Model Options dialog box and persist until the next time you modify them.

To customize the Model Viewer

1. Choose **Data Director→Options→Model** from the Visual Basic menu bar.

   A Model must be open in the Model Viewer. If it is not, you cannot choose the **Model** menu item.

2. On the **Display** page of the Model Options dialog box, click the options you want to use.

Tip: *For information about setting import options before capturing a database structure, see* *"Setting Import Options" on page 4-13**.*

Display options let you configure the information that Data Director displays in the Model Viewer window.

The **Display** page is shown in Figure 3-9.

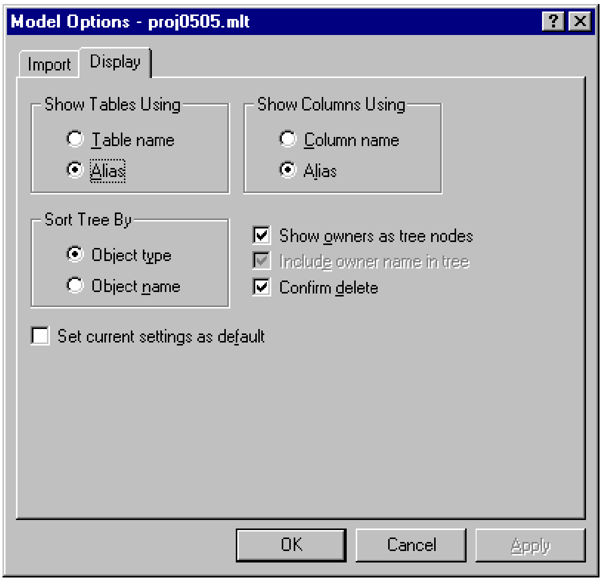


Figure 3-9
Display Page

The **Sort Tree By** options sort items in the Model Viewer, either by object name or by object type. Sorting by object type displays the items sorted by table, then view, and then routine. Sorting by object name displays the items sorted in alphabetical order, according to the name of the object.

For a description of each option, use Microsoft's What's This? help.

Setting Project Options

This section describes how to enable the automatic creation of a trace file for your project. It also describes how to enable shared connections to the database.

Setting the Trace File Option

Trace files help you debug your application, and they let you view the SQL statements Data Director generates. You can enable the trace file functionality and specify additional information for a trace file (see "Using the Trace File to Debug Your Application" on page 11-7).

The **Trace** page is shown in Figure 3-10.

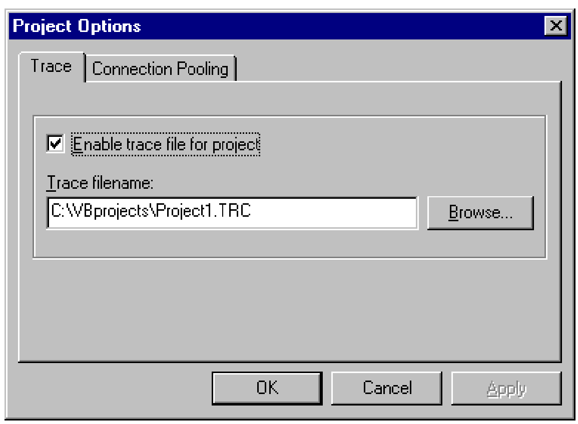


Figure 3-10
Trace Page

To set trace file options

1. Choose **Data Director→Options→Project** from the Visual Basic menu bar.
2. Check the **Enable trace file for project** check box and supply a filename for the trace file.

If you do not specify a filename (including path), Data Director provides a default name based on the corresponding project name. For example, a project named **sports.vbp** has an associated trace file named **sports.trc** by default. If you do not specify a file location, Data Director saves the trace file in your **temp** directory as specified by your **TEMP** environment variable in your **autoexec.bat** file.

If you modify the default settings, your changes take effect on closing the Project Options dialog box and persist until the next time you modify them.

Warning: *When specifying the name of the trace file and its location, consider that if a project and its log file are stored in the same directory, multiple users might unintentionally overwrite one another's trace file entries.*

Setting Connection Pooling Options

Connection pooling lets a read-only DataGroup share its connection to the database with all other read-only DataGroups in the application. This scenario typically occurs in Internet commerce applications that have hundreds or even thousands of concurrent users. You can set the number of connections to any number.

The **Connection Pooling** page is shown in Figure 3-11.

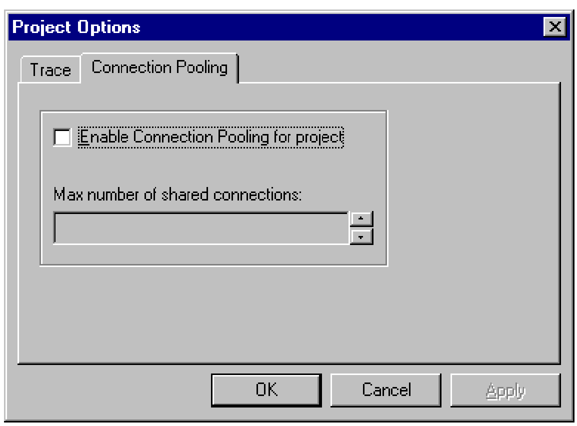


Figure 3-11
Connection Pooling Page

Once you check **Enable Connection Pooling for project**, you can then set the maximum number of shared connections to any integer.

To make a DataGroup read-only, you can set a property on the **Connection** page in the DataGroup Properties dialog box. See "Setting Connection Properties" on page 5-24 for more information.

Chapter

Modeling Your Database

4

In This Chapter

To develop a relational database application, you must understand the basic elements of a relational database and know how to manipulate them:

- Tables and views
- Relationships between tables
- Primary and foreign keys
- Stored procedures
- SQL statements

This chapter shows how you can use the Data Director Model to simplify working with these elements. The Model is the first step to rapid graphical application development.

Model Basics

A Model is a "snapshot" of the physical structure of a database and is made up of elements—tables, columns, and relationships—that describe a database structure in entity-relationship terms. A Model tells Data Director about the structure of the data your application will access and facilitates most of Data Director's design-time and runtime functionality.

You can create a Model before you have a database or a connection to a database. For more information, see "Developing Applications Without an Active Connection" on page 4-10.

What Happens to Your Data?

It is important to remember that the schema in your database is separate from the representation of the schema in the Model:

- When you change the Model, there is no effect on the underlying database.
- When you change the database, the Model does not automatically change until you refresh it.

The Relationship Between the Model and the Data Source

The data source enables the driver software to find the database server. A *data source* is a name that combines references to a database-management software product, the remote operating system on which the software resides, and any network information necessary to access the software.

You must set up a data source before you can import a database into a Model. You set up a data source through **Settings** in the Windows Control Panel. For information on how to set up a data source, as well as other information about connecting to a server, see the installation instructions that come with your Data Director CD.

You can change a Model's data source in the Select Model Data Source dialog box. See "Setting a DBMS" on page 4-24.

A Model corresponds to a single data source, and thus to a single database. One Model cannot span data sources or databases.

How Data Director Locates the Model File

Data Director looks for a Model file in the development environment using the following information, in this order:

1. The hard-coded path of the Navigation Control's **ModelFile** property, as set in the Visual Basic Properties dialog box

   If there is no Navigation Control, the **.ddx** file stores relative paths from the directory where the **.ddx** file is located.
2. The directory containing the Data Director project (**.ddx**) file
3. The current directory

If you change the location of a compiled Model (**.mlx**) file, you might see an error when you run your project, stating that Data Director cannot locate your Model file. If you see this error message, either update the Navigation Control's **ModelFile** property in the Visual Basic Properties dialog box, or move the compiled Model (**.mlx**) file to the directory containing the Data Director project (**.ddx**) file.

Tables, Views, and Virtual Tables

A table contains information about specific elements in a database. For example, a **customer** table would contain information about customers, such as customer names and addresses.

The Model displays the names of some or all of the tables in your database, depending on which tables you select during the Model import process.

You can work with tables, views, and virtual tables in Data Director:

- *Tables* are the standard SQL database tables.

  The data in table columns appears within fields in the windows of your application. For example, the information in a **Customer Name** field would come from the **lname** and **fname** columns of the **customer** table.

  A table can contain standard columns or calculated columns. In addition, you can specify an alias for a column, and you can modify the SQL text that is derived for calculated columns.
- *Views* are the standard SQL database views.
- *Virtual tables* are constructed by Data Director based on user-defined SQL statements, routines, or a combination of both.

  You can use virtual tables to pass custom SQL statements and routines directly through Data Director to your database. Setting SQL statements in a virtual table allows you to customize the commands that Data Director executes. For example, when a user deletes a record, Data Director automatically executes the DELETE statement associated with the virtual table. A customized routine associated with the virtual table could then run in place of the DELETE statement when Data Director attempts to delete a record.

  For more information about using a routine with a virtual table, see "Executing a Routine with Parameters" on page 9-49.

Primary Keys and Foreign Keys

A primary key column uniquely identifies each record stored in a table. A foreign key column is a column in a table that is related to a primary key in a different table.

For example, if the **orders** table has a primary key column of **order_num** and the **items** table has a related foreign key column **order_num**, the **order_num** column in the **orders** table is used to uniquely identify orders, while the **order_num** column in the **items** table is used to identify to which order a particular item belongs.

At runtime, Data Director uses the value of the primary key column to perform a lookup query on the related table. The result set then includes records in the related table if the value in the foreign key column matches that in the primary key column in the first table.

Table Relationships

Data Director uses relationships between tables to represent the organization of data in a database and to find associated information stored in your database.

The relationship between two tables is determined by the relationship between specific columns in each of the tables. A relationship starts at the *origin* column and ends at the *destination* column. For example, the relationship between an **orders** table and an **items** table is that for every order, there can be one or more items. In this example, the **orders** table contains the origin column and the **items** table contains the destination column.

Relationships between tables affect the runtime behavior of Data Director. For example, relationships determine how Data Director coordinates data displayed in master-detail records and lookup data fields.

Data Director supports the following relationship types:

- One-to-one
- One-to-many
- Many-to-one

Data Director treats one-to-one and many-to-one relationships as equivalent for the purposes of generating SQL statements (both involve creating a join).

You can see the relationships for any of the tables in your Model by right-clicking the table name in the Model View pane of the Model Viewer. The relationships are displayed to the right of the Model, in the Table Relationships pane.

One-to-Many Relationships

In a one-to-many relationship, one record in the origin table corresponds to multiple records in the destination table. For example, one relationship between an **orders** table and an **items** table specifies that each order can contain one or more items.

Figure 4-1 shows this relationship.

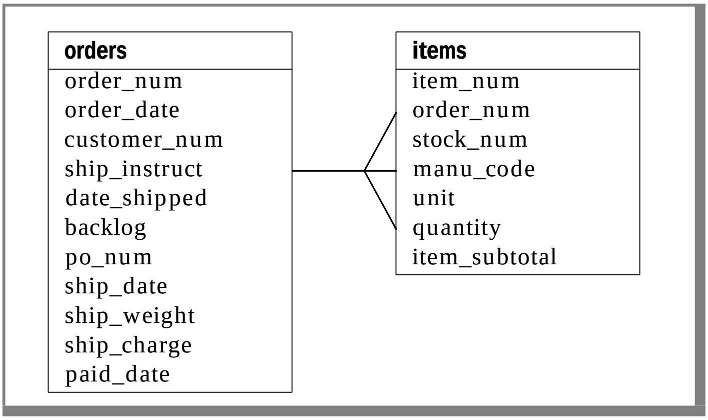


Figure 4-1
A One-to-Many Relationship

Many-to-One Relationships

In a many-to-one relationship, one record in the destination table corresponds to many records in the origin table. For example, multiple orders can be associated with the same customer.

Figure 4-2 shows this relationship.

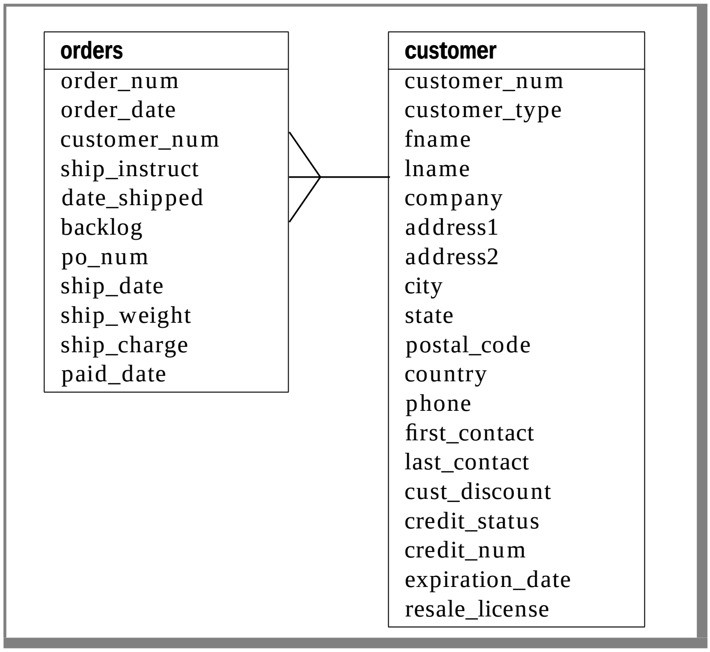


Figure 4-2
A Many-to-One Relationship

Implied Relationships

Relationships are bidirectional. That is, when you create a relationship, an implied relationship exists in the opposite direction (the origin and destination columns switch places). Data Director automatically creates the implied relationship for you.

For example, if you create a relationship in which the **customer** table is the origin table (**customer.customer_num**) and the **orders** table is the destination table (**orders.customer_num**), Data Director automatically creates the implied relationship in which the **orders** table is the origin table (**orders.customer_num**) and the **customer** table is the destination table (**customer.customer_num**).

Using a Single Model Versus Using Multiple Models

Generally, you import a subset of your database structure when you plan to develop a single application, and you import the entire structure of your database when you plan to share the Model with other developers to develop multiple applications within a work group.

You can also use multiple Models in your application if you use databases from different vendors. For example, you can use databases from both Informix and Microsoft to develop an application.

Using a Model to Develop a Single Application

If you are developing a single application, you probably do not need to work with the entire structure of your database. In this case, you can import a subset of your database scheme, using only the specific tables that are relevant to the application you are designing.

For example, if you are developing an accounts payable application, you can import tables related to vendors and employees so that end users can enter invoices and employee expense reports. You do not need to import tables related to inventory items because these tables have no relevance to an accounts payable application.

Following are some of the advantages of working with a single Model:

- The Model is easily maintainable.
- The Model contains only those elements crucial to your development effort.

Sharing a Model Across Applications

If you plan to use a Model for a development project that involves multiple applications, you probably want to work with the entire contents of your database. In this case, import the entire structure of your database into a single Model. You can then share the Model across a work group or development team.

For example, if you are developing a complex suite of financial applications, many of the tables in your database will be shared between different applications. Customers are entered into a Customers application, orders are entered for customers when they purchase products, invoices are generated from a Receivables application to send to customers that ordered products, and finally, account transactions are posted to a General Ledger application to maintain account reconciliations.

While you might not use all of the tables in a Model to develop each application, many of the tables in the Model are shared across applications. For example, the table that contains customer information can be shared between the Customers application and the Orders application.

Although you might end up with a large Model when you import your entire database structure to one Model, there are a number of benefits to sharing a Model across applications:

- The Model is easily shared and maintainable across a work group.
- One change to the Model propagates to all users of the Model.

Developing Applications Without an Active Connection

The Model allows you to work on your application without a database connection. You can use the Model Viewer to create tables, columns, and foreign keys, which then exist only in the Model. Later, when you connect to a database, you can refresh the Model from the database, and Data Director resolves any differences between the database and the Model for you.

You might also use a small, local database (such as a Microsoft Access database) to develop an initial version of your application, and then switch to your production database for deployment. You can save scarce connection resources and develop your application more efficiently without worrying about the availability of the underlying database.

Sharing Models Between Development Teams

You can easily share a single Model across development teams and projects for the simultaneous development of multiple applications. Sharing a Model enables you to easily update and maintain information about your database in a central location—one change to a Model propagates across work groups and projects and is visible to all users of the Model.

Because Data Director stores Model information in a text Model (**.mlt**) file, you can save the shared Model file in a network directory, and thereby make the file accessible to any developer who has access to the directory and file.

When sharing Models, keep the following considerations in mind:

- **Source code control system.** If you use a source code control system, you can check out Model files and make changes to them. When you check in the file, your changes are merged into the development environment. For more information about using a source code control system, see Chapter 10, “Shared Development.”
- **Changes to shared Models.** Any saved changes to the Model affect all projects that use the Model. If you plan to modify a shared Model, you should ensure that any changes you make will not adversely affect the work of other developers.

  For example, you might delete a table in the Model because you no longer have a use for the table. However, if another developer has a project that uses the deleted table, that developer will experience problems. If another developer uses a DataGroup that contains the deleted table and has DataLinks based on the deleted table, all of the DataLinks to the deleted table will be broken.
- **Local copies of Models.** If you are going to make significant modifications to a Model file that is specific only to your project, make a local copy of that Model and then make the required modifications.

  If you make local copies of a Model, ensure that your changes are integrated into the Model that you ship with the deployed application, or ensure that the changed Model file is shipped with the deployed application (see Chapter 12, “Deploying Your Application”).

Importing Your Database Schema

You can import most databases to a Model. The Data Director database drivers automate connectivity to many commercially available databases. This section includes information on importing your database schema to a Model.

Preparing to Import Your Database Schema

Before you can import a Model of your database, you must be able to connect to the database. If your database resides on a remote server, you will first need to set up a data source, to allow the driver to find the database. For details about setting up a data source, see the installation instructions that come with your Data Director CD.

Steps for Importing Your Database Schema

Here is an overview of the steps required to import your database schema into a Model (details of each step appear in the sections that follow):

1. Decide which tables to import.
2. Launch the Model Import Wizard.
3. Set import options.
4. Specify a database driver and data source.
5. Log on to the database.
6. Select items to import.

Deciding Which Items to Import

Before you capture your database structure, you should determine which tables and routines you want to import. You can import a subset of the items in your database, or you can import the entire structure of your database. You can also import database objects that you own or all objects to which you have access.

Generally, a database administrator creates a database and grants privileges to users of the database.

Important: *When you import a database to a Model, you must ensure that you have read privileges to all the items you want to import.*

Launching the Model Import Wizard

Data Director provides you with a Model Import Wizard that enables you to easily capture your database structure. When you run the Model Import Wizard, it guides you through each step of the import process.

Tip: *Importing a database to a Model does not modify the database.*

To launch the Model Import Wizard, choose **Data Director→Model→Import** from the Visual Basic menu bar.

The Model Import Wizard is shown in Figure 4-3.

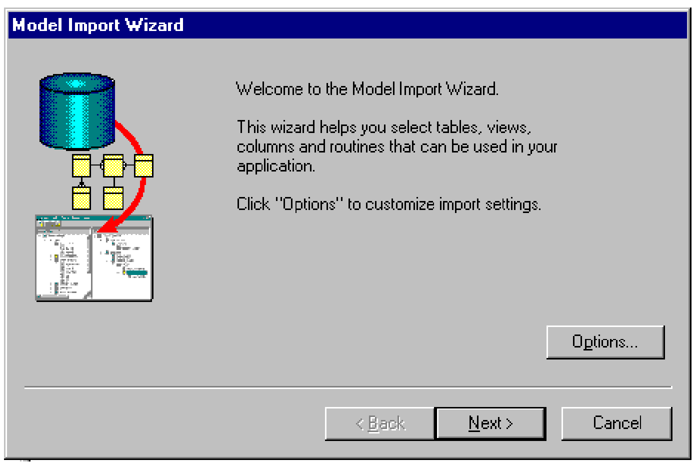


Figure 4-3
Model Import Wizard

Setting Import Options

Click **Options** on the first page of the Model Import Wizard to review and set import options. Data Director provides you with import options that help facilitate the import process, as shown in Figure 4-4. Before you import a database, you should review the options and update them for the database that you are importing. Default options are set after you import a database for the first time. For descriptions of specific options, use the What's This? help.

***Tip:** If you want to import routines as well as tables, you must first check the **Routines** and (for Informix databases only) the **Include system objects** check boxes in the Import Options dialog box.*

Import options persist for the current Model. You can also save preference settings to use as default settings for any additional databases that you import. When you import a database, Data Director reads the database system catalog to determine the structure of your database.

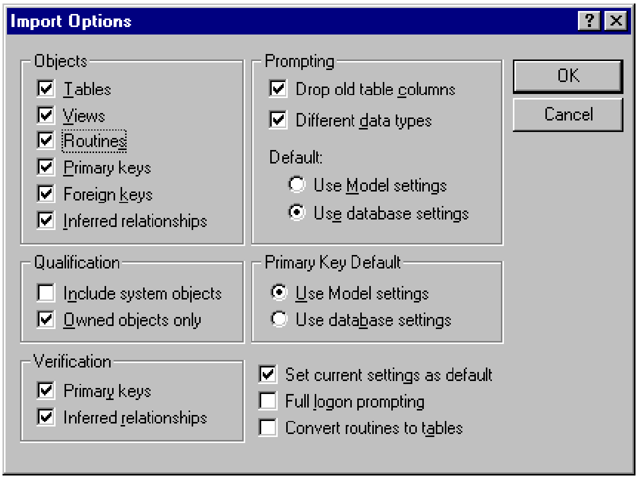


Figure 4-4
Import Options Dialog Box

For more information about the Import Options dialog box, use the What's This? help.

Selecting a Database Driver and Data Source

The Model Import Wizard prompts you to select the database driver and data source that you want to import, as shown in Figure 4-5.

Once you import a database to a Model, you cannot change the type of database that the Model is associated with using the Model Import Wizard.

Tip: *To change the target database that a Model is associated with, set the DBMS after you import a database to a Model (see “Setting a DBMS” on page 4-24).*

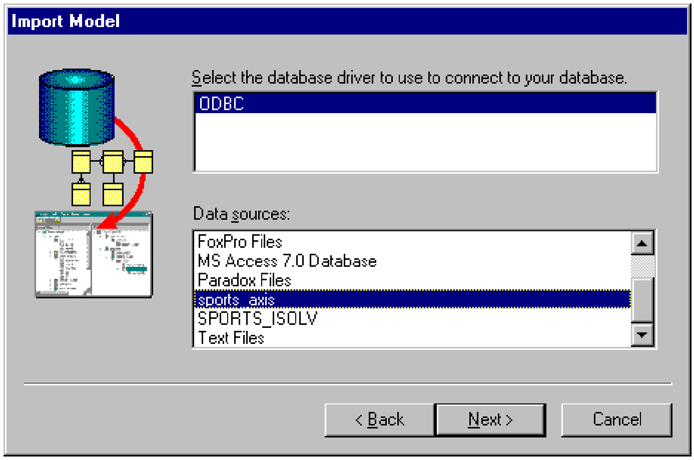


Figure 4-5
Database Driver and Data Source Page

Logging On to Your Database

After you choose a data source, you must specify which database to import and provide a logon ID and password for the database.

If you already associated a database name, logon ID, and password for the data source when you set up the data source, and did not specify **Full logon prompting** in the Import Options dialog box (see “Setting Import Options” on page 4-13), Data Director does not prompt you for logon information. Otherwise, the Logon dialog box appears, as shown in Figure 4-6.

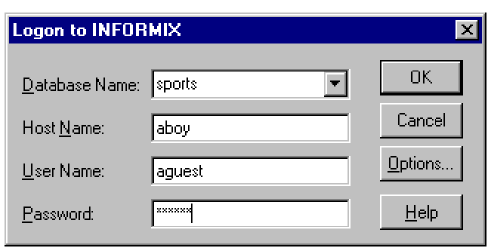


Figure 4-6
Logon Dialog Box

After you enter logon information for a database, the Model Import Wizard prompts you to choose the database that you want to import to a Model.

Database logon information such as user name, database, and server, which is used when importing a Model, is stored in the Model file. Additionally, some DBMS-specific information might be contained in the ODBC connect string stored in the Model file.

Data Director re-uses this logon information for convenience at runtime. If a Model file created against one DBMS is used by an application running against another DBMS, application users might be prompted for logon information, because they are logging on with default values that were appropriate for the first DBMS but not for the current one.

For security reasons, passwords are not stored in the Model file. Application users will typically be prompted for a password if the database requires one. During development, you will be prompted for a password if the **Full logon prompting** check box is checked in the Import Options dialog box.

Selecting Import Items

During the import process, Data Director logs on to your database to provide you with a live view of the database elements available for import. You can easily select the tables and views that you want to import into your Model.

Select the database items you want to import to a Model by checking the boxes next to the items that you want to import, as shown in Figure 4-7. If you do not want a database item in your Model, uncheck the check box next to the item name.

When you click a “+” check box, Data Director queries your database to display the detail items for the table. If there are no detail items for the table, the “+” check box disappears when you click it.

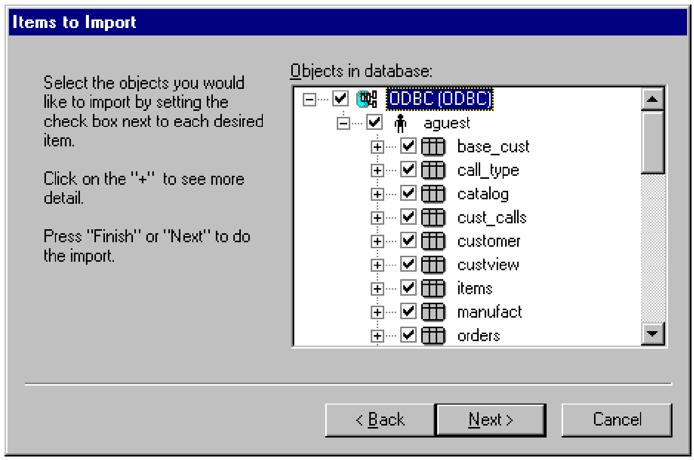


Figure 4-7
Items to Import Page

Additional Operations in the Import Process

During the import process, Data Director might also perform the following operations:

- Updating columns and column attributes
- Verifying primary key columns
- Verifying inferred relationships

This section gives further details about each of these operations.

Updating Columns and Column Attributes

When you refresh an existing Model, Data Director compares the imported database structure with the elements in the Model.

If your Model contains columns that conflict with those specified in your database, the Model Import Wizard prompts you to remove any columns in the Model that are not consistent with those in the underlying database. (This prompt appears only if you set the **Drop old table columns** import option.)

Similarly, if your Model contains column attributes that conflict with those specified in the database, the Model Import Wizard prompts you to update the columns and column attributes. You can use either the database column attributes or the Model column attributes to update the Model.

Verifying Primary Key Columns

Data Director uses primary key information when updating and deleting rows of data. Relationship types are also determined by primary key information.

If Data Director encounters any tables in the underlying database that are missing primary key columns, it prompts you to select primary key columns for those tables, as shown in Figure 4-8.

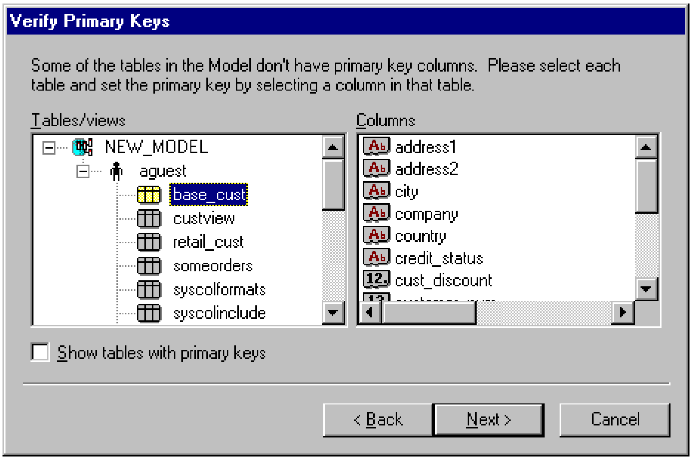


Figure 4-8
Verify Primary Keys Page

Tip: *If you choose to skip the primary key selection for a particular table, you can specify a primary key for a column at a later time using the Primary Keys dialog box (see "Setting Primary Keys" on page 4-26).*

When you refresh an existing Model, if the Model contains primary key information that conflicts with primary key information specified in the database, Data Director uses the primary key default that is set in the Import Options dialog box.

Important: *If you specify primary keys in the Model but not in the underlying database, the Model primary keys are automatically used when you refresh a Model.*

Verifying Inferred Relationships

When Data Director encounters a primary key column, it searches for columns of the same name in other tables in the Model. Each time Data Director finds a match between two columns, it infers a relationship based on the primary key information; you can then accept the inferred relationship if it is valid.

For example, if Data Director encounters the primary key **customer_num** in the **customer** table, it searches for columns having the same name in other tables. When Data Director finds the **customer_num** column in the **orders** table, it is able to infer a one-to-many relationship between the **customer** table and the **orders** table based on the **customer_num** column in the **orders** table.

Data Director infers a one-to-many relationship between an origin table and a destination table if the origin column is the primary key in its table and the destination column is not the primary key in its table. If the destination column is the primary key in its table, Data Director infers a one-to-one relationship between the tables.

When you verify relationships, the Model Import Wizard displays the originating column as the origin column and the inferred foreign key column as the destination column, as shown in Figure 4-9. Selecting each origin column updates the destination column with the inferred relationships. You can also change the relationship type of each inferred relationship (also see "Modifying Relationships" on page 4-53).

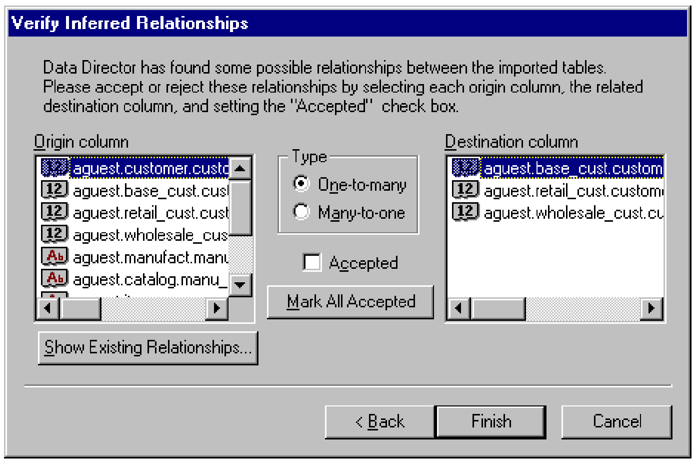


Figure 4-9
Verify Inferred Relationships Page

During a refresh, you can view the currently defined relationships between origin tables and destination tables (as shown in Figure 4-10) by clicking **Show Existing Relationships**.

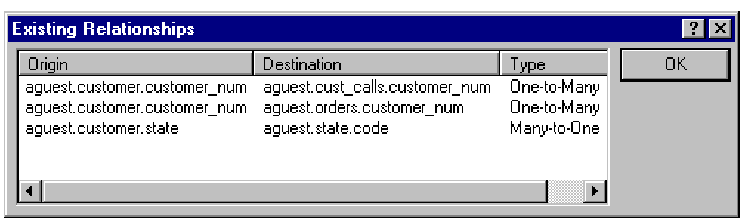


Figure 4-10
Existing Relationships Dialog Box

Tip: *Data Director imports but does not infer multiple-key relationships. If you use multiple-column primary keys and have not yet defined them in the database, you might need to manually create relationships in the Model Viewer (see "Creating Relationships" on page 4-48).*

Using an Imported Model

Once you capture your database structure, there are various tasks that you can perform using a Model:

- Viewing the Model
- Setting Model properties
- Setting a DBMS
- Connecting to and disconnecting from a database
- Setting primary keys
- Refreshing and saving the Model
- Working with tables
- Working with columns

Tip: *If you import a Model with the Model Viewer open, it remains open. If you import a Model without opening the Model Viewer, it is opened during the import and closes after you are prompted to save the Model.*

Viewing the Model

You can open a Model by choosing **File→Open Model** from the Visual Basic menu bar. After you import your database, Data Director displays information about the database in the Model Viewer window, as shown in Figure 4-11.

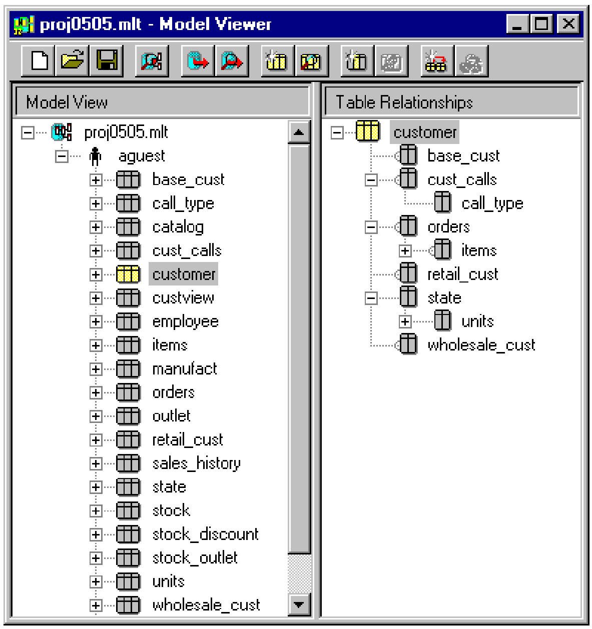
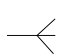


Figure 4-11
An Imported Model in the Model Viewer

The left side of the Model Viewer shows the tables and columns in the Model, including the table owners. The right side of the Model Viewer shows the relationships between tables in the Model.

The expanded Model in the Table Relationships pane indicates the type of relationship between tables. Relationships are graphically denoted in the table tree view as follows:

- Many-to-one (and one-to-one) relationships are denoted by a straight line.
- One-to-many relationships are denoted by the following symbol:



For example, Figure 4-11 on page 4-22 indicates that **customer** is related to **state** through a many-to-one relationship and that **customer** is related to **orders** through a one-to-many relationship.

The Model Viewer has a toolbar with buttons to perform common tasks. All of the commands in the Model Viewer toolbar are also accessible either through the Visual Basic menus or through the popup menus available when you right-click a Model item.

Setting Model Properties

There are several properties for Models that you can view or modify by choosing **Data Director→Model→Properties** from the Visual Basic menu bar. For example, you can see information about an imported Model, such as the database associated with the current Model.

If you modify the property settings for a Model, your changes take effect when you close the dialog box, and they persist until the next time you modify them.

Properties contain general information about a Model, such as the type of database with which the Model is associated and the number of tables and columns in the Model. You set properties on the **General** page of the Model Properties dialog box, shown in Figure 4-12.

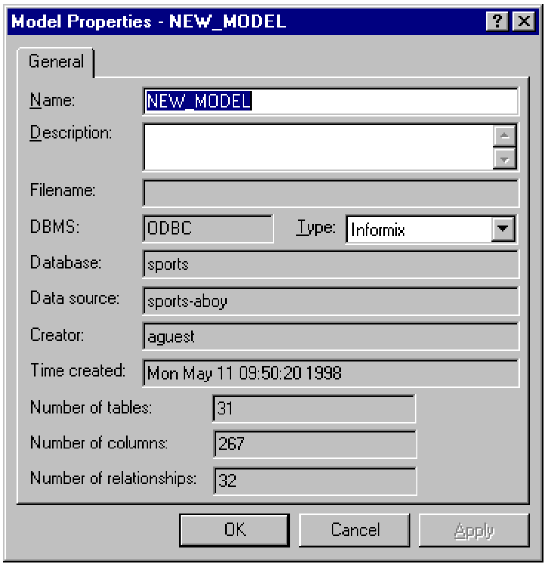


Figure 4-12
Model Properties Dialog Box

For descriptions of specific properties you can set, use the What's This? help.

You can modify the Model name, description, and database type. To change additional Model information, either refresh the database or save the Model to a new name or file location using the **File** menu.

Setting a DBMS

You can change the type of database on which a Model is based. For example, you can design an application using a Microsoft Access database, and then deploy the application using an Informix Dynamic Server database.

To ensure that the Model is consistent with the data types supported by the DBMS, refresh your model after changing the DBMS (see "Refreshing and Saving the Model" on page 4-27).

To set a DBMS

1. Select the Model in the Model Viewer.
2. Right-click the Model.
3. Choose **Set DBMS** from the popup menu.
4. Select a database driver and a data source from the lists in the Select Model Data Source dialog box, as shown in Figure 4-13.

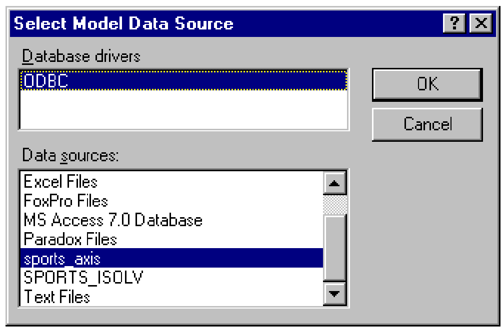


Figure 4-13
Select Model Data Source Dialog Box

Tip: *This dialog box is useful when you develop your application against a local database and then deploy against an enterprise database.*

Creating and Terminating Database Connections

Some operations, such as executing the SQL statements in a virtual table to check your syntax, require an active database connection. You can connect to a database from the Model Viewer after you associate a Model with a database.

To connect to a database

1. Right-click the Model in the Model Viewer.
2. Choose **Connect** from the popup menu.
3. Enter your password, and click **OK**.

Important: *Some databases do not require a password. In these cases, you will not see a dialog box asking for a password.*

After you complete your database operations, you can disconnect from your database.

To disconnect from a database

1. Right-click the Model in the Model Viewer.
2. Choose **Disconnect** from the popup menu.

Setting Primary Keys

If you did not choose to set primary keys during the import process, you can set primary key information for columns after you import a Model.

To set primary keys

1. Right-click the Model in the Model Viewer.
2. Choose **Set Primary Keys** from the popup menu.
3. Select primary key columns for any tables that are missing primary key information.

Figure 4-14 shows the Primary Keys dialog box.

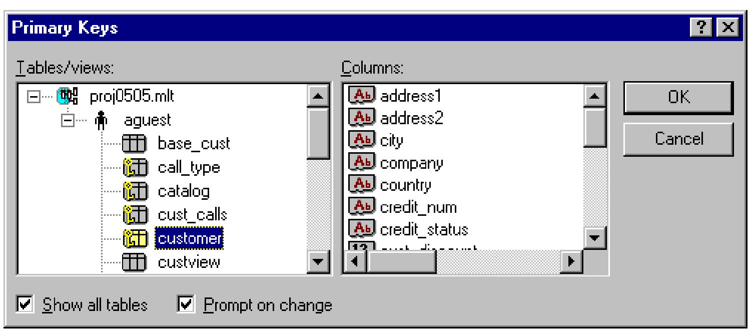


Figure 4-14 *Primary Keys Dialog Box*

Refreshing and Saving the Model

If you make changes to your database, you should ensure that the following Model elements match the underlying database:

- Table names and owners
- Column names, data types, and sizes

All other Model information, such as table and column descriptions, is database-independent. For example, if you enter a column default value in your Model and then update the Model, the column default value is not overwritten.

If the structure of your database changes, you should refresh your Model to ensure that both your database and the Model contain the same information.

If you make changes to the underlying database that is associated with a Model, you can easily refresh the Model by importing your database to the existing Model. Data Director identifies any differences between the database and your Model and prompts you to refresh your Model to reflect those differences.

For example, if a column has the same name in both the Model and the database, but the data type does not match in both sources, Data Director notifies you that the columns do not match and prompts you to specify whether you want to overwrite the Model with the database data type value.

To refresh a Model

1. Open the Model in the Model Viewer.
2. Choose **Data Director→Model→Import** from the Visual Basic menu bar.
3. Click **Refresh Current** in the Refresh or Import? dialog box.
4. Click **Options** to modify the last-used import options.

Important: *If you do not change the import options, the Model will be refreshed as specified by the last-used options. Only changed and additional database items specified by the Model Import Options criteria will be used to refresh the Model.*

5. Follow the Model Import Wizard instructions.

To save a Model

1. Choose **File→Save Model** from the Visual Basic menu bar.

Working with Tables

You can work with tables through the Model without connecting to the underlying database. You can create both base and virtual tables, and modify existing tables.

Creating Tables

When you import a database, Data Director automatically creates tables in a Model based on the items that you import. However, there might be times when you need to create a new table, such as when you want to quickly update your Model to match your database without refreshing your entire Model.

Data Director requires that you provide each table in a Model with a unique name; all other information is optional. The description can include anything you like, such as the purpose of the table or the type of information it stores.

Important: *The table names in your Model must match the table names in your database. If the database is case sensitive, consider case when naming tables.*

To create a table

1. Choose the **Add Table** menu item:

   VB 4 — Choose **Insert→Add Table** from the Visual Basic menu bar. ♦

   VB 5 — Choose **Project→Add Table** from the Visual Basic menu bar. ♦

   For both versions of Visual Basic, you can also right-click the Model (the **.mlt** icon) in the Model Viewer, and choose **Add Table** from the popup menu.
2. Enter a name for the new table in the Table Properties dialog box.
3. Optionally, enter additional information for the new table.

The Table Properties dialog box is shown in Figure 4-15.

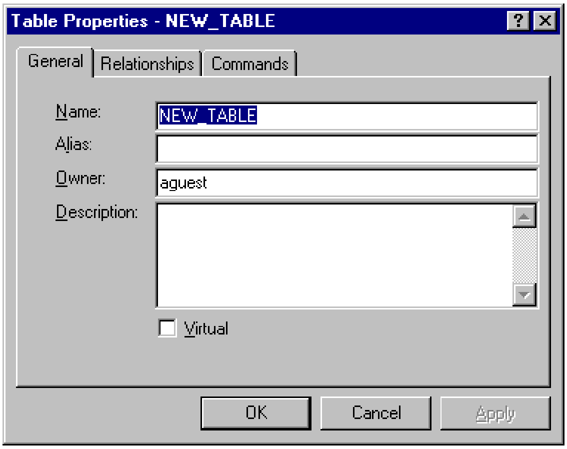


Figure 4-15
Table Properties Dialog Box

Specifying a Table Alias

A table alias provides a name for a table that is easily identifiable. For example, if a database table is named **cust_trx** in the underlying database, you can specify an alias of **Customer Transactions** to easily identify the table while designing an application.

Creating Virtual Tables

Virtual tables let you pass custom SQL statements directly through Data Director to your database. For example, virtual tables can be used to execute the following commands and queries:

- Ad hoc queries
- Subqueries
- Group-by clauses
- Stored procedures
- User-defined routines

At design time, you can create a virtual table and assemble the SQL statements that construct the table. When you check the syntax for a virtual table, Data Director executes the SQL statement specified for the table and returns the columns generated by the SQL statement. You can then link those columns to Visual Basic controls in an application window.

At runtime, Data Director executes the SQL statements stored in a virtual table instead of dynamically creating SQL statements as it does for regular tables. Virtual tables also store the result set returned by the statements, allowing you to access and manipulate the data in an application window.

To create a virtual table

1. Create a table using the Table Properties dialog box.

   For detailed instructions, see "To create a table" on page 4-28.

2. Check the **Virtual** check box on the **General** page to enable the **Commands** page.

   The **Commands** page is enabled only when a virtual table is created or selected.

3. On the **Commands** page, enter the SQL statements that you want to associate with the virtual table.

4. Click **Check Syntax** to check your SQL syntax.

The virtual table will not be created unless the syntax is checked and passes.

Tip: *If you later change the SQL syntax for a virtual table command, and the* ***Check Syntax*** *command fails, your changes to the SQL syntax will not be saved until you correct the syntax.*

The **Commands** page is shown in Figure 4-16.

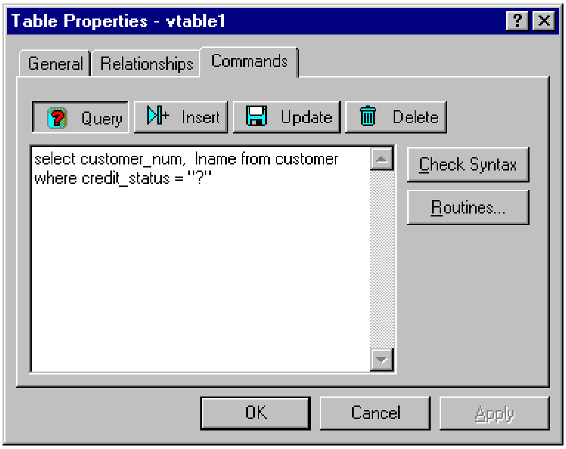


Figure 4-16
SQL Statements for a Virtual Table

Specifying SQL Statements

This section discusses how to specify the SQL statements that Data Director executes in the virtual table. It includes the following topics:

- Entering the statements
- Using data from a physical table
- Checking your syntax

Entering the Statements

To enter a SELECT statement, click **Query** on the **Commands** page of the Table Properties dialog box. The SELECT statement determines the set of columns that the virtual table contains, while the remaining statements determine what operations Data Director performs with the data in the table.

To enter additional statements, such as INSERT statements, click the appropriate button, and enter the rest of your SQL statements.

If your initial SQL query is a single-table SELECT statement, Data Director automatically generates any necessary INSERT, UPDATE, and DELETE statements in the virtual table. For example, if a virtual table is created using the SQL statement `SELECT * from customer`, and the user changes a customer's last name to `Burke` for customer number 101 and then clicks the **Save** button on the Navigation Control, Data Director generates the following UPDATE statement:

```
update customer set lname to "Burke" where customer_num = 101
```

You can override this behavior by explicitly setting each command in the virtual table.

Using Data from a Physical Table

You can retrieve data from the current record in a physical table and place it into the SQL statement used by the virtual table. For example, if you have a virtual table called **orders**, and you want to show only orders for the current customer in your query, your SQL statement would look like this:

```
SELECT * from orders where customer_num =
                       ::customer.customer_num
```

The double colon (: :) indicates to Data Director that `customer.customer_num` is a replacement parameter. Data Director then replaces the variable with the value in the current record of the **customer** table for the **customer_num** column.

You can use replacement parameters for the SELECT, INSERT, UPDATE, and DELETE statements. Because parameter replacement provides a runtime binding to a fetched data value in the current record of a referenced table, you can use replacement parameters only at runtime.

With some database drivers, you can also use binding (question mark) parameters to replace a column value with any compatible value at runtime. For more information about parameters, see "Parameters in the DDO" on page 9-17.

Checking Your Syntax

Click **Check Syntax** to check the syntax in your SQL statements. When you check syntax, Data Director connects to your database if necessary and describes the command. Checking your syntax allows Data Director to catch errors in your commands and is a required step for Data Director to extract column information from your database.

After Data Director describes the SELECT statement, it creates a column in the virtual table for each output column. You can see the columns in the Model Viewer and DataLink Manager and link them to Visual Basic controls.

If the SQL statement contains parameters, you can set parameter values through the Parameter Values dialog box, which is shown in Figure 4-17.

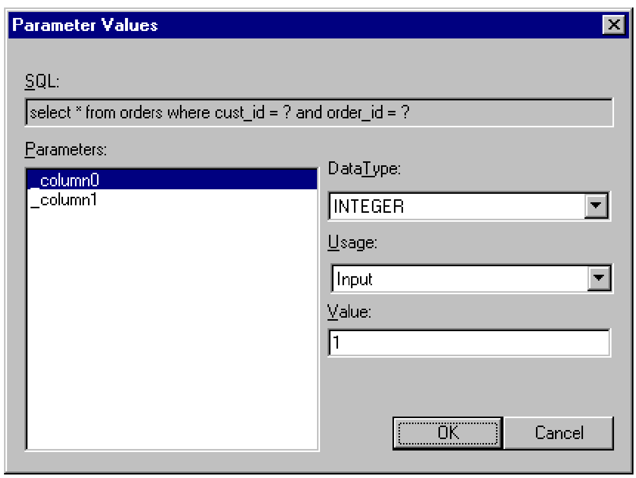


Figure 4-17
Parameter Values Dialog Box

Tip: *For Informix databases, you must manually set the data type of the parameter. No Informix ODBC driver supports the **SQLDescribeParam** function.*

Using Routines

Data Director supports both stored procedures (including functions) and user-defined routines: all three are known as *routines* within Data Director. Routines owned by user **informix** in an Informix database are treated as system objects. Therefore, to import Informix routines, you must check the **Include system objects** check box in the Import Options dialog box.

Before importing routines using the Model Import Wizard, you should verify that the **Routines** and (for Informix databases only) the **Include system objects** check boxes are both checked in the Import Options dialog box (see "Deciding Which Items to Import" on page 4-12).

To use a routine in your virtual table, click **Routines** on the **Commands** page of the Table Properties dialog box. Data Director displays a list of the routines you imported into your Model during the Model import process. Select a routine from the list to include it in your SQL statement.

The Routine Selection dialog box is shown in Figure 4-18.

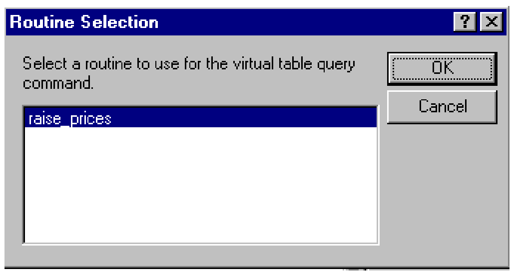


Figure 4-18
Routine Selection Dialog Box

To use a routine with Informix Dynamic Server databases, you can use the standard Informix syntax; for example:

```
execute procedure raise_prices(5)
```

If you are using Informix Dynamic Server with Universal Data Option, you must include type casts with the parameters; for example:

```
execute procedure raise_prices(5::integer)
```

For more information about user-defined routines in Informix databases, see the manual *Extending Informix Dynamic Server*. For details and examples about using routines programmatically, see "Working with Routines" on page 9-48.

Modeling Routines

You can import routines using the Model Import Wizard (see "Selecting Import Items" on page 4-16).

Routines appear in the Model Viewer with their parameters and return values as shown in Figure 4-19, where **oidsetprime** is the name of a user-defined routine.

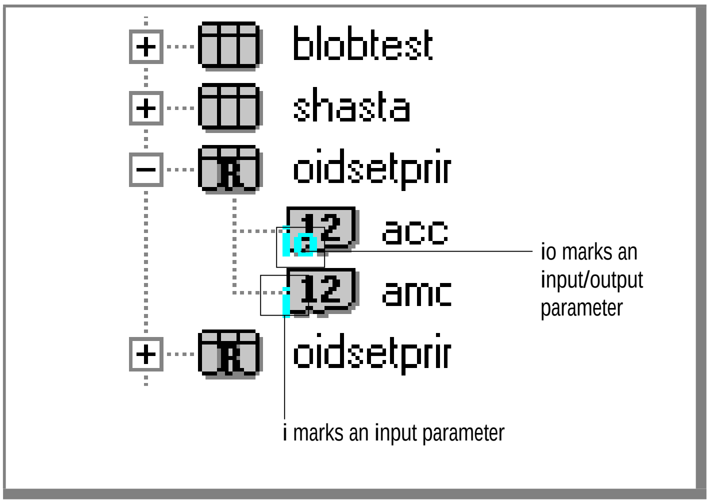


Figure 4-19
A User-Defined Routine in the Model Viewer

Parameters are displayed with an indicator that shows whether the parameter is input (**i**), output (**o**), input/output (**io**), or a return value (**r**).

You can add and modify any of the parameters in the parameter list.

Modifying Routine Properties

You can verify and edit the routine name, server name, and owner, and verify the routine signature, through the Routine Properties dialog box, as shown in Figure 4-20.

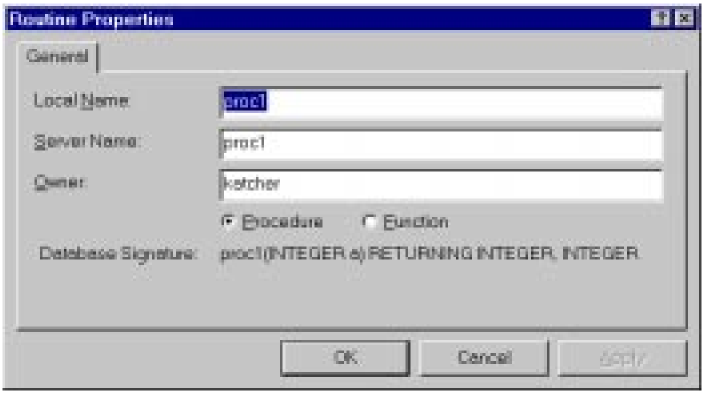


Figure 4-20 *Routine Properties Dialog Box*

To access the Routine Properties dialog box, right-click the routine name in the Model Viewer and select **Properties** from the popup menu. Click **Procedure** or **Function** to indicate whether Data Director should generate EXECUTE PROCEDURE or EXECUTE FUNCTION in the SQL statement that executes the routine.

Using Parameters with Routines

In the code example beneath Figure 4-18 on page 4-34, you could also use a binding parameter (signified by a question mark) in place of the value 5. The parameter is replaced with a value supplied by virtual-table code at runtime. Data Director automatically generates code to call your routines using parameters and places it on the **Commands** page in the Table Properties dialog box, as shown in Figure 4-21.

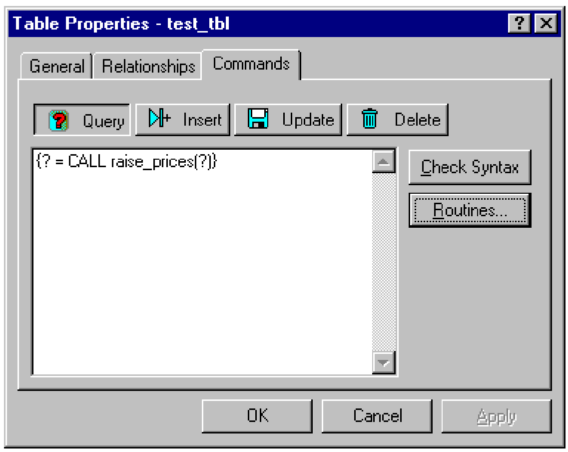


Figure 4-21
Commands Page for a Routine

Using parameterized statements in this way is more flexible and can speed the performance of your application. For more information, and for general information about parameters, see "Parameters in the DDO" on page 9-17.

You can add parameters to your routine within the Model as follows.

To add a parameter to a routine

1. Right-click the routine name.

A popup menu appears, as shown in Figure 4-22.

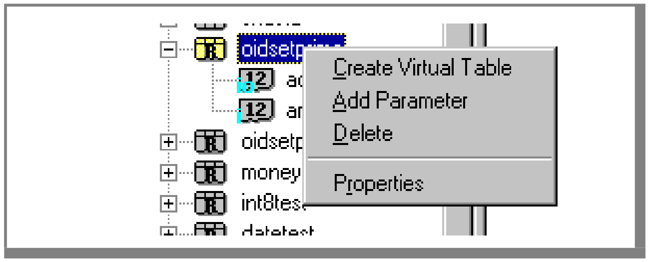


Figure 4-22
Routines Popup Menu

2. Choose **Add Parameter**.

The Parameter Properties dialog box appears, as shown in Figure 4-23.

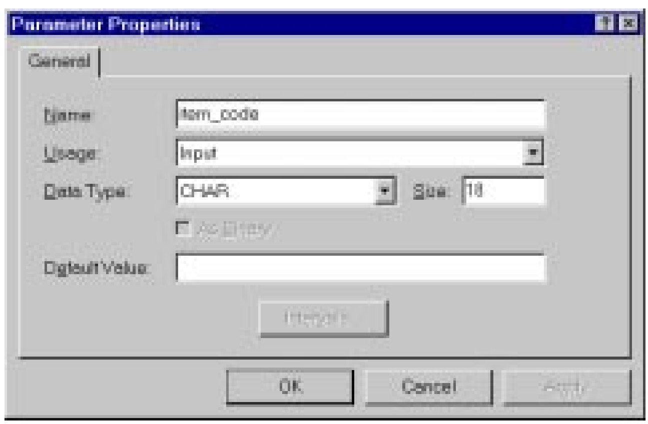


Figure 4-23
Parameter Properties Dialog Box

3. Specify a name for the parameter.

4. Select a usage type (input, output, input/output, or return) from the **Usage** list box.

5. Select a data type from the **Type** list box.

6. Specify a size in the **Size** text box.

For more information, see "Specifying a Numeric Column Size" on page 4-42.