



Sybase® Adaptive Server™ Enterprise
Getting Started with Sybase Adaptive Server Enterprise
for Windows NT

Adaptive Server™

Document ID: 36097-01-1150

September 1997

Copyright Information

Copyright © 1989–1997 by Sybase, Inc. All rights reserved.

Sybase, Inc., 6475 Christie Avenue, Emeryville, CA 94608.

Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

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

Sybase, the Sybase logo, APT-FORMS, Certified SYBASE Professional, Data Workbench, First Impression, InfoMaker, PowerBuilder, Powersoft, Replication Server, S-Designer, SQL Advantage, SQL Debug, SQL SMART, SQL Solutions, Transact-SQL, VisualWriter, and VQL are registered trademarks of Sybase, Inc. Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Monitor, ADA Workbench, AnswerBase, Application Manager, AppModeler, APT-Build, APT-Edit, APT-Execute, APT-Library, APT-Translator, APT Workbench, Backup Server, BayCam, Bit-Wise, ClearConnect, Client-Library, Client Services, CodeBank, Column Design, Connection Manager, DataArchitect, Database Analyzer, DataExpress, Data Pipeline, DataWindow, DB-Library, dbQ, Developers Workbench, DirectConnect, Distribution Agent, Distribution Director, Dynamo, Embedded SQL, EMS, Enterprise Client/Server, Enterprise Connect, Enterprise Manager, Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, EWA, Formula One, Gateway Manager, GeoPoint, ImpactNow, InformationConnect, InstaHelp, InternetBuilder, iScript, Jaguar CTS, jConnect for JDBC, KnowledgeBase, Logical Memory Manager, MainframeConnect, Maintenance Express, MAP, MDI Access Server, MDI Database Gateway, media.splash, MetaWorks, MethodSet, Net-Gateway, NetImpact, Net-Library, ObjectConnect, ObjectCycle, OmniConnect, OmniSQL Access Module, OmniSQL Toolkit, Open Client, Open ClientConnect, Open Client/Server, Open Client/Server Interfaces, Open Gateway, Open Server, Open ServerConnect, Open Solutions, Optima++, PB-Gen, PC APT-Execute, PC DB-Net, PC Net Library, Power++, Power AMC, PowerBuilt, PowerBuilt with PowerBuilder, PowerDesigner, Power J, PowerScript, PowerSite, PowerSocket, Powersoft Portfolio, Power Through Knowledge, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, Quickstart Datamart, Replication Agent, Replication Driver, Replication Server Manager, Report-Execute, Report Workbench, Resource Manager, RW-DisplayLib, RW-Library, SAFE, SDF, Secure SQL Server, Secure SQL Toolset, Security Guardian, SKILLS, smart.partners, smart.parts, smart.script, SQL Anywhere, SQL Central, SQL Code Checker, SQL Edit, SQL Edit/TPU, SQL Modeler, SQL Remote, SQL Server, SQL Server/CFT, SQL Server/DBM, SQL Server Manager, SQL Server SNMP SubAgent, SQL Station, SQL Toolset, Sybase Client/Server Interfaces, Sybase Development Framework, Sybase Gateways, Sybase IQ, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase Synergy Program, Sybase Virtual Server Architecture, Sybase User Workbench, SybaseWare, SyBooks, System 10, System 11, the System XI logo, SystemTools, Tabular Data Stream, The Architecture for Change, The Enterprise Client/Server Company, The Model for Client/Server Solutions, The Online Information Center, Translation Toolkit, Turning Imagination Into Reality, Unibom, Unilib, Uninull, Unisep, Unistring, Viewer, Visual Components, VisualSpeller, WarehouseArchitect, WarehouseNow, Warehouse WORKS, Watcom, Watcom SQL, Watcom SQL Server, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, and XA-Server are trademarks of Sybase, Inc. 6/97

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

Table of Contents

About This Book

Audience	xiii
How to Use This Book	xiii
Related Documents	xiii
Other Sources of Information	xiv
Sybase Certifications on the Web	xiv
Conventions	xv
If You Need Help	xvi

1. Overview

About the Tutorial	1-1
When and Where to Begin	1-1
The Business Problem	1-2
The Solution	1-3
What You Will Do in the Tutorial	1-4
Adaptive Server Concepts	1-7
Client/Server	1-8
About SQL and Transact-SQL	1-9
About APIs	1-9
APIs and the Internet	1-10
What's Next	1-10

2. Sybase Central and Adaptive Server System Administration Basics

Where You Are	2-1
Project One: Getting Ready for the Tutorial	2-1
Where to Find the Applications and Utilities	2-2
About the Adaptive Server Environment	2-2
Task 1: Find Out About Your Environment	2-2
Task 2: Copy the <i>tutorial</i> Directory from the Product CD	2-3
Running Transact-SQL Scripts	2-3
Project Two: Getting Started with Sybase Central	2-3
About the Sybase Central Interface	2-4
Task 1: Start Adaptive Server	2-4
About the "sa" Login	2-5
Task 2: Log into Adaptive Server with the "sa" Login	2-6

Task 3: Examine the “sa” Login Properties	2-7
Task 4: Examine the Current Processes	2-7
Task 5: Examine the <i>master</i> Database	2-7
Project Three: Preparing Adaptive Server for Development	2-8
Task 1: Create the “student” Login for Use in This Tutorial	2-8
Task 2: Give the New Login Roles	2-9
Task 3: Log In As “student”	2-9
Task 4: Create a Device for the Events Database	2-9
Task 5: Examine the size of <i>tempdb</i>	2-10
Task 6: Extend <i>tempdb</i> to the <i>master</i> Device (Optional)	2-12
Task 7: Create a New Device and Extend <i>tempdb</i> (Optional)	2-13
Task 8: Set the Database Option <i>truncate log on checkpoint</i>	2-14
What’s Next	2-14

3. Designing the Events Database with SQL Modeler

Where You Are	3-1
Project One: Designing the Events Database	3-1
Tables, Columns, and Rows	3-2
Relationships and Keys	3-3
Relationships and Data Retrieval	3-4
Business Rules	3-5
Enforcing Business Rules	3-6
Indexes	3-7
Project Two: Getting Started with SQL Modeler	3-8
Task 1: Start SQL Modeler	3-8
Task 2: Configure an ODBC Data Source for Database Creation	3-9
Task 3: Create the <i>category</i> Table	3-9
Project Three: Working with the Predefined <i>eventsdb.PDM</i>	3-11
Task 1: Open <i>eventsdb.PDM</i>	3-11
Task 2: Examine the Club Table and Work with a Business Rule	3-12
Task 3: Examine the Model Properties	3-12
Task 4: Examine the Business Rules	3-14
Task 5: Define the References	3-14
Task 6: Associate the Reference with Business Rules	3-15
Task 7: Create the Date Index	3-16
Task 8: Create the <i>artist_club_date</i> View	3-16
Project Four: Generating the Database	3-18
Task 1: Open <i>after.PDM</i>	3-18
Task 2: Generate a Script	3-19
Task 3: Generate the Database	3-20

Task 4: Generate the Stored Procedures	3-20
Task 5: Save the PDM File	3-21
What's Next	3-21

4. Entering and Importing Data with InfoMaker

Where You Are.....	4-1
Project One: Connecting InfoMaker to Adaptive Server.....	4-1
Connectivity Overview	4-2
Task 1: Verify Installation of or Install the Powersoft Stored Procedures	4-2
Task 2: Create a Database Profile	4-3
Task 3: Examine the Powersoft Repository Tables in Sybase Central ...	4-5
Task 4: Run the Scripts to Populate the Powersoft Repository	4-6
Project Two: Entering and Importing Data into the <i>eventsdb</i> Database.....	4-6
Data Entry Overview.....	4-6
Task 1: Add Text to the PowerBar Display	4-7
Task 2: Open the Tutorial Library.....	4-7
Task 3: Open the Data Manipulation Painter for the Category Table ...	4-8
Task 4: Add Rows to the Category Table.....	4-9
Task 5: Import a File	4-9
Project Three: Piping Data from Another Database.....	4-10
Task 1: Configure an ODBC Data Source for <i>msclub</i>	4-10
Task 2: Create a Database Profile for the MS Access Database.....	4-11
Task 3: Create the Data Pipeline	4-11
Project Four: Defining Extended Attributes	4-13
Task 1: Examine a Validation Rule	4-13
Task 2: Define a Display Format	4-13
About Edit Styles	4-14
Project Five: Creating Forms	4-15
Task 1: Create the Event Data Entry Form	4-16
Task 2: Add an Action Button.....	4-16
Task 3: Run the Form	4-17
What's Next	4-18

5. Creating Reports and Applications with InfoMaker

Where You Are.....	5-1
About the Reports	5-1
Project One: Developing the Event Listing Report.....	5-2
Task 1: Create the New Report	5-2
Task 2: Preview the Report (Retrieve Rows).....	5-3

Task 3: Add a Filter	5-4
Project Two: Developing the City-Type Report	5-6
Task 1: Start the New Report	5-6
Task 2: Define the Data Source	5-7
Project Three: Enhancing the Revenues Report	5-9
Task 1: Open the Predefined Report	5-9
Task 2: Add a Retrieval Argument	5-10
Project Four: Creating an InfoMaker Application	5-11
Task 1: Create an Application	5-11
Task 2: Use the Application	5-12
About Deploying Applications	5-12
What's Next	5-13

6. Building a Dynamic Web Site with NetImpact Dynamo

Where You Are	6-1
NetImpact Dynamo As an Application Server	6-2
Beyond Client/Server: The Internet	6-2
Beyond HTML: Dynamic Content	6-3
Project One: Getting Started with NetImpact Dynamo	6-4
Task 1: Configure an ODBC Data Source for the Web Site	6-4
Task 2: Create a Connection Profile	6-5
Task 3: Connect to <i>eventsdb</i>	6-5
Task 4: Examine the Web Site Tables	6-6
Project Two: Creating a Template	6-7
About Embedded Queries and Formatting	6-8
Task 1: Create the Calendar Template with an Embedded Query	6-9
Task 2: Configure the NetImpact Dynamo Personal Web Server	6-11
Task 3: Start the Personal Web Server and a Web Browser	6-12
Task 4: Map MusicSite to a Port Number (Optional)	6-13
Task 5: View the Output	6-14
Project Three: Improving the Template	6-16
Task 1: Refine the Formatting and the Query	6-16
Task 2: Save Changes to the Database and Reload the URL	6-18
Project Four: Working with DynaScript	6-19
About DynaScript	6-19
Task 1: Add DynaScript to the Template	6-21
About Script Error Messages	6-22
Task 2: Pass an Argument to a Template	6-22
Task 3: Open a Template That References a Query with DynaScript ..	6-24
Project Five: Working with Forms	6-26

Task 1: Work with a Form Page.	6-26
Task 2: Edit the Calendar Template to Receive the Argument.	6-28
Task 3: Pass the Argument to the Template via the Form.	6-28
Project Six: Importing the Web Site.	6-29
About the Application	6-29
Task 1: Import the Predefined Site	6-31
Task 2: Try Out the Application from a Web Browser	6-31
How the Application Works	6-32
About the Script Files.	6-32
Navigation Bar Generation in <i>navbar.ssc</i>	6-33
HTML Formatting in <i>corefmt.ssc</i>	6-33
Client-Side Data Checking with JavaScript	6-33
What's Next	6-34

7. When You Have Completed the Tutorial

Where You Are.	7-1
Project: Undoing the Tutorial Activities.	7-1

A. Using *isql* to Run Scripts

Getting a Command Prompt	A-1
Tips	A-1
How to Run Scripts.	A-2
Example	A-2

Index

List of Figures

Figure 1-1:	Working with Sybase Central	1-5
Figure 1-2:	SQL Modeler's graphical view of the musical events database	1-5
Figure 1-3:	The dbArts InfoMaker application.....	1-6
Figure 1-4:	The dbArts Web application	1-7
Figure 1-5:	A Sybase client/server configuration.....	1-8
Figure 2-1:	Available servers in Sybase Central	2-5
Figure 2-2:	Adaptive Server information displayed in Sybase Central.....	2-6
Figure 2-3:	Setting the device size	2-10
Figure 2-4:	Examining the size of <i>tempdb</i>	2-11
Figure 3-1:	The <i>eventsdb</i> database model.....	3-4
Figure 3-2:	Selecting columns and rows from the club table	3-4
Figure 3-3:	A join	3-5
Figure 3-4:	The tool palette.....	3-8
Figure 3-5:	Columns Definition dialog box	3-10
Figure 3-6:	Title box.....	3-12
Figure 3-7:	Eventsdb options	3-13
Figure 3-8:	Selecting columns for the view.....	3-17
Figure 4-1:	How InfoMaker connects to Adaptive Server.....	4-2
Figure 4-2:	InfoMaker menu bar	4-4
Figure 4-3:	Database Profile list.....	4-4
Figure 4-4:	Partial list of objects in the default library	4-8
Figure 4-5:	Tutor library.....	4-8
Figure 4-6:	Data Pipeline painter workspace	4-12
Figure 4-7:	DropDownDataWindow edit style	4-15
Figure 4-8:	The data entry form.....	4-17
Figure 5-1:	Scrolling through a report	5-4
Figure 5-2:	Specifying a filter	5-5
Figure 5-3:	A crosstab.....	5-6
Figure 5-4:	Select painter workspace	5-7
Figure 5-5:	Defining a crosstab	5-8
Figure 5-6:	Select all reports and forms for the application.	5-11
Figure 6-1:	A NetImpact Dynamo Web site configuration.....	6-3
Figure 6-2:	Working with Adaptive Server and NetImpact Dynamo connections.....	6-7
Figure 6-3:	A basic Dynamo template	6-9
Figure 6-4:	Configuring the Personal Web Server	6-12
Figure 6-5:	Template source code and the resulting HTML code.....	6-15
Figure 6-6:	Final output in a browser	6-16
Figure 6-7:	Altering the template	6-17

Figure 6-8:	Refining the query	6-18
Figure 6-9:	A complex Dynamo template	6-20
Figure 6-10:	Adding DynaScript	6-21
Figure 6-11:	Adding DynaScript	6-22
Figure 6-12:	Using a text replacement macro	6-23
Figure 6-13:	Advanced elements of the calendar template	6-25
Figure 6-14:	An HTML form	6-27
Figure 6-15:	Referencing an argument from a form	6-28
Figure 6-16:	The dbArts Web application	6-30
Figure 6-17:	How the dbArts Web application interacts with the database	6-31

About This Book

This book, *Getting Started with Adaptive Server Enterprise for Windows NT*, provides a hands-on introduction to Adaptive Server Enterprise for Windows NT, an integrated set of Sybase® products for developing and deploying client/server and Internet-based relational database applications on desktop platforms.

Audience

This guide is for system administrators, developers, managers, or anyone who will be involved in setting up or using Adaptive Server Enterprise for Windows NT or who wants to understand how the product's components work together.

How to Use This Book

The purpose of this tutorial is to give you a hands-on introduction to Adaptive Server Enterprise for Windows NT. It is a good idea to go through the tutorial soon after installation, before your site sets up a production server.

The tutorial includes seven chapters; each chapter is a lesson. Lessons are comprised of multiple projects. The projects themselves are made up of tasks.

For more information on using the tutorial, see “When and Where to Begin” on page 1-1.

Related Documents

- *Introducing Adaptive Server Enterprise for Windows NT* provides an overview of Adaptive Server Enterprise for Windows NT. It includes a glossary that you might want to refer to when using this tutorial.
- *NetImpact Dynamo User's Guide* includes hands-on lessons that will help you get up and running with NetImpact™ Dynamo as quickly as possible.

Other Sources of Information

Use the SyBooks™ and SyBooks-on-the-Web online resources to learn more about your product:

- SyBooks documentation is on the CD that comes with your software. The DynaText browser, also included on the CD, allows you to access technical information about your product in an easy-to-use format.

Refer to *Installing SyBooks* in your documentation package for instructions on installing and starting SyBooks.

- SyBooks-on-the-Web is an HTML version of SyBooks that you can access using a standard Web browser.

To use SyBooks-on-the-Web, go to <http://www.sybase.com>, and choose Documentation.

Sybase Certifications on the Web

The Technical Support information on the Sybase Web site is updated frequently.

For the latest information on product certifications and/or the EBF Rollups:

1. Point your Web browser to the Technical Information Library at the following URL:
<http://techinfo.sybase.com>
2. In the Browse section, click on the Hot entry.
3. Explore your area of interest: Hot Docs covering various topics, or Hot Links to Technical News, Certification Reports, Partner Certifications, and so on.

If you are a registered SupportPlus user:

1. Point your Web browser to the Technical Information Library at the following URL.
<http://techinfo.sybase.com>
2. In the Browse section, click on the Hot entry.
3. Click on the EBF Rollups entry.

You can research EBFs using the Technical Information Library, and you can download EBFs using Electronic Software Distribution (ESD).

4. Follow the instructions associated with the *SupportPlus*SM Online Services entries.

If you are not a registered *SupportPlus* user and want to become one:

You can register by following the instructions on the Web.

To use *SupportPlus*, you need:

- A Web browser that supports the Secure Sockets Layer (SSL), such as Netscape Navigator 1.2 or later
- An active support license
- A named technical support contact
- Your user ID and password

Whether or not you are a registered *SupportPlus* user:

You may use Sybase's Technical Information Library. Certification Reports are among the features documented at this site.

1. Point your Web browser to the Technical Information Library at the following URL:
`http://techinfo.sybase.com`
2. In the Browse section, click on the Hot entry.
3. Click on the topic that interests you.

Conventions

This manual uses the following style conventions:

- The names of files, directories, and database objects appear in italics:
`\data\master.dat`
- Examples showing the use of Transact-SQL[®] commands are printed like this:

```
select titles, pub_id
from titles
```
- The names of utilities, system procedures, commands, and scripts appear in the following font:
`sp_revokelogin`
- This manual instructs you to choose commands from menus as follows:

Choose Rows→Filter

If You Need Help

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

1

Overview

Adaptive Server Enterprise for Windows NT integrates a high-performance relational database with a full suite of tools for managing databases and Web sites. This book provides a simple example of how the database and tools fit together and helps you get them up and running as quickly as possible.

This chapter describes a business problem and an Adaptive Server Enterprise for Windows NT solution to the problem. It summarizes the steps that you will go through in the lessons that follow in order to implement the solution. Finally, it discusses a few concepts that will be useful to you as you go through the tutorial. The following topics are included:

- About the Tutorial 1-1
- The Business Problem 1-2
- The Solution 1-3
- What You Will Do in the Tutorial 1-4
- Adaptive Server Concepts 1-7

About the Tutorial

The purpose of this tutorial is to give you a hands-on “tour” of Adaptive Server Enterprise for Windows NT and to show you how the components work together. It also addresses Adaptive Server-specific connectivity issues that will make working with Adaptive Server Enterprise for Windows NT easier.

This tutorial does not teach you how to use the specific components in any detail. For complete information about each component, see the component-specific documentation.

When and Where to Begin

It is recommended that you do this tutorial soon after Adaptive Server Enterprise for Windows NT is installed. After you have set up Adaptive Server as a production database server, you may not want to do the kind of experimentation required in the following lessons. If a new staff member wants to do the tutorial later, the System

Administrator should work closely with him or her to monitor storage issues and user roles.

Begin with the first lesson—the lessons are designed to be completed in order; each lesson builds on the previous one. You are free, however, to skip any lesson, except Chapter 2, “Sybase Central and Adaptive Server System Administration Basics.” Also, you must complete the tasks in the last chapter, Chapter 7, “When You Have Completed the Tutorial.” If you do skip a lesson, the lesson that follows gives instructions on how to update the database in order to do the current lesson.

► *Note*

Be sure to complete the tasks in the last chapter, Chapter 7, “When You Have Completed the Tutorial.” It is important to undo the work you have done for the sake of Adaptive Server security and to leave a clean slate for the next student.

The Business Problem

You work for the headquarters of a non-profit organization called “dbArts,” which promotes public enjoyment of the performing arts. dbArts has four offices in different locations in the western United States; the offices are linked by a computer network.



One of your projects is to develop a database to track musical events in the East Bay region of the San Francisco Bay Area. The California office, which already tracks some of the data on a desktop database, will handle data entry and will also provide a listing of upcoming events happening in the area to the local free newspaper.

Managers at the headquarters, who work with corporate donors to generate funding for the arts, need to report on and analyze the data. For example, they want an overview of revenues and to know which venues are the most successful commercially. They also want to find

out which types of music are most popular in specific cities and which clubs offer support for non-commercial music.

The organization also wants to publish this database on the World Wide Web. They want residents to be able to find out about local musical events directly over the Internet. Also, the club managers or the artists themselves should be able enter information about upcoming events, taking the data entry work out of your hands and making the data immediately available to everyone else.

If the initial project is successful, you will need a system that can accommodate future expansion. You may want to install Adaptive Server in multiple offices but administrate the system from one location.

dbArts is a small organization that wants you to handle all phases of the project:

- Database and system administration
- Database design
- Creation and deployment of a front end for data entry and report generation
- Conversion of existing data
- Development of the Web application
- Performance tuning and system administration as the system grows

The Solution

Your assignment is to develop the system with Adaptive Server Enterprise for Windows NT, which includes:

- **Adaptive Server™**

Adaptive Server is a high-performance, fully scalable relational database management system (RDBMS).

- **Sybase Central™**



Sybase Central helps you manage database objects and perform common administrative tasks. You also create and manage Web sites in the Sybase Central window.

- **SQL Modeler™**



SQL Modeler helps you design database models, and generate, document, and maintain databases. You can also use SQL Modeler to reverse-engineer existing databases.

- **InfoMaker™**



InfoMaker is a reporting and data-maintenance tool that helps you make forms, reports, graphs, crosstabs, and tables, as well as simple applications that use these as building blocks. You can also move data between databases using the InfoMaker data pipeline.

- **NetImpact Dynamo**

NetImpact Dynamo is a suite of tools for building and managing Web applications linked to your Adaptive Server databases. It allows you to store, manage, and access both static HTML and dynamic database data locally or over a corporate intranet or the Internet.

- **Adaptive Server Monitor™**

Adaptive Server Monitor is a client/server application that helps you capture, display, and evaluate Adaptive Server performance data and tune Adaptive Server performance. You work with Adaptive Server Monitor through the Sybase Central interface.

- **Component Integration Services**

Component Integration Services is an Adaptive Server feature that allows users to connect to Sybase databases on remote servers as if the servers were local.

What You Will Do in the Tutorial

This tutorial leads you, step by step, through the major phases of the dbArts project. You will:

- Use Sybase Central to perform some basic system administration tasks that are necessary for the subsequent lessons. You will do this in Chapter 2, “Sybase Central and Adaptive Server System Administration Basics.”

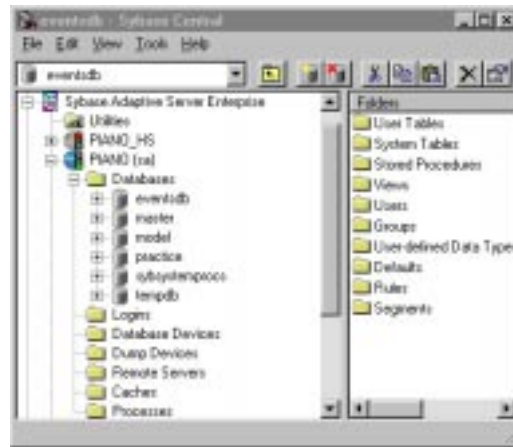


Figure 1-1: Working with Sybase Central

- Use SQL Modeler to design and generate the database for East Bay musical events, called *eventsdb*. You will do this in Chapter 3, “Designing the Events Database with SQL Modeler.”

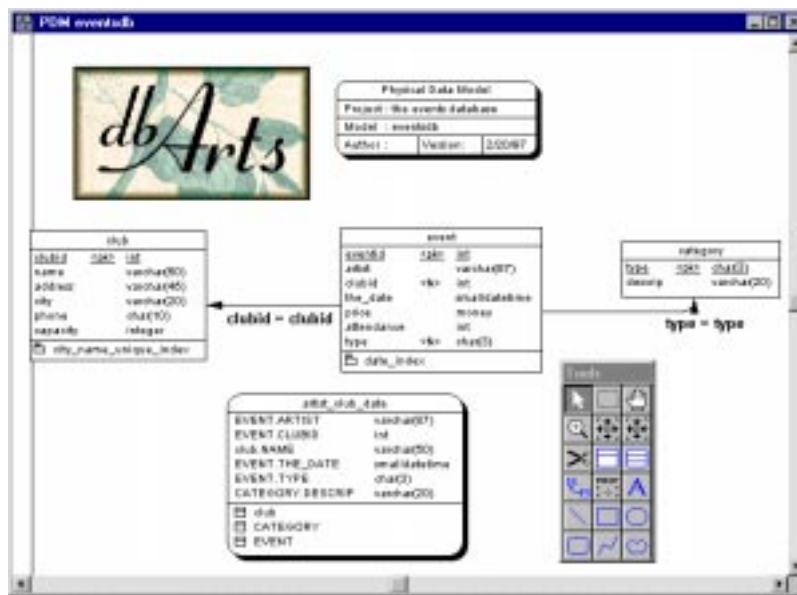


Figure 1-2: SQL Modeler's graphical view of the musical events database

- Use InfoMaker to enter and import data and to create a data-entry front end. You will do this in Chapter 4, “Entering and Importing Data with InfoMaker.”
- Use InfoMaker to create the *city_club*, *event_listing*, and *revenues* reports and to create an InfoMaker application. You will do this in Chapter 5, “Creating Reports and Applications with InfoMaker.”

Report - revenues					
Revenues					
Title of Music	Date and Time	Price	Attendance	Revenue	Percent Filled
Rock/Blas	28-Feb	\$0.00			
	1-Mar	\$0.00			
	7-Mar	\$13.00			
	8-Mar	\$14.00			
Number of events:	4			subtotal:	\$27.00
Cajun/Spyroes	26-Feb	\$5.00			
	27-Feb	\$7.00			
	27-Feb	\$6.00			
	5-Mar	\$5.00			
	6-Mar	\$5.00			
Number of events:	5			subtotal:	\$28.00

Figure 1-3: The dbArts InfoMaker application

- Create and manage the musical events database Web site, called MusicSite. You will do this in Chapter 6, “Building a Dynamic Web Site with NetImpact Dynamo.” Adaptive Server Enterprise for Windows NT includes a 10-user Personal Web Server that you can use for development.

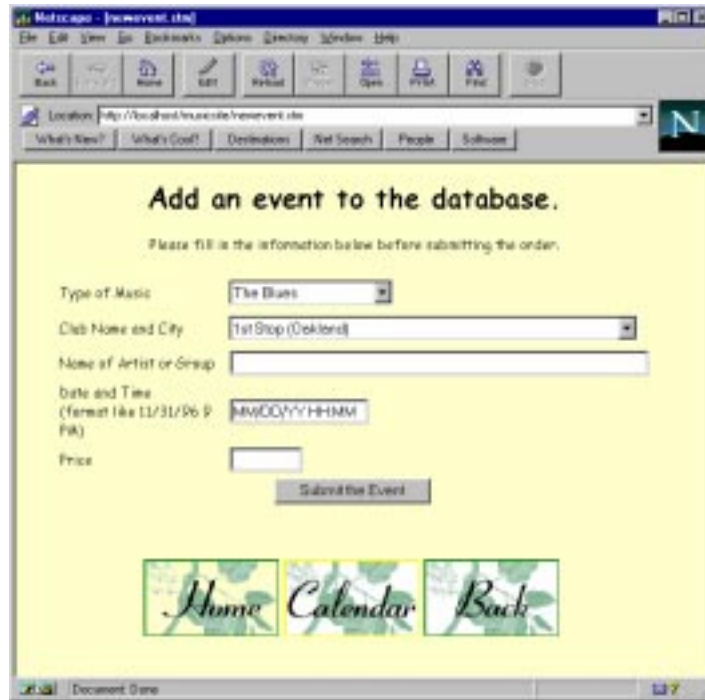


Figure 1-4: The dbArts Web application

As your system grows, you will want to start using Adaptive Server Monitor to monitor system performance. Although this tutorial does not include a lesson on using Adaptive Server Monitor, this tool is an important part of using Adaptive Server Enterprise for Windows NT. For information on using this tool, see the Adaptive Server Monitor documentation.

You may also want to deploy Adaptive Server Enterprise for Windows NT in all your office locations. Component Integration Services simplifies connecting to and using remote Adaptive Servers.

Adaptive Server Concepts

Before you begin using Adaptive Server, you need to understand the basic concepts discussed in this section. Also, see Chapter 3, "Designing the Events Database with SQL Modeler," for other basic

concepts that you will find useful. Some of the terms defined in this section include:

- Relational database management system (RDBMS)
- Client/server
- Application programming interfaces (APIs)
- SQL and Transact-SQL
- Open Database Connectivity (ODBC)
- Web browsers

Client/Server

Adaptive Server is a **relational database management system (RDBMS)**. In an Adaptive Server environment, the databases and the RDBMS software reside on one or more servers, where multiple clients can access them concurrently over a network. Applications that access and manipulate the shared data, as well as the libraries or **application programming interfaces (APIs)** that allow the applications to communicate with the RDBMS, reside on the clients. Figure 1-5 illustrates the Sybase client/server configuration.

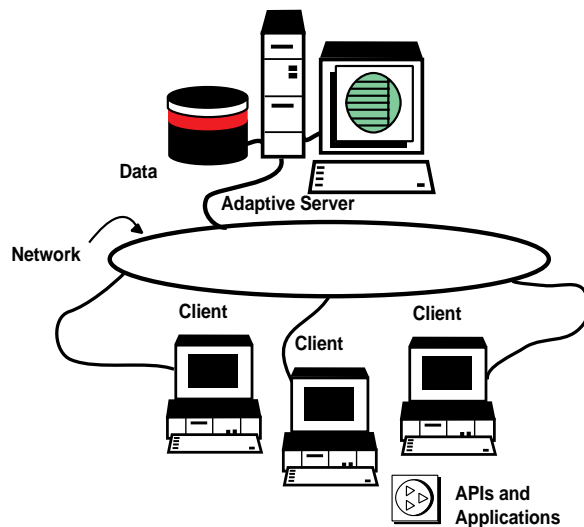


Figure 1-5: A Sybase client/server configuration

About SQL and Transact-SQL

SQL (Structured Query Language) is the standardized database language designed for the RDBMS model, which makes it easy to set up, use, and maintain a relational database. Sybase's enhanced version of SQL is called **Transact-SQL**. SQL looks like English, and is not that difficult to learn.

Sophisticated client tools and applications such as SQL Modeler and InfoMaker and other tools that you may purchase shield you from having to learn Transact-SQL. However, it is a good idea to become familiar with the basic commands. Doing so will help you with every component of Adaptive Server Enterprise for Windows NT; SQL is what ties all the components together. If you learn to use Transact-SQL, you can become a more sophisticated user of your client tools. Many of the more sophisticated tasks require you to work with Transact-SQL commands.

The lessons in this tutorial do not attempt to teach you Transact-SQL. They do, however, include some examples of Transact-SQL code. This is a good opportunity to become familiar with SQL.

About APIs

Applications do not submit SQL queries directly to Adaptive Server. The queries go through an application programming interface (API). APIs are usually vendor-specific; Open Client™, another component of Adaptive Server Enterprise for Windows NT, contains the Sybase APIs: these APIs are Client-Library™, DB-Library™, and CS-Library. Client applications such as InfoMaker, as well as other applications that you purchase or develop, require that these Open Client libraries be installed so that they can communicate with Adaptive Server. For example, you need to install the Open Client software on the machines on which you deploy the applications that you create in InfoMaker.

Open Client also includes character sets, configuration files, network address information, and other items that you need to connect to Adaptive Server.

ODBC (Open Database Connectivity) is the API developed by Microsoft to allow clients to connect to a heterogeneous RDBMS. Some applications that have been developed to be used with more than one vendor's RDBMS use the ODBC interface. SQL Modeler and NetImpact Dynamo access Adaptive Server through the ODBC

interface. You need the Open Client software in addition to the ODBC drivers in order to connect to Adaptive Server.

You will name and configure specific ODBC connections called ODBC data sources for different tasks in the tutorial using a program called ODBC Administrator. Specific instructions are provided where appropriate.

APIs and the Internet

Internet clients, called **Web browsers** in this book, do not need any client APIs installed locally in order to connect to Adaptive Server through the NetImpact Dynamo application server.

What's Next

In the next lesson, "Sybase Central and Adaptive Server System Administration Basics," you will work with Sybase Central, become familiar with some basic system administration tasks, and accomplish some tasks needed to prepare Adaptive Server for the tutorial.

2

Sybase Central and Adaptive Server System Administration Basics

► *Note*

This lesson takes about 25 minutes. Although you can skip other lessons in the tutorial, you need to do this one.

Where You Are

Chapter 1, “Overview,” described the scenario of the dbArts business problem and how you will use Adaptive Server Enterprise for Windows NT to solve that problem.

This chapter is the first lesson in the tutorial.

Sybase Central is a tool that you will use for system administration and for generating and managing the Web application that you will build in a later chapter.

In this lesson, you will see what kind of information Sybase Central displays about Adaptive Server and get a taste of what kind of system administration tasks you can accomplish with this tool. You will also do a few tasks necessary to prepare Adaptive Server for the work you will be doing in this tutorial.

This lesson includes the following projects:

- Project One: Getting Ready for the Tutorial 2-1
- Project Two: Getting Started with Sybase Central 2-3
- Project Three: Preparing Adaptive Server for Development 2-8

Project One: Getting Ready for the Tutorial

Before you begin the tutorial, you need some basic information and to do a few tasks, which are covered in this section:

- Where to Find the Applications and Utilities
- About the Adaptive Server Environment
- Task 1: Find Out About Your Environment
- Task 2: Copy the tutorial Directory from the Product CD
- Using isql to Run Scripts

Where to Find the Applications and Utilities

The applications and utilities mentioned in this tutorial are part of the Sybase and Powersoft® program groups.

- Windows NT 3.51

A program group appears on your desktop in the Program Manager. To start a program, double-click on its icon in the program group

- Windows NT 4.0 and Windows 95

A program group appears in the Start menu. To start a program, select it from the program group in the Start menu.

About the Adaptive Server Environment

Adaptive Server is deployed in a client/server environment, which means that the RDBMS, or database software, resides on a server, a central computer accessible by desktop or workstation computers, called clients, over a network. In most client/server environments, system administrators, database designers, developers, and users work at client machines. The processing power of the server is usually dedicated to doing the work of the database server.

Adaptive Server for Windows NT (Intel) can run on the same operating system and even on the same machine as the client tools and applications, so it is possible to install and run all components on the same NT machine. You can be a network of one, as it were. Even when Adaptive Server is local, however, client applications still must use network protocols to access the server.

In an Adaptive Server Enterprise for Windows NT (Alpha) environment, Adaptive Server is on a separate machine from the client tools and applications included with Adaptive Server.

Task 1: Find Out About Your Environment

Find out about your environment. To do this tutorial, you need to be sitting at the computer where the client utilities and reporting and development tools were installed. The default directories for these tools are `\sybtools` and `\pwrs`. The Open Client software, which contains the libraries, or APIs, needed to communicate with Adaptive Server, also needs to be installed locally (see “About APIs” on page 1-9). The default directory for the libraries is `\sybase\lib`.

You also need to know the name of the Adaptive Server that you will be working with. When you install Adaptive Server, you designate a name by which it will be known to itself and client applications. The default installation value for the Adaptive Server name is the machine name; for example [PIANO]. Your environment may include more than one Adaptive Server. If you have a Adaptive Server installed locally, use that one for this tutorial. You will learn more about names and how to determine whether a server is local in one of the tasks in the first project.

Task 2: Copy the *tutorial* Directory from the Product CD

The scripts and files that you need to do the lessons in this tutorial are located on the product CD in the *tutorial* directory. Copy the entire directory to a local hard disk.

If you downloaded this tutorial from the Web, the *tutorial* directory is in the compressed file that you downloaded. Make a copy of that directory on your local hard disk.

After you download or copy the directory to your local hard disk, change permissions on all files to allow writes.

Running Transact-SQL Scripts

Throughout this tutorial, you will be using SQL scripts, a collection of Transact-SQL commands stored in an ASCII file. These scripts are located in the *tutorial* directory that you copied from the product CD.

You will find instructions on how to run scripts in Appendix A, "Using isql to Run Scripts"

Project Two: Getting Started with Sybase Central

Sybase Central provides online help for database administration issues and wizards for accomplishing Adaptive Server database administration tasks. For more information about Adaptive Server system administration, see the *System Administration Guide* and *Configuring Adaptive Server Enterprise for Windows NT*.

In this project, you will work with Sybase Central to examine some basic aspects of Adaptive Server. This project involves the following tasks:

- Task 1: Start Adaptive Server

- Task 2: Log into Adaptive Server with the “sa” Login
- Task 3: Examine the “sa” Login Properties
- Task 4: Examine the Current Processes
- Task 5: Examine the master Database

About the Sybase Central Interface

Sybase Central adheres to the design of the Windows 95 interface. It displays information about Adaptive Server as a hierarchy of containers and their contents in a window that looks a lot like the Windows 95 Explorer.

Sybase Central provides wizards for accomplishing specific tasks. The wizards are located in the folder or hierarchical container that contains other elements pertaining to the task. For example, the Add Database wizard is in the Databases folder.

When you right-click an item in the Sybase Central interface, a pop-up menu appears. Properties is the last choice in the pop-up menu. If you want information about an item, right-click its icon and choose Properties.

Task 1: Start Adaptive Server

You can start Adaptive Server with Sybase Central when it is installed on the same machine on which Adaptive Server is running.

1. From the Adaptive Server host machine, start Sybase Central, which is located in the Adaptive Server program group.

Available servers appear in the Sybase Central window. Since Adaptive Server Enterprise for Windows NT includes several server products, more than one server may appear for each Adaptive Server name, as shown in Figure 2-1.

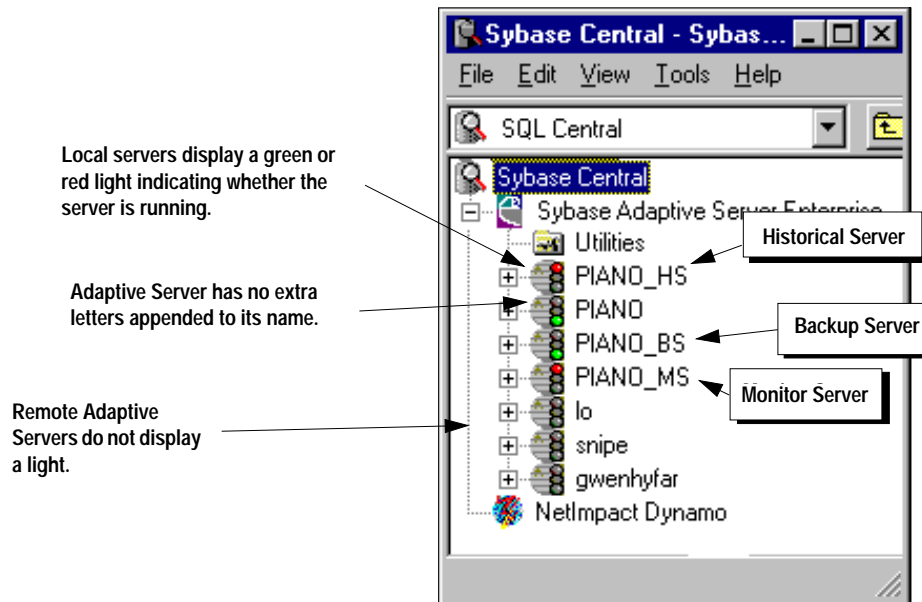


Figure 2-1: Available servers in Sybase Central

2. Select the Adaptive Server you will be working with.

If the server is not running, a dialog box appears asking if you want to start it. Click Yes.

A login dialog box appears; you will log into Adaptive Server in “Task 2: Log into Adaptive Server with the “sa” Login” on page 2-6.

3. From the client machine, start Sybase Central and select the Adaptive Server you selected in step 2.

About the “sa” Login

In order to access Adaptive Server, users need to be added as logins to Adaptive Server. When you first install Adaptive Server, a single login called “sa”, with a NULL password, is automatically configured with System Administrator and System Security Officer roles, which include permission to do high-level tasks such as adding logins, creating databases, and creating storage devices.

Task 2: Log into Adaptive Server with the “sa” Login

If you are doing this tutorial immediately after installation, you will log into the server using the “sa” login, as described here.

If you are doing this tutorial at a later time, consult with the System Administrator for your organization, find out your login and password, and ask to be given the System Administrator and System Security Officer roles. Then follow the directions in this task, using your own login and password instead of the “sa” login and password.

1. In the login box that appeared when you selected or started Adaptive Server in Task 1, step 2, type “sa” as the Login name, and leave the password blank. Or type your login name and password.
2. Click OK.

Figure 2-2 shows details of Adaptive Server as displayed in the Sybase Central window in a hierarchical format. Double-click any icon, or click the plus sign to the left of the icon, to go down a level in the hierarchy.

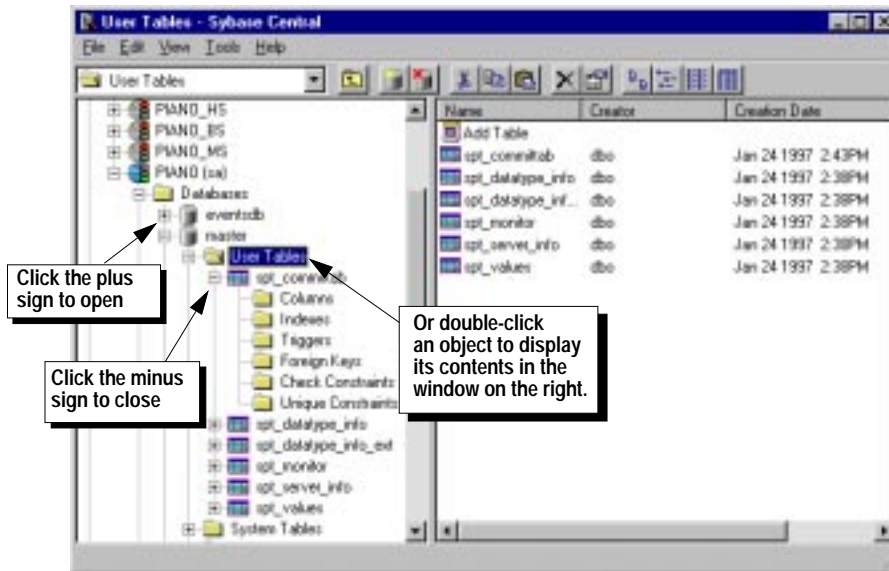


Figure 2-2: Adaptive Server information displayed in Sybase Central

Task 3: Examine the “sa” Login Properties

You can display information about logins. Information about logins includes things like the default database and the roles assigned to the login.

1. Double-click the Logins folder in the Sybase Central window on the right side of the screen.
2. Select the “sa” login, and display its properties (right-click and choose Properties from the pop-up menu).
3. Look at the information displayed on each tab.
4. Click Cancel.

Task 4: Examine the Current Processes

A process is a task that is being executed by Adaptive Server. A process can be initiated by a user giving a command or by Adaptive Server itself. Each process is assigned a unique process identification number when it starts. View the current Adaptive Server processes.

1. Click the Processes folder in the Sybase Central window on the left side of the screen.
2. Look at the current processes.

In the list you will see a process for Sybase Central and the “sa” login. That’s you. If you are the only person logged into Adaptive Server, the other processes are system processes.

Task 5: Examine the *master* Database

The *master* database is a system database that controls the operation of Adaptive Server as a whole and stores system information about all user databases.

1. Click the Databases folder.
2. Double-click the *master* database icon in the Sybase Central window on the right side of the screen.
3. Double-click the System Tables folder.

A list of tables appears. Each table name begins with the letters “sys.” This prefix indicates that the tables are system tables. If the tables do not begin with the letters “sys,” double-click the System Tables folder again.

4. Close all open folders by clicking the minus sign that appears to the left of each folder (see Figure 2-2 on page 2-6).

Project Three: Preparing Adaptive Server for Development

In order to complete the lessons that follow this one, you need to accomplish a few system administration tasks. They are:

- Task 1: Create the “student” Login for Use in This Tutorial
- Task 2: Give the New Login Roles
- Task 3: Log In As “student”
- Task 4: Create a Device for the Events Database
- Task 5: Examine the size of tempdb
- Task 6: Extend tempdb to the master Device (Optional)
- Task 7: Create a New Device and Extend tempdb (Optional)
- Task 8: Set the Database Option truncate log on checkpoint

Task 1: Create the “student” Login for Use in This Tutorial

You will use a special login for the tutorial that will give you powerful permissions. In the final lesson of the tutorial, you will drop the login with a script.

1. In the window on the left, select your server, if it is not already selected.
2. Double-click the Logins folder in the Sybase Central window on the right.
3. Double-click Add Login.
4. Type “student” as the login name.
5. Type a password of your choice, and confirm it.
6. Click Next.
7. Select *master* as the default database for the login.
8. Click Next and Finish.

Task 2: Give the New Login Roles

1. Display Properties for the “student” login (right-click, and choose Properties from the pop-up menu).
2. Select the Roles tab.
3. Select the System Administrator, System Security Officer, and Operator roles for the new login.
4. Click OK.

Task 3: Log In As “student”

1. In the Sybase Central menu, choose Tools→Disconnect.
2. Select the current connection, which appears in the list as your server name followed by “(sa)”.
3. Click Disconnect.
4. Connect again by selecting your server.
5. Type the “student” login and password.
6. Click OK.

Your server icon now appears with “(student)” after its name.

Task 4: Create a Device for the Events Database

You need to create storage space for the events database that you will build in the next lesson. An Adaptive Server database is stored on a **database device**. In Windows NT, a database device is usually an operating system file that you initialize to store Adaptive Server data.

You will initialize a device called *tutorial_dev* with enough room for the events database. In the final lesson of the tutorial, you will drop the device.

1. Double-click the Database Devices folder.
2. Double-click Add Database Device.
3. Type “tutorial_dev” as the device name.

tutorial_dev is the logical name of the device, which is how the device is referenced within Adaptive Server.

The physical name of the device is the operating system name of the file to be initialized.

4. Designate a path and file name for the physical name:
 - If your Adaptive Server is local, click the Browse button to navigate to the data folder in the *sybase* directory. Type “tutor.dat” as the file name, and click Open.

Or

- If your Adaptive Server is remote, type “C:\sybase\tutor.dat” or “D:\sybase\tutor.dat” as the file name, where “C” or “D” is the disk drive on which the *sybase* directory is installed.
5. Click Next.
6. Accept the default device number.
7. Set the device size as 6MB, as shown in Figure 2-3. Do not fill in the Starting Address box.

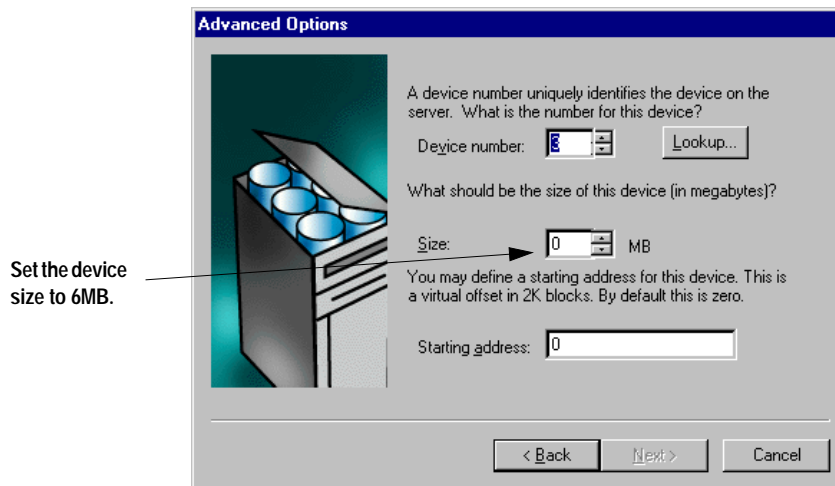


Figure 2-3: Setting the device size

8. Click Next.
9. Accept all default settings and click Next again.
10. Click Finish.

Task 5: Examine the size of *tempdb*

Adaptive Server includes a temporary database called *tempdb*. It provides a storage area for temporary tables and other temporary

working storage needs. Some client applications such as NetImpact Dynamo can generate a lot of activity in the temporary database. The default size for *tempdb* at installation is 2MB. It is a good idea to increase the size of *tempdb* to allow for the increased activity.

1. Click the Databases folder.
2. Display Properties for *tempdb* (right-click, and choose Properties).
3. Select the Devices tab.

The tab displays storage information about the database. It shows all the database devices on which the database is stored and the amount of storage space on each device dedicated to the database.

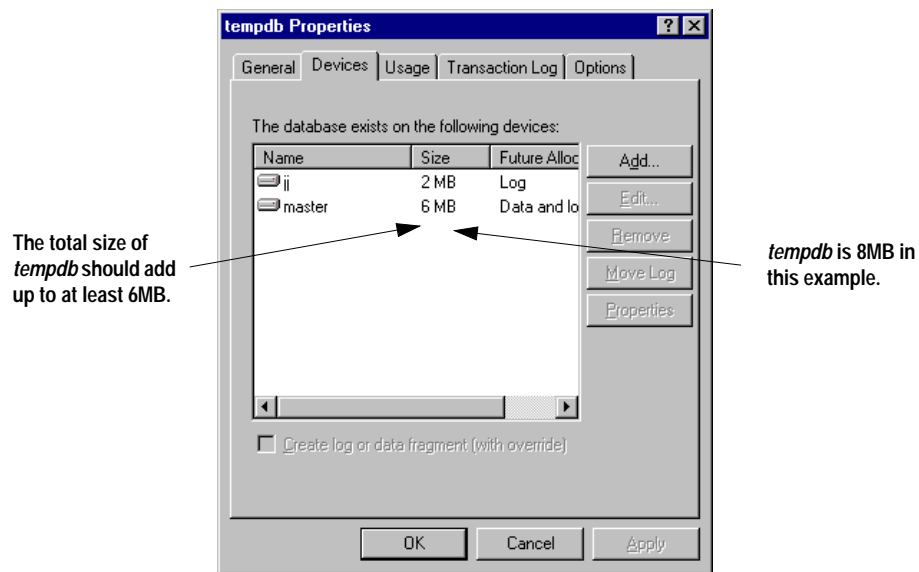


Figure 2-4: Examining the size of *tempdb*

4. Click OK.

If the total size of the dedicated space on all devices is less than 6MB, you need to increase the total size to 6MB. Note the amount by which you need to increase the device, and follow the instructions in the next task, "Task 6: Extend tempdb to the master Device (Optional)". If the total size of dedicated space is

already 6MB or more, click Cancel and skip to “Task 8: Set the Database Option truncate log on checkpoint” on page 2-14.

Task 6: Extend *tempdb* to the *master* Device (Optional)

In this task, you will look for available space on the *master* device. If there is room, you will extend *tempdb* to the master device, by the appropriate amount. If there is not enough room, you will create a new device and extend *tempdb* to that device. Do not use *tutorial_dev*, because you will be dropping that device at the end of the tutorial.

1. Double-click the Database Devices folder.

Device names and statistics appear in the Sybase Central window on the right.

2. Note the size of the master device and whether there is any free space.

If, during installation, you set the size of the master device at 30MB, Sybase Central would display a size of 30 MB and might display 15MB unused. If you need to extend *tempdb* by 4MB, there is more than enough space. If there is not enough space to extend *tempdb*, skip to “Task 7: Create a New Device and Extend tempdb (Optional)”.

3. Click the Databases folder.
4. Display Properties for *tempdb* (right-click, and choose Properties).
5. Select the Devices tab.
6. Select the master device icon.
7. Click Edit to extend *tempdb* to the master device.
8. Set the amount by which you want to extend the master device—enough to make the total size of *tempdb* 6MB— if it is 2 now, set it to 4.
9. Click OK to close each window.
10. Skip to “Task 8: Set the Database Option truncate log on checkpoint” on page 2-14.

Task 7: Create a New Device and Extend *tempdb* (Optional)

Complete this task only if *tempdb* is less than 6MB, and then only if there is not enough room on the master device to increase the size of *tempdb*.

1. Double-click the Database Devices folder.
2. Double-click Add Database Device.
3. Type “tempdb_dev” as the device name.
4. Designate a path and file name for the physical name:
 - If your Adaptive Server is local, click the Browse button to navigate to the data folder in the *sybase* directory. Type “tempdb.dat” as the file name, and click Open.
- Or
- If your Adaptive Server is remote, type “C:\sybase\tempdb.dat” or “D:\sybase\tutor.dat” as the file name, where “C” or “D” is the disk drive on which the *sybase* directory is installed.
5. Click Next.
6. Accept the default device number.
7. Set the device size as 4MB, or the amount necessary for increasing the size of *tempdb*.
8. Click Next.
9. Click Next again.
10. Click Finish.
11. Click the Databases folder.
12. Display Properties for *tempdb* (right-click, and choose Properties).
13. Select the Devices tab.
14. Click Add to extend *tempdb* to the *tempdb_dev* device.
15. Select *tempdb_dev*.
16. Select the Data option button.
17. Set the amount by which you want to extend the *tempdb_dev* device—enough to make the total size of *tempdb* 6MB—if it is 2 now, set it to 4.
18. Click OK to close each window.

Task 8: Set the Database Option *truncate log on checkpoint*

You can further prevent running out of space in *tempdb* by turning on the database option *truncate log on checkpoint*. When this option is on, the transaction log, a table in which all changes to the database are recorded, is truncated frequently. If this option is turned off, development work can quickly overload the database. In the final lesson of the tutorial, you will turn this option off using a script.

1. Display Properties for *tempdb*.
2. Select Options.
3. Select Truncate Log on Checkpoint.
4. Click OK.
5. Right-click *tempdb*, and choose Checkpoint.
6. Keep Sybase Central open on your desktop while you are doing the rest of the lessons in the tutorial.

What's Next

In the next lesson, "Designing the Events Database with SQL Modeler," you will design and generate the *eventsdb* database using SQL Modeler.

3

Designing the Events Database with SQL Modeler

► **Note**

This lesson takes about 30 minutes. Complete the lesson in Chapter 2, “Sybase Central and Adaptive Server System Administration Basics,” before doing this one.

Where You Are

In Chapter 2, “Sybase Central and Adaptive Server System Administration Basics,” you worked with Sybase Central and completed a few system administration tasks that prepared your system for the tutorial activity.

In this lesson, you will design a conceptual model of the events database, and then you will use SQL Modeler to create the physical implementation and generate the events database.

This lesson includes the following concepts and projects:

- Project One: Designing the Events Database 3-1
- Project Two: Getting Started with SQL Modeler 3-8
- Project Three: Working with the Predefined eventsdb.PDM 3-11
- Project Four: Generating the Database 3-18

Project One: Designing the Events Database

With SQL Modeler, you develop the logical, or conceptual, model of your database first, before you use the tool. SQL Modeler helps you develop a physical implementation of a conceptual database design. The Adaptive Server implementation of the events database will be called *eventsdb*.

Since it is conceptual, this project does not include any hands-on tasks. However, it does walk you through the logical design of the events database and covers these relational database concepts:

- Tables, columns, and rows
- Relationships and keys
- Relationships and data retrieval

- Business rules
- Indexes

Tables, Columns, and Rows

In a relational database, data is represented in two-dimensional **tables**. A table contains information about one kind of entity. The basic entity that you are tracking is an event. A row, or record, describes one instance of that entity—a specific event. A column, or field, describes an attribute of that entity, such as where it is held, how much it costs, or what type of music is being featured.

For the musical events database, you have a few entities in addition to the event entity:

- Musical events
- Clubs and venues that sponsor the events
- Categories of musical styles
- Artists

The dbArts organization has decided to track only the first three entities:

- Event
- Club
- Category

For the entities in the database, you will track the following attributes:

Table 3-1: Entities and Attributes in *eventsdb*

Event	Club	Category
Artist	Name	Type (abbreviation of category name)
Club	Address	Descrip (full category name)
Date	City	
Price	Phone	
Attendance	Capacity	
Category		

A relational database consists of a collection of interrelated tables. In a physical database design, an entity takes the form of a table, and an attribute takes the form of a column. The idea is to store data about separate entities in separate tables, with a column for each attribute. Then, you define the relationships between the tables. The *eventsdb* database will consist of three tables that correspond to the three entities and several attributes defined above and listed in Table 3-1.

One thing you might notice is that the tables in *eventsdb* are interrelated; two of the entities are actually columns of another table. *club* and *category* which are tables themselves, are both columns of the *event* table. The next step in designing the database consists of defining how the tables are interrelated.

Relationships and Keys

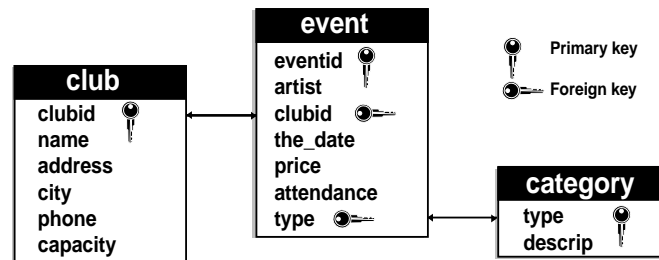
Tables in a relational database are related to each other when the same column appears in more than one table. Columns that appear in more than one table and that link the tables together are called **keys**.

A **primary key** uniquely identifies a row in a table. A **foreign key** relates a row in one table to a row in another table by matching its value to the value of the other table's primary key.

Each table usually needs a primary key; a primary key column may or may not already exist as a column in the table. You probably need to create a new primary key column for the *club* table, because there is not a single column for which the value would be unique for each row. For example, there might be two clubs with the same name. For the *club* and *event* tables, you will create identification columns called *clubid* and *eventid* that will hold a unique integer for each row. The *category* table already has such a column: the *type* (abbreviation) column will be unique for each type of music.

Each of the key columns can also be used in the related table to define the relationships between them. For example, in the *event* table, you will replace the generic *club* column with the new key column, *clubid*. In the *club* table, *clubid* is the primary key; in the *event* table, *clubid* is a foreign key.

Figure 3-1 illustrates the tables and their relationships.

Figure 3-1: The *eventsdb* database model

Relationships and Data Retrieval

Understanding relationships between tables is necessary when defining queries, which is how you request data from Adaptive Server. This concept will be important in the upcoming chapters, but it is easiest to understand here, in the context of relationships.

You retrieve data from Adaptive Server with the Transact-SQL `select` command. You select specific rows and columns from a table. For example, you can request information about cities and phone numbers for clubs in the city of Berkeley with the following SQL `select` statement:

```
select city, phone from club
where city = "Berkeley"
```

Club ID	Club Name	Club Address	City	Phone	Club Capacity
1	El Flanker	45 Hegenberger Loop	Oakland	(510) 562-2679	80
2	Fandango Latino	9829 San Leandro St.	Oakland	(510) 638-4730	95
3	Kimball's Carnaval	5800 Shellmound St.	Emeryville	(510) 653-5300	150
4	Mexicali Rose	29097 Mission Blvd.	Hayward	(510) 889-1048	94
5	Club Kaos	39148 State St.	Fremont	(510) 792-6500	100
6	Black Diamond Brewing Company	2330 N. Main St.	Walnut Creek	(510) 933-4271	250
7	Brennan's	4th and University	Berkeley	(510) 841-0960	125
8	Starry Plough	3101 Shattuck Ave.	Berkeley	(510) 841-2082	90
9	Tr's	2001 Salvio St.	Concord	(510) 841-2082	75

Figure 3-2: Selecting columns and rows from the club table

In order to retrieve data that spans two tables, you must specify the relationships between the tables. A `select` statement that specifies the relationships between two tables is called a **join**. For example, you might want to find out which artists play a particular kind of music. You could submit the following SQL `select` statement:

```

select artist, descrip
  from event, category
 where event.type = category.type
    and type = "sal"

```

Figure 3-3 illustrates the join required for this select statement.

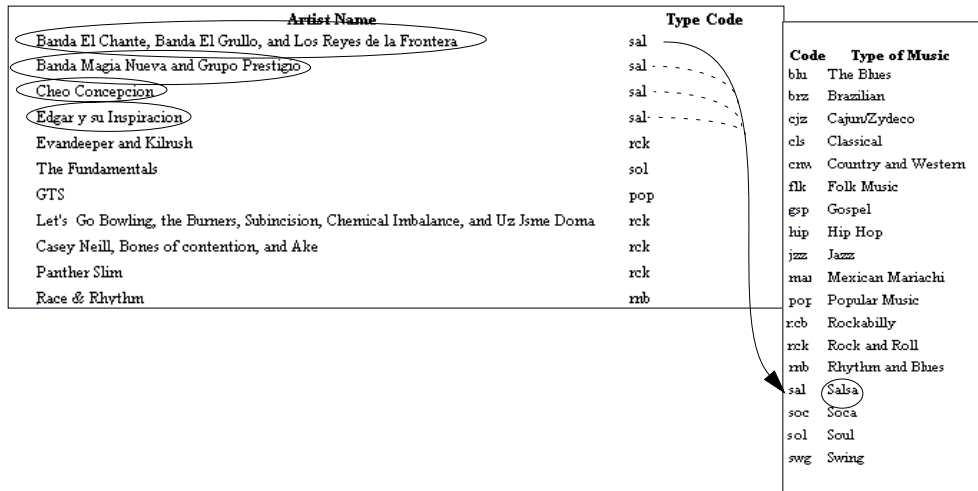


Figure 3-3: A join

Without specifying the join across two tables, Adaptive Server returns what is called a Cartesian product—all possible combinations of rows from each of the two tables.

Business Rules

Another step in designing a database can be to define the business rules to which the database must adhere. For simplicity's sake, you have pared down dbArts's business rules to the following four rules:

1. Every event must be associated with only one club.
2. Every event must be associated with only one type of music.
3. The city must be one of the following East Bay cities:
 - Alameda
 - Albany
 - Berkeley

- Concord
 - Emeryville
 - Hayward
 - Kensington
 - Pleasanton
 - Point Richmond
 - Richmond
 - San Ramon
 - Walnut Creek
4. The combination of name and city in the *club* table must be unique—there cannot be two clubs of the same name in one city. Although the primary key column uniquely identifies the club, the key is an integer, so you may accidentally enter data about the same club twice. Such a rule helps prevent such mistakes in data entry.

Enforcing Business Rules

In a relational database such as Adaptive Server, there are many ways to enforce business rules. You will use a few of them in this lesson:

- Referential integrity
- Domain checking (check constraints)
- Unique constraints

Referential Integrity

Enforcing the first two rules involves referential integrity, which enforces the rules governing the relationships between primary keys and foreign keys. You will define referential integrity constraints to require that Adaptive Server check that values for the *clubid* and *type* columns in the *event* table have matching values in the related tables. Enforcing referential integrity in this way is sometimes called **declarative referential constraints**.

Another way to enforce referential integrity is by using triggers. A **trigger** is a special set of Transact-SQL commands that goes into a effect when a user makes a change on a table. The distinction between the two ways of enforcing referential integrity will be important when you work with SQL Modeler—you will be asked to

choose between the two. For detailed information on triggers, see the *Transact-SQL User's Guide*.

Domain Checking (Check Constraints)

Enforcing the third rule involves checking that data inserted into a specific column match a list of allowable values, or a **domain**.

You can enforce this rule in either Adaptive Server or a client application. When enforced on Adaptive Server, this kind of checking is called **check constraints**. Rules that are not incorporated into the database design can be built into the client program and involve what is called client-side (as opposed to server-side) checking. When enforced in InfoMaker, this kind of checking is called **domain checking**. You will use client-side domain checking for the third business rule (see "Task 1: Examine a Validation Rule" on page 4-13).

Unique Constraints

Enforcing the fourth rule involves defining a unique constraint. You will define a composite unique constraint on the *name* and *city* columns to require that Adaptive Server check on the combination of the two columns for uniqueness.

Indexes

An **index** helps Adaptive Server locate data. Indexes speed up data retrieval by pointing Adaptive Server to the location of a table column's data on disk.

When you define a column as a primary key, Adaptive Server automatically creates an index on that column. Adaptive Server also automatically creates an index for the columns on which you place a unique constraint. In addition, it is a good idea to define an index on a column that is often searched in sorted order. You will define an index on the *event* table's *the_date* column because you are going to be creating calendars, which will require Adaptive Server to scan the column in sorted order.

Indexes are an important design consideration for optimizing Adaptive Server performance. For an in-depth discussion of the performance implications of indexes, see the *Performance and Tuning Guide*.

Project Two: Getting Started with SQL Modeler

SQL Modeler helps you make a physical data model (PDM) for a database. A PDM is a vendor-specific version of a logical database design. SQL Modeler creates Sybase-specific PDMs.

Instead of creating the database using a series of Transact-SQL commands, you use SQL Modeler's graphic user interface to build the database. When you are finished with the design phase, you instruct SQL Modeler to build the database on Adaptive Server. SQL Modeler constructs and issues the Sybase-specific commands to build the database.

This project involves the following tasks:

- Task 1: Start SQL Modeler
- Task 2: Configure an ODBC Data Source for Database Creation
- Task 3: Create the category Table

Task 1: Start SQL Modeler

Start SQL Modeler for Adaptive Server. When you first start SQL Modeler, you have a blank PDM workspace and a new PDM file.

You use the tools from the tool palette to create new objects such as tables in the PDM workspace. You will work with the Pointer tool, the Table tool, the View tool, and the Reference tool.

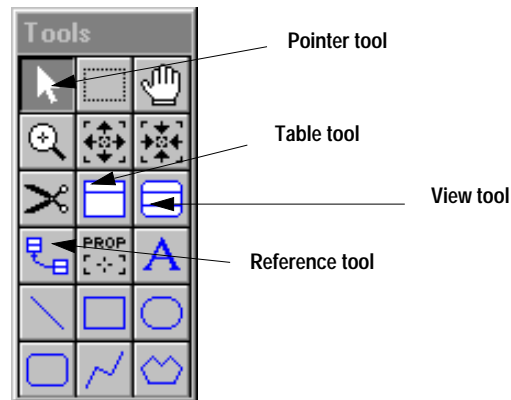


Figure 3-4: The tool palette

Task 2: Configure an ODBC Data Source for Database Creation

SQL Modeler connects to Adaptive Server through the ODBC interface. In this task, you will define an ODBC data source for use by SQL Modeler. (For more information on the ODBC interface, see “About APIs” on page 1-9.)

You will not actually connect to Adaptive Server until you generate the database (see “Project Four: Generating the Database” on page 3-18).

1. Choose Database→Configure Database.
2. Click Add.
3. Select the Sybase System 11 ODBC Driver.
4. Click OK.
5. Type “create_events” as the Data Source name.
6. Type the name of your Adaptive Server.
7. Type “master” as the Database Name.
8. Click Connection.
9. Type “student” as the Default Logon ID.
10. Click Performance.
11. Check for the following two settings:
 - Prepare Method should be 2-Full.
 - Select Method should be 1-Direct.
12. Click OK.
13. Click Close.

Task 3: Create the *category* Table

In this task, you will use the Table tool to create a table.

1. Click the Table tool in the tool palette.



2. Click anywhere in the PDM workspace.

The Table symbol appears.

- Click the Pointer tool in the tool palette.



- Double-click the new table symbol.

The Table Properties dialog box appears. This is where you define the table.

- Type “Category” in the Name box and in the Code box.
- Click the Columns button.

The Columns Definition dialog box appears.

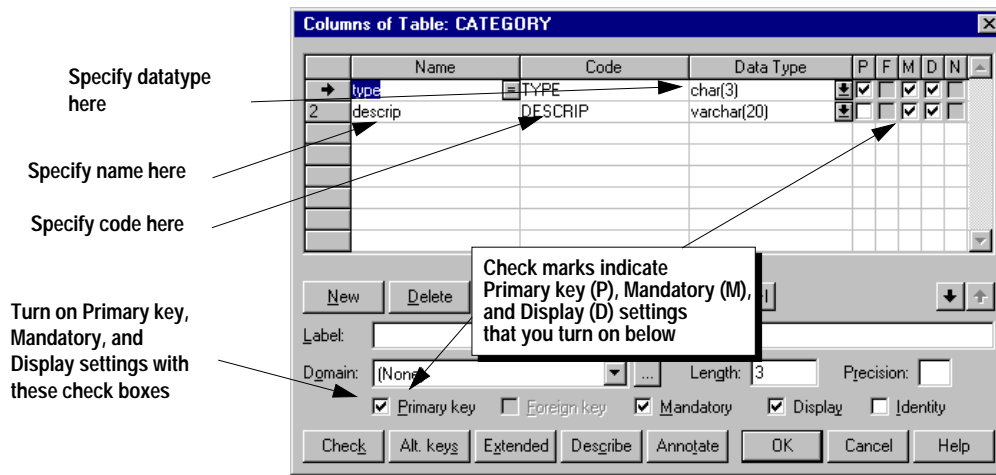


Figure 3-5: Columns Definition dialog box

- Enter the column information shown in Table 3-2. As shown in Figure 3-5, specify the name, code, and datatype in the appropriate column, and turn on the Primary key, Mandatory, and Display settings by clicking the appropriate check boxes at the bottom of the screen.

Table 3-2: Columns in the category table

Name	Code	Datatype	Settings
<i>type</i>	Type	<i>char(3)</i>	Primary key Mandatory Display

Table 3-2: Columns in the category table (continued)

Name	Code	Datatype	Settings
<i>descrip</i>	Descrip	<i>varchar(20)</i>	Mandatory Display

- Click OK in each dialog box.

The table appears in the workspace with the appropriate name and columns.

- Choose File→Close.

Do not save any changes.

Project Three: Working with the Predefined *eventsdb.PDM*

Instead of defining all the database objects this way, you can open a predefined PDM file called *eventsdb.PDM*. You will open this file, examine some of the predefined objects, and create new ones.

This project involves the following tasks:

- Task 1: Open *eventsdb.PDM*
- Task 2: Examine the Club Table and Work with a Business Rule
- Task 3: Examine the Model Properties
- Task 4: Examine the Business Rules
- Task 5: Define the References
- Task 6: Associate the Reference with Business Rules
- Task 7: Create the Date Index
- Task 8: Create the *artist_club_date* View

Task 1: Open *eventsdb.PDM*

- Choose File→Open PDM.
- Navigate to the *tutorial* subdirectory, and select the file *eventsdb.PDM*.
- Click OK.

When you try to open the file, SQL Modeler asks if you want it to generate default references between tables. Click No. You will do that in a later task, “Task 5: Define the References” on page 3-14.

This *eventsdb.PDM* file contains most of the completed database design.

Task 2: Examine the Club Table and Work with a Business Rule

1. Select the Pointer tool.



2. Double-click the table *club*.
 - Notice that a label has been defined in the label box.
 - The primary key is associated with a constraint, named *clubid_constraint*.

3. Click the Indexes button.

A composite unique index named *city_name_unique_index* has been defined for the table.

4. Click Rules to associate a business rule with the index.
5. Click Add to add a rule.
6. Select Rule 2, clubname_city rule.
7. Click OK.

The rule name and text of the rule appears in the *city_name_unique_index* rules box.

8. Close each box by clicking OK.

Task 3: Examine the Model Properties

An object in the workspace lists the name of the project and the model. This is called the Title box.

Physical Data Model		
Project : the events database		
Model : eventsdb		
Author :	Version:	2/18/97

Figure 3-6: Title box

1. Select the Pointer tool.
2. Double-click the Title box.
The Model Property Sheet appears.
3. Type your name in the Author box.
4. Click Options.
5. Select the second item in the list, as shown in Figure 3-7:



Figure 3-7: Eventsdb options

The Use check box has been selected for this item, which defines the size and storage device of the database. Six megabytes will be allocated for *eventsdb* on *tutorial_dev*, the device you created in “Task 4: Create a Device for the Events Database” on page 2-9.

6. Scroll through the text in the Other Options list box. This text is the Transact-SQL command to turn on the database option `truncate log on checkpoint`. In the previous lesson, you used Sybase Central to set this option for *tempdb* (see “Task 8: Set the Database Option truncate log on checkpoint” on page 2-14). This option is good for a development environment. When you put a database into production, it is a good idea to turn this option off.
7. Close each dialog box by clicking OK.

Task 4: Examine the Business Rules

SQL Modeler allows you to catalog your business rules. When you associate a rule with any part of the database design, a check appears next to the rule in the list. This helps you remember to take all rules into account.

1. Choose Dictionary→List of Business Rules.

All four of the rules defined in “Business Rules” on page 3-5 appear in the list. Only the first two have been checked off. Rules 3 and 4 have not been checked off, because you have not yet defined any referential constraints.

“E” means the rule is associated with an expression, which can be implemented by Adaptive Server or by a client application. An expression is a mathematical or logical statement that performs a computing action, and that, in this case, will fulfill the business rule requirement. The third business rule shows an “E” because it is associated with a client-side expression (see “Domain Checking (Check Constraints)” on page 3-7 and “Task 1: Examine a Validation Rule” on page 4-13).

“U” means the rule is associated with an object in the PDM.

You will define referential constraints in the next task.

2. Close the dialog box by clicking Cancel.

Task 5: Define the References

SQL Modeler makes it easy to define references between tables. You define references graphically with the Reference tool.

The foreign key columns in the *event* table, *clubid* and *type*, do not currently appear in the *event* table. When you draw a reference between two tables, you can **automatically migrate** primary key columns to foreign key columns, which means you duplicate the column definition of the column designated as the primary key in the first table to a new column in the second table, which is designated a foreign key.

In this task you will automatically migrate the columns defined as primary keys in the *club* and *category* tables to foreign keys in the *event* table.

1. Choose File→Display Preferences.
2. Select the Join check box, if it is not already selected.

3. Click OK.
4. Choose File→Model Options.
5. Select the Auto-Migrate FK check box, if it is not already selected.
6. Click OK.
7. Select the Reference tool.



8. Click anywhere in the *event* table.
9. Drag the reference to the *club* table.

The *clubid* column now appears in the *event* table, and an arrow links the two tables. This is the symbol for the reference. The reference labels indicates the join expression:

`clubid = clubid`

10. Repeat the process for the *event* and *category* tables, beginning again with the *event* table. The *type* column now appears in the *event* table, and an arrow links the two tables. SQL Modeler displays the following join expression for the reference:
`type = type`
11. Right-click in the workspace to release the Reference tool.

Task 6: Associate the Reference with Business Rules

The references you created in the last task satisfy the two remaining dbArts business rules. In this task, you will associate the references with business rules.

1. Select the Pointer tool.
2. Double-click the `clubid = clubid` reference.
3. Click Rules.
4. Click Add.
5. Select `event_club` rule.
6. Click OK.
7. Click OK in each of the dialog boxes.
8. Repeat the process for the `type = type` reference, choosing `event_type` rule where appropriate.

9. To double-check the List of Business Rules, choose Dictionary→List of Business Rules.
All the rules should now be checked off.
10. Click OK in each of the dialog boxes.

Task 7: Create the Date Index

The *city_name_unique_index* has already been defined. You need to define an index on the *event* table's *the_date* column, which you will do in this task.

1. Select the Pointer tool. Double-click the *event* table.
2. Click the Indexes button.
3. Type "date_index" as the name of the index.
4. Click the Add button to designate the column being indexed.
5. Select *the_date*.
6. Close each window by clicking OK.

Task 8: Create the *artist_club_date* View

A **view** is a virtual table, composed of specific columns and/or rows from one or more tables or other views. A view is virtual because it has no physical existence in the database. You define a view as a Transact-SQL select statement.

You are going to create a view called *artist_club_date* as a SQL statement with a join that includes all three tables. This view will make it easier to create calendars.

1. Select the View tool.



2. Click anywhere in the PDM workspace.
A graphical representation of a view appears.
3. Select the Pointer tool. Double-click the new view object.
4. Type the name "artist_club_date" in both the Name and Code boxes.
5. Type "A view for making calendars" in the Label box.

6. Click the Tables button to choose the tables for constructing the view.
7. Click Add All to select all three tables.
All three tables plus the two references appear in the Selected objects list.
8. Click OK.
9. Click the Query button to define the query.
10. Deselect all the columns in the Available Columns list.

Figure 3-8 shows the window you use to define queries.

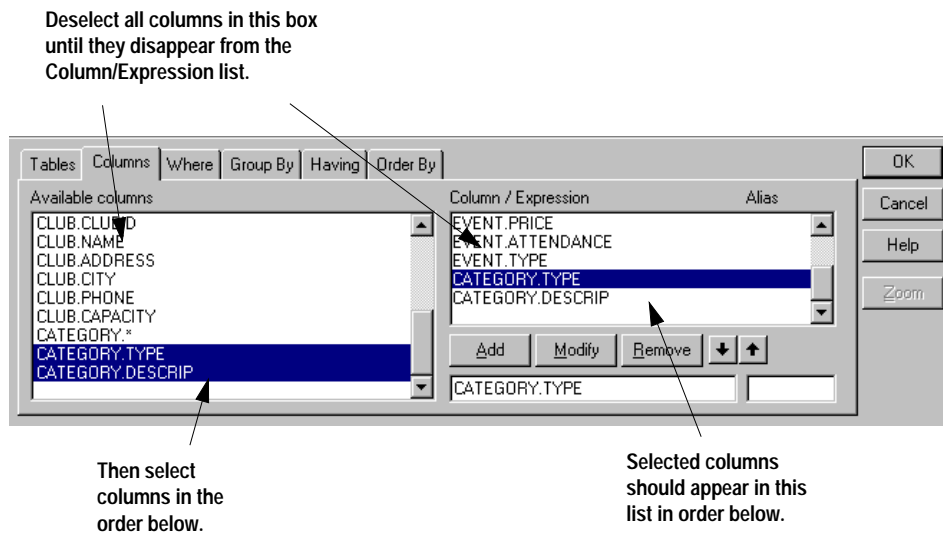


Figure 3-8: Selecting columns for the view

11. Select the following columns, in this order:

- *event.artist*
- *event.clubid*
- *club.name*
- *event.the_date*
- *category.type*

The selected columns appear in the Column/Expression list.

SQL Modeler constructs a SQL statement, based on the columns you selected and on how you have defined the references. If you like, you can look at the actual SQL statement generated by SQL Modeler.

12. Choose Query→Show SQL.

A SQL statement similar to the following appears:

```
select EVENT.ARTIST, EVENT.CLUBID, CLUB.NAME, EVENT.THE_DATE,
CATEGORY.TYPE
from CLUB, CATEGORY, EVENT
where CLUB.CLUBID = EVENT.CLUBID
and CATEGORY.TYPE = EVENT.TYPE
```

13. Close both windows by clicking OK.

Project Four: Generating the Database

After you define the PDM for your database, you instruct SQL Modeler to generate the actual database in Adaptive Server.

You can have SQL Modeler generate a SQL script that can be run later, or you can immediately execute the commands in Adaptive Server through the ODBC interface.

First, you will generate a script. Before you generate the script, you will close the current PDM and open another PDM that contains all the information you need to describe the database.

This project involves the following tasks:

- Task 1: Open *after.PDM*
- Task 2: Generate a Script
- Task 3: Generate the Database
- Task 4: Generate the Stored Procedures
- Task 5: Save the PDM File

Task 1: Open *after.PDM*

1. Choose File→Close.
Do not save any changes.
2. Choose File→Open PDM.
3. Navigate to the *tutorial* directory, and select the file *after.PDM*.

4. Click OK.
5. Click OK in the Messages box.

Task 2: Generate a Script

1. Choose Database→Generate Database.
The Database Generation dialog box opens to the Schema page.
2. Select the following settings on the Schema page; deselect any settings not listed.

Table 3-3: Database generation settings on Schema page

Setting	Purpose
Create table	Creates a table.
Primary key	Creates a primary key.
Foreign key	Creates a foreign key.
Decl. integrity	Enforces referential integrity through declarative referential constraints rather than triggers (see “Referential Integrity” on page 3-6).
Other indexes	Creates an index.
Create view	Creates a view.

3. Click the Database page tab, and select the following settings on the Database Page. Deselect any settings not listed.

Table 3-4: Database generation settings on Database page

Setting	Purpose
Create database	Creates a database.
Physical options	Executes commands that allocate storage on <i>tutorial_dev</i> .
Open database	Executes the command that allows tables to be created in the new database.

4. Click Generate Script.
If you are using Adaptive Server for the first time, Adaptive Server asks if you want to create the *script* directory. Click Yes.

SQL Modeler generates the script and asks you if you want to view it. Click Yes.

5. Scroll through the script to familiarize yourself with the Transact-SQL commands.
6. Close the script.

Task 3: Generate the Database

Instead of running the script later, you can have SQL Modeler generate the database directly. When you instruct SQL Modeler to generate the database, you must connect to Adaptive Server.

1. Choose Database→Generate Database.
2. Click Create Database.

SQL Modeler displays a message asking if you want to replace the current script file.

3. Click Yes.

The Connect dialog box appears.

4. If it is not already selected, select *create_events* from the list of defined data sources.
5. Type the “student” user name and password.
6. Click Connect; then click OK.

SQL Modeler generates the database and gives you a message if the creation was successful. The status of the generation appears in the Messages console.

7. Scroll through the Messages console.
8. Click OK.

Task 4: Generate the Stored Procedures

The model includes a stored procedure. A stored procedure is a set of one or more commands stored in a database. A stored procedure is partially processed before it is stored, so it executes faster than it would if you executed its constituent commands individually. Scripts that you will run in later lessons call stored procedures to prepare the database for a specific lesson.

SQL Modeler generates procedures in a separate step from database generation.

1. Choose Database→Generate Triggers and Procedures.
2. Under Orders to Be Generated, select Create Procedure, and clear Create Trigger.
3. Clear Open Database, because you opened the database when you generated the database.
4. Click Create Triggers to generate all selected triggers and stored procedures instead of merely generating a script.
The Connect dialog box appears.
5. Type the “student” user name and password, and click Connect.
SQL Modeler generates the procedures and gives you a message if the creation was successful. Status appears in the Messages console.
6. Click OK.

Task 5: Save the PDM File

1. Choose File→Save As.
2. Type “events2.PDM” as the file name.
3. Click OK.
4. Exit SQL Modeler.

What's Next

In the next lesson, “Entering and Importing Data with InfoMaker,” you will learn how to connect InfoMaker to Adaptive Server and how to use InfoMaker to populate a database.

4

Entering and Importing Data with InfoMaker

► **Note**

This lesson takes about 30 minutes.

If you skipped the last chapter, run the script `design.sql` located in the *tutorial* directory. For information on running scripts, see Appendix A, “Using isql to Run Scripts.”

If you plan to do the lesson in the next chapter, Chapter 5, “Creating Reports and Applications with InfoMaker,” you need to complete all the activities in the first project in this chapter, “Project One: Connecting InfoMaker to Adaptive Server”. You can skip the other projects.

Where You Are

In the previous lessons, you worked with Sybase Central and accomplished a few system administration tasks that prepared your system for the tutorial activity. Then, you created a physical data model using SQL Modeler and generated the *eventsdb* database in Adaptive Server.

In this lesson, you will use InfoMaker to put data into a database, whether through direct or form-based data entry, importing text from a file, or piping in data from a foreign database.

This lesson includes the following projects:

- Project One: Connecting InfoMaker to Adaptive Server 4-1
- Project Two: Entering and Importing Data into the eventsdb Database 4-6
- Project Three: Piping Data from Another Database 4-10
- Project Four: Defining Extended Attributes 4-13
- Project Five: Creating Forms 4-15

Project One: Connecting InfoMaker to Adaptive Server

Setting up InfoMaker for this tutorial requires verifying installation of stored procedures, creating a database profile, and populating InfoMaker-specific tables with a script.

This project involves the following tasks:

- Task 1: Verify Installation of or Install the Powersoft Stored Procedures
- Task 2: Create a Database Profile
- Task 3: Examine the Powersoft Repository Tables in Sybase Central
- Task 4: Run the Scripts to Populate the Powersoft Repository

Connectivity Overview

InfoMaker sends calls to Adaptive Server through a Powersoft library, also known as a **Dynamic Link Library (DLL)**. It is called *pbsync050.dll*, and it makes calls to the Open Client Client-Library.

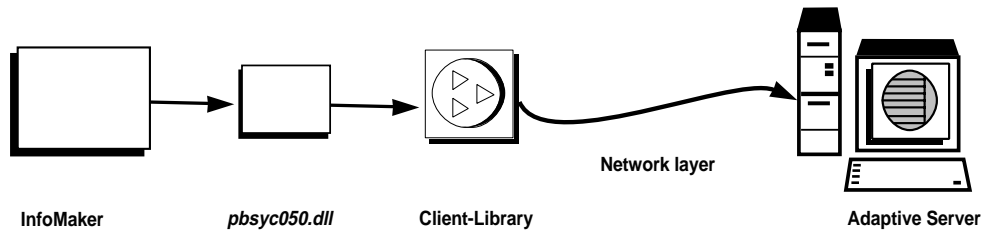


Figure 4-1: How InfoMaker connects to Adaptive Server

When Adaptive Server Enterprise for Windows NT is installed, *pbsync050.dll* is installed automatically, along with InfoMaker. However, Open Client is not installed automatically—it must be explicitly chosen.

Open Client should be installed on the same machine as InfoMaker. For more information about Open Client, see “About APIs” on page 1-9.

Task 1: Verify Installation of or Install the Powersoft Stored Procedures

InfoMaker uses special Powersoft stored procedures to create the Powersoft system tables, called the Powersoft Repository, the first time you connect to Adaptive Server (see “Task 3: Examine the Powersoft Repository Tables in Sybase Central” on page 4-5). These stored procedures may have already been installed on your

Adaptive Server, as detailed in *Installing Sybase Adaptive Server Enterprise on Windows NT*.

You can verify installation of the Powersoft stored procedures from the Sybase Central interface.

1. Start SQL Central, if it is not already running, and connect to your server.
2. Double-click the *sybssystemprocs* icon in the Databases folder.
sybssystemprocs is a system database that stores all the system procedures provided by Sybase. Powersoft stored procedures are also stored in this database.
3. Double-click the Stored Procedures folder to display the system procedures.
4. Use the scroll bar to scroll through the list, and look for procedures that begin with "sp_pb." These are the Powersoft stored procedures.
If the procedures have been installed, you should about twelve stored procedures with the "sp_pb" prefix.
5. If the procedures have not been installed, run the script *pbsync.sql*, located in the *\sybtools* directory.
For information on running scripts, see Appendix A, "Using isql to Run Scripts."
6. Leave Sybase Central up and running for the duration of the tutorial.

Task 2: Create a Database Profile

In order to connect to *eventsdb*, you need to create a database profile for it when you first start InfoMaker. In the final lesson of the tutorial, you will delete this profile.

1. Start InfoMaker.
If you are using InfoMaker for the first time, a login dialog box may appear. Click Cancel.
The Database Profile list appears.
If you are not using InfoMaker for the first time, click the Database Profile button, and the Database Profiles dialog box

appears. The Database Profiles button is in the PowerBar at the top of the screen:

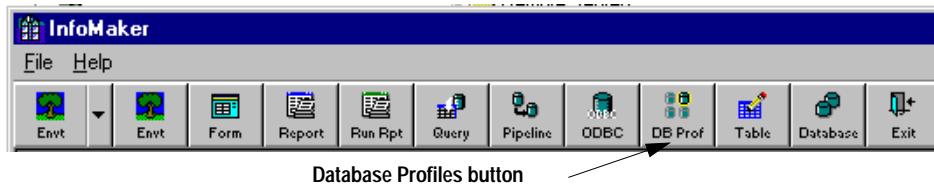


Figure 4-2: InfoMaker menu bar

The Database Profile list appears:

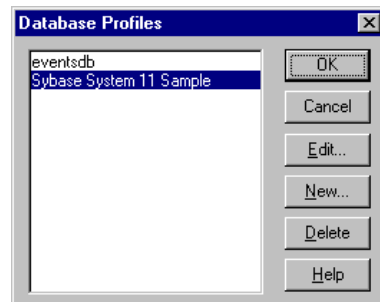


Figure 4-3: Database Profile list

2. Click New to create a new database profile.
3. Enter the following values:

Field Name	Value
Profile Name	<i>eventsdb</i>
DBMS	<i>syc</i>
User ID	Leave blank
Password	Leave blank
Database Name	<i>eventsdb</i>

InfoMaker gives you the option of using the ODBC interface instead of the native driver; ODBC appears as a pull-down option in the DBMS field. Use the native interface (*syc*) instead, as it requires less overhead.

4. Click More, and fill in the following values:

Field Name	Value
Server Name	Your server name
Login ID	student
Login password	The "student" password
DBPARM	Leave blank
Isolation Level	Leave blank

5. Click OK.
6. The new profile appears as selected in the Database Profiles dialog box.
7. Click OK.
8. Exit InfoMaker.

Task 3: Examine the Powersoft Repository Tables in Sybase Central

The first time you connect to Adaptive Server, InfoMaker automatically creates the five tables that constitute the Powersoft Repository. InfoMaker uses these tables to store extended attributes, InfoMaker-specific information about the tables and columns in the database. Some of this information is automatically generated by InfoMaker; some of this information is user-defined (see "Task 2: Define a Display Format" on page 4-13).

The five tables should now exist in the *eventsdb* database and are visible from SQL Central.

1. Start SQL Central, if it is not already running, and connect to your server.
2. Double-click the *eventsdb* icon in the Databases folder.
3. Open the User Tables folder.
4. Choose View→Refresh Folder.

In addition to the three tables that you created in the last lesson, *category*, *club*, and *event*, you should see five additional tables: *pbcatcol*, *pbcatdtd*, *pbcatfmt*, *pbcattbl*, and *pbcatvld*. These are the newly created Powersoft Repository tables that store the extended attributes for the tables in *eventsdb*.

Task 4: Run the Scripts to Populate the Powersoft Repository

Instead of taking the time now to define extended attributes for *eventsdb*, you can run a script to populate these tables with the appropriate values.

Run the script *repos.sql*, located in the *tutorial* directory.

For information on running scripts, see Appendix A, “Using isql to Run Scripts.”

Project Two: Entering and Importing Data into the *eventsdb* Database

You can enter data directly or import it. In this project, you will work with a tutorial library file, enter data directly, and import data from a file.

This project involves the following tasks:

- Task 1: Add Text to the PowerBar Display
- Task 2: Open the Tutorial Library
- Task 3: Open the Data Manipulation Painter for the Category Table
- Task 4: Add Rows to the Category Table
- Task 5: Import a File

Data Entry Overview

The most straightforward way to put data in a database is to enter the data directly into a grid. InfoMaker provides an easy-to-use interface that makes data entry easy.

In many real-world situations, some or all of the data already exists in or can be exported to files. For example, you may have some of your data in a spreadsheet or desktop database. You can export data from these programs into comma- or tab-delimited files. Or you can download some of your data from the World Wide Web into a tab-delimited file. You can use InfoMaker to import data from files into your Adaptive Server database.

In this project, you will use InfoMaker’s Data Manipulation painter to enter and import data into the *category* table. A **painter** is an InfoMaker tool for creating and working with objects of a particular type.

Task 1: Add Text to the PowerBar Display

This tutorial does not teach you the ins and outs of the user interface, which is very flexible (there are several different ways to accomplish most actions). For complete information about the InfoMaker interface, see the InfoMaker documentation. For simplicity, this tutorial provides only one way to accomplish each task.

However, take the time now to customize the toolbar, called the PowerBar, that appears at the top of the screen. You can customize the PowerBar to show text with each button so that it is easier to find the correct button.

1. Start InfoMaker.
2. Choose File→Toolbars.
3. Select PowerBar1.
4. Select the Show Text check box.
5. Click Close.

Task 2: Open the Tutorial Library

InfoMaker stores reports, forms, and other objects you create in an InfoMaker library file, which is a file with a *.pbl* extension.

We have created several objects for use in this tutorial. The objects are in the file *tutor.pbl*. InfoMaker's default library file, named *tutor_im.pbl*, is the one created for an InfoMaker-specific tutorial. In this task, you will close the default library and open the *tutor.pbl* file in the *tutorial* directory.

1. Click the Environment button in the PowerBar.

The current library displays. If your current library is the default library, *tutor_im.pbl*, it displays more than 40 InfoMaker objects—a partial list is shown in Figure 4-4.

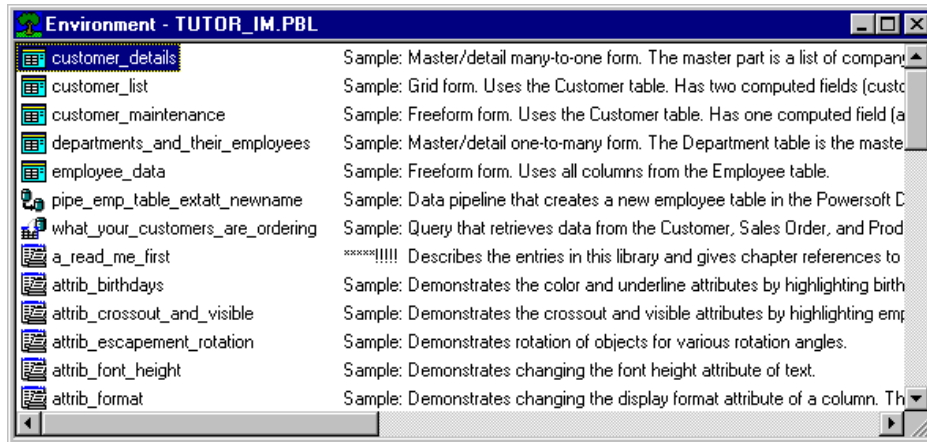


Figure 4-4: Partial list of objects in the default library

2. Select File→Open.
3. From the *tutorial* directory, select the file *tutor.pbl*.
4. Click Open.

The new library is displayed, as shown in Figure 4-5:

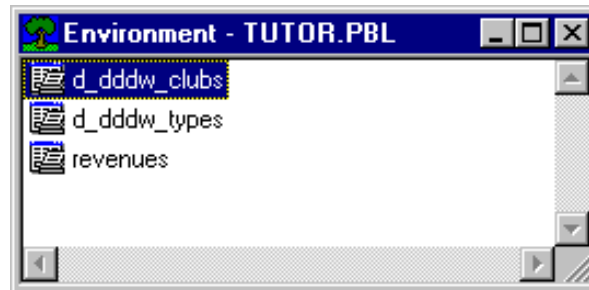


Figure 4-5: Tutor library

Leave the library open.

Task 3: Open the Data Manipulation Painter for the Category Table

In this task, you will open the *category* table in the Data Manipulation painter so that you can begin data entry.

1. Click the Database button in the PowerBar.
2. Select the *category* table, and click Open.
3. Choose Object→Edit Data→Grid to display the data in the *category* table in grid style in the Data Manipulation painter.

Since the table is empty, only the column heads appear.

You can add data to the table in this window.

Task 4: Add Rows to the Category Table

In this task you will add data to the *category* table grid.

1. Choose Rows→Insert.

The cursor appears in the first column of the first row.

2. Type “blu”, the code for the first category.
3. Press the Tab key.

The cursor is now in the second column of the first row.

4. Type “The Blues”, a full description of category of music.
5. Press the Tab key.

The cursor is now in the first column of a new row. Notice that you do not have to do anything special to insert a new row.

6. Enter the following rows:

c&w Country and Western

jzz Jazz

cjz Cajun/Zydeco

Instead of populating the entire table by typing the values, you can finish populating the table by importing data from a file.

Delete the rows you just added to the grid so that you can import all the rows from a file.

7. Choose Rows→Delete until the grid is empty.

Task 5: Import a File

1. Choose Rows→Import.
2. Navigate to the Text folder in the *tutorial* directory.
3. Select the file *category.txt*.

4. Click Open.

The grid is now populated with 18 rows.

5. Choose File→Save Changes to the Database to write the new data to the database.
6. Choose File→Close.

Project Three: Piping Data from Another Database

► **Note**

If Microsoft Access is installed on your computer, you can do this project. Microsoft Access is a common desktop database program that is part of the Microsoft Office suite. If you do not have Microsoft Access, import the file *club.txt* from the Text folder of the *tutorial* directory into the *club* table, following the directions in “Task 5: Import a File” on page 4-9. Choose File→Close. Then skip to “Project Four: Defining Extended Attributes” on page 4-13.

All or part of your data may already exist in a desktop database. You can use InfoMaker to import data from that database directly into your Adaptive Server database using a data pipeline.

Working with a data pipeline requires a configured ODBC data source and an InfoMaker database profile. This project involves the following tasks:

- Task 1: Configure an ODBC Data Source for *msclub*
- Task 2: Create a Database Profile for the MS Access Database
- Task 3: Create the Data Pipeline

Task 1: Configure an ODBC Data Source for *msclub*

You can define an ODBC data source from within InfoMaker.

1. Click the ODBC button in the PowerBar.
2. Select the Microsoft Access Driver.
3. Click Create.
4. Type “msclub” for the data source name.
5. Click Select.

6. Select the file *msclub.mdb* from the *tutorial* directory.
7. Close each window by clicking OK.
8. Click Close in the Configure ODBC window.

Task 2: Create a Database Profile for the MS Access Database

1. Click the Database Profiles button to call up the list of database profiles.
2. Click New.
3. Fill in the following values:

Field Name	Value
Profile name	msclub
DBMS	ODBC
User ID	Leave blank
Password	Leave blank
Database Name	Leave blank

4. Click OK.
The SQL Data Sources box appears.
5. Select *msclub*.
6. Click OK.
7. Close the Database Profiles box by clicking OK.
You should now be connected to the Microsoft Access database, as indicated in the status bar in the database workspace.

Task 3: Create the Data Pipeline

1. Click the Pipeline button in the topmost PowerBar.
2. Click New.
3. Select Quick Select as the Data Source.
In InfoMaker, the term **data source** refers to the method of specifying the data to retrieve.

4. Select *msclub* as the source connection and *eventsdb* as the destination connection, if they are not already selected.

5. Click OK.

The Quick Select window appears.

6. Select the table *club*.

7. Click Add All to select all the columns.

8. Click OK.

The Data Pipeline painter workspace appears.

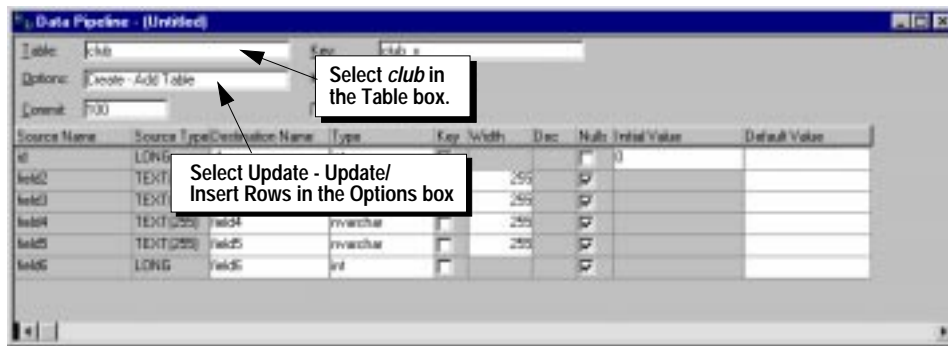


Figure 4-6: Data Pipeline painter workspace

9. Choose File→Save, and save the pipeline as *pipe_msclub_club*.

10. Select Update - Update/Insert Rows from the Options box, as shown in Figure 4-6. InfoMaker does not create a new table, but inserts the rows into a table that is already defined.

11. Select the destination table *club* from the Table box, as shown in Figure 4-6.

12. Choose Design→Execute.

If the pipeline was successful, the number of rows read and written appears at the bottom of the pipeline window. The correct number is 85.

13. Choose File→Close. Do not save your changes.

14. Click the Database Profiles button in the PowerBar.

15. Select *eventsdb*, and click OK to return to the *eventsdb* database.

16. Do not exit—leave InfoMaker up and running.

Project Four: Defining Extended Attributes

In InfoMaker, you can define what are called **extended attributes** for tables and columns in tables. Extended attributes let you store information about columns in the database for use in InfoMaker reports and forms.

In this project, you will work with a few different kinds of extended attributes. This project involves the following tasks:

- Task 1: Examine a Validation Rule
- Task 2: Define a Display Format

Task 1: Examine a Validation Rule

Extended attributes are stored in the database in the Powersoft Repository.

One example of an extended attribute is a validation rule. A validation rule has been defined for the *city* column of the *club* table, requiring that InfoMaker validate the data entry for that column. The city must be in the domain that you defined in the *eventsdb* business rules in a previous lesson (see “Business Rules” on page 3-5).

1. Click Database in the PowerBar.
2. Select the *club* table, and click Open.
3. Right-click in the *city* column, and choose Properties.
4. Select the Validation tab.

City_rule is highlighted, meaning that it is attached to the *city* column.

5. Click edit to display the rule.

The expression that defines the rule appears.

6. Click Cancel to close all open boxes.
7. Choose File→Close.

Task 2: Define a Display Format

Another kind of extended attribute is a display format. A display format defines how data is displayed in all forms, reports, and data grids that you create. For example, you can define how you want dates to be displayed. This information is stored in the *pbcattfmt* table.

In this task you will define a display format for the *price* column in *event* table.

1. Click Database in the PowerBar.
2. Select the *event* table, and click Open.
3. Right-click in the price column, and choose Properties.
4. Click the Display tab.
5. Select
`$#,##0.00;($#,##0.00)`
if it is not already selected.
6. Click Edit.
7. At the end of the string in the format box, type:
`;Free`
Adding a new field, which is delimited by the semicolon, defines how values that are equal to 0 will be displayed.
Values of 0 will be displayed as the word “Free.”
8. Test the display format. Type “0” in the Test Value box, and click Test.
“Free” should be displayed.
9. Click OK to close each window.
10. Choose File→Close.

About Edit Styles

Another powerful extended attribute is edit styles, which are stored in *pbcatd.t*. Edit styles are similar to display formats, but they also make data entry easier as well as define how data is displayed. If a column has both an edit style and a display format, the edit style has precedence.

A prominent edit style is the DropDownDataWindow edit style. This type of edit style is useful for columns that must meet a referential integrity constraint. This style allows the user to choose from a list of legal values for data entry. This list of legal values is dynamic—it comes from the referenced table itself and is updated at run time.

In addition, you can display a value that is more meaningful to the user than the actual data being stored. Figure 4-7 shows how

defining a DropDownDataWindow edit style for the *eventid* column in the *club* table makes data entry easier.

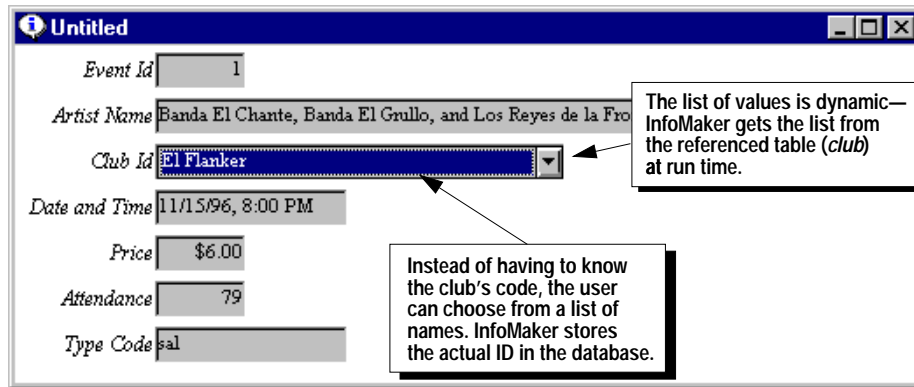


Figure 4-7: DropDownDataWindow edit style

InfoMaker supports a similar edit style called the DropDownListBox edit style. The difference is the list of values is a static code table and is not derived from the database itself.

Project Five: Creating Forms

Forms are the easiest and most sophisticated way to enter data in a database using InfoMaker.

This project involves the following tasks:

- Task 1: Create the Event Data Entry Form
- Task 2: Add an Action Button
- Task 3: Run the Form

In this project, you will create a form for entering data into the *event* table. Before you begin this project, the *club* and *category* tables should be populated. If you skipped “Task 5: Import a File” on page 4-9, run the script `club_cat.sql` now. For information on running scripts, see Appendix A, “Using isql to Run Scripts.”

Task 1: Create the Event Data Entry Form

1. Click Form in the PowerBar.
The Select Form window appears.
2. Click New.
The New Form dialog box appears.
3. Select Quick Select as the type of data source.
4. Select free-form as the presentation style.
In InfoMaker, the term data source refers to the method of specifying the data to be retrieved. The presentation style you choose determines the layout of a form or report.
5. Click OK.
6. Select the table *event*.
7. Click Add All to add all the columns in the table.
8. Click OK.
The Form painter workspace appears.
9. Choose File→Save, and save the form as *event_data_entry*.

Task 2: Add an Action Button

The Form painter workspace is where you design and enhance the form. You can move objects around and add static text or buttons. In this task you will add buttons.

Adding buttons to a form makes it easy for the user to perform common tasks such as adding a row.

1. Choose Controls→CommandButton.
2. Click anywhere in the design space.
3. Type "Insert Row." The text appears on the face of the button.
4. Right-click the button, and choose Action to associate the button with a predefined action.
5. Select Insert_Row from the list, and click OK.

The completed form will look something like the one shown in Figure 4-8.

Figure 4-8: The data entry form

Task 3: Run the Form

1. Choose Design→Run.
2. Click OK.
3. Type “1” in the Event Id box.
4. Press the Tab key.

You can move between entry boxes using the Tab key.

Two boxes have DropDownDataWindows, *Club Id* and *Type Code*. To enter data in those boxes, tab to the box to select it.

Then, choose the value from the DropDownDataWindow, and press the Tab key.

5. Enter the values shown in the table below:

Column Name	Value
<i>Artist Name</i>	Banda El Chante
<i>Club ID</i>	El Flanker
<i>Date and Time</i>	12/27/96 8 PM
<i>Price</i>	6
<i>Attendance</i>	
<i>Type Code</i>	Salsa

6. Choose Rows→Update to save the changes to the database.
7. Choose File→Close.
8. Exit InfoMaker.

What's Next

In the next lesson, “Creating Reports and Applications with InfoMaker,” you will create reports in InfoMaker and build an InfoMaker application.

5

Creating Reports and Applications with InfoMaker

► **Note**

This lesson takes about 25 minutes.

If you skipped the lesson “Designing the Events Database with SQL Modeler,” run the script `design.sql`. If you skipped the lesson “Entering and Importing Data with InfoMaker,” complete the first project in that chapter, “Project One: Connecting InfoMaker to Adaptive Server” on page 4-1. All users must run the script `data.sql` to populate the database with data for the reports before moving on to this lesson. For information on running scripts, see Appendix A, “Using isql to Run Scripts.”

Where You Are

In the previous lessons, you accomplished some system administration tasks, designed and built the events database, *eventsdb*, and entered and imported data. You also learned about the Powersoft Repository and extended attributes.

In this lesson, you will develop three reports and then build an InfoMaker application. This lesson includes the following concepts and projects:

- About the Reports 5-1
- Project One: Developing the Event Listing Report 5-2
- Project Two: Developing the City-Type Report 5-6
- Project Three: Enhancing the Revenues Report 5-9
- Project Four: Creating an InfoMaker Application 5-11
- About Deploying Applications 5-12

About the Reports

In this lesson you will develop three reports:

- Event Listing Report

This report uses the grid presentation style. It lists events that you will submit to the local newspaper, detailing the date, time,

artist, and club information about the event. You will create a client-side filter for this report.

- City-Type Report

This report uses the crosstab presentation style. You will define the cross-tabulation, which displays the relationship between cities and styles of music. A **crosstab** displays data in a spreadsheet-like grid. Instead of the value of specific rows, it displays summary data—sum, average, count, and the like.

- Revenues Report

This report uses the freeform presentation style. It includes price and attendance data and computed fields and groups data by club name. You will create a server-side filter.

Project One: Developing the Event Listing Report

The first report, called *event_report*, uses the *artist_club_date* view. As you may recall, you created this view in Chapter 3, “Designing the Events Database with SQL Modeler,” to make compiling calendar information easy.

Remember that you need to run the script `data.sql` before doing this lesson.

This project involves the following tasks:

- Task 1: Create the New Report
- Task 2: Preview the Report (Retrieve Rows)
- Task 3: Add a Filter

Task 1: Create the New Report

1. Start InfoMaker.
2. Click the Report button in the PowerBar to open the Report painter.
3. Click New.
4. Select Quick Select for the data source and Grid for the Presentation Style.

In InfoMaker, the term data source refers to the method of specifying the data to be retrieved. The presentation style you choose determines the layout of the report.

5. Clear the Preview When Built check box, if it is selected.
6. Click OK.
7. Select the *artist_club_date* view.
8. Select the following columns in the order shown:
 - *the_date*
 - *artist*
 - *name*
 - *type*

The columns appear at the bottom of the window in the order in which you selected them.

9. Click OK.
10. Choose Edit→Properties.
11. Select the Print Specifications tab, and select Landscape as the paper orientation.
12. Click OK.

Task 2: Preview the Report (Retrieve Rows)

1. Choose Design→Preview.

You can scroll through the report by clicking the arrows in the PainterBar.

Scroll through the pages in a report with the arrows in the PainterBar.

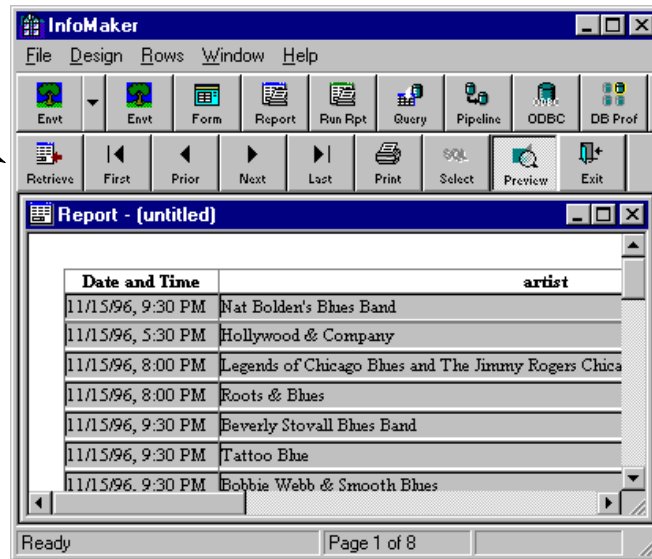


Figure 5-1: Scrolling through a report

2. Choose Design→Preview to go back to design mode.

Task 3: Add a Filter

You can filter out rows that do not meet a certain condition, either on the server side or on the client side. InfoMaker makes it easy to create a client-side filter. Adaptive Server returns **all** the rows; InfoMaker displays only the desired rows. For information on creating a server-side filter, see “Task 2: Add a Retrieval Argument” on page 5-10. It is easier to create a client-side filter, but server-side filters cause less network traffic and are more efficient.

In this task, you will define a filter that displays only the rows for which the date stored in *the_date* column is equal to a specified date.

1. Choose Rows→Filter.

A Specify Filter dialog box appears.

InfoMaker provides a graphical interface for defining the expression.

2. Select date(s) from the list of functions.

`date()` is an InfoMaker function that returns the date of the parameter. “s” is a placeholder for the parameter. The “s” should be highlighted, ready for you to type or select a parameter from the list of columns.

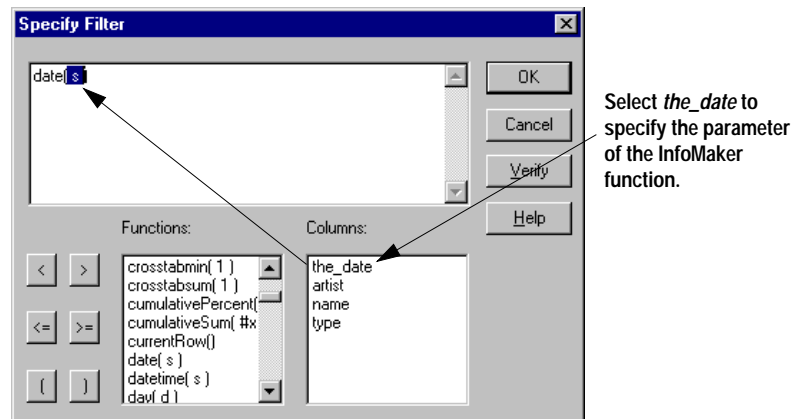


Figure 5-2: Specifying a filter

3. Select *the_date* from the list of columns as the parameter, as shown in Figure 5-2.
4. Insert the cursor to the right of the current expression, and type “=”.
5. Select date(s) again from the list of functions, and type the following parameter:
`'11/7/96'`
 This parameter is one of the dates in the *event* table. Be sure to include the single quotation marks surrounding the date.
6. Click Verify to check the validity of the expression. The message “Filter is OK” appears.
7. Click OK.
8. Click OK to return to the Report painter design space.
9. Choose Design→Preview.
 Only the rows for 11/7/96 should be visible.
10. Choose Design→Preview to go back to design mode.

11. Choose File→Save and save the report as *event_report*.
12. Choose File→Close.

Project Two: Developing the City-Type Report

The second report, called *city_type*, is a crosstab that uses all three tables in the data source. You will use the SQL Select data source type and will build a query using a tool, which is called the Select painter.

This project involves the following tasks:

- Task 1: Start the New Report
- Task 2: Define the Data Source

A **crosstab** displays data in a spreadsheet-like grid. Instead of the value of specific rows, it displays summary data—sum, average, count, and the like. Figure 5-3 shows a subset of the crosstab you will make in this project.

Count Of Descrip	City			
Descrip	Albany	Berkeley	Oakland	Grand Total
Folk Music			21	21
Jazz		25	36	63
The Blues		6	28	34
Grand Total	2	52	64	118

Summary data goes in the interior cells.

Jazz events happened 25 times in Berkeley.

Figure 5-3: A crosstab

Task 1: Start the New Report

1. Click Report in the PowerBar.
2. Click New.
3. Select SQL Select for the data source and Crosstab for the presentation style.
4. Clear the Preview When Built check box, if it is selected.
5. Click OK.

Task 2: Define the Data Source

1. In the Select Tables dialog box, select the following tables:

- *category*
- *event*
- *club*

2. Click Open.

The three tables appear in the Select painter workspace, with a red line showing probable joins.

3. Select *city* from the *club* table and *descrip* from the *category* table.

The columns you have selected appear at the top of the workspace in the Selection List, as shown in Figure 5-4.

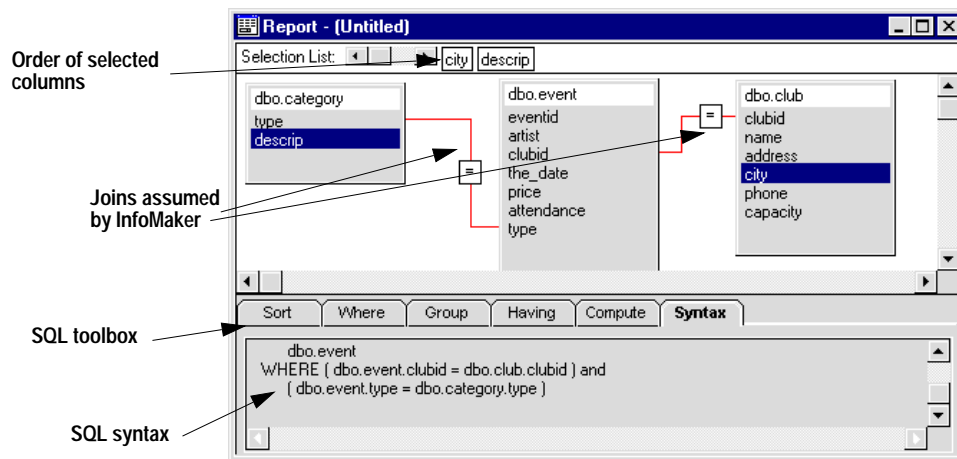


Figure 5-4: Select painter workspace

4. Click the Syntax tab in the SQL toolbox to examine the syntax of the query built by InfoMaker. InfoMaker determines the joins, based on key definitions. InfoMaker displays a Transact-SQL statement similar to the following:

```

SELECT club.city,
category.descrip
FROM category,
club,
event
WHERE (event.clubid = club.clubid) AND
(event.type = category.type)

```

5. Choose Design→Data Source.

The Crosstab Definition dialog box appears.

6. Drag and drop *descrip* from the Source Data box to the Rows box, as shown in Figure 5-5. When the crosstab is built, descriptions of the types of music will appear down the left side of the crosstab.

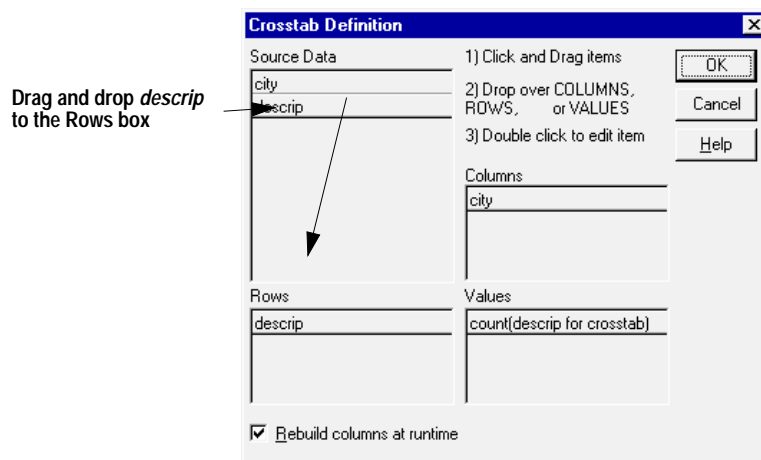


Figure 5-5: Defining a crosstab

7. Drag and drop *city* from the Source Data box to the Columns box. When the crosstab is built, names of cities will appear across the top of the report.

8. Drag and drop *descrip* from the Source Data box again—this time to the Values box.

InfoMaker guesses the type of aggregate column that will appear in the interior cells of the crosstab and displays `count(descrip for crosstab)` in the Values box. This is the correct aggregate value for the crosstab.

9. Click OK.

10. Choose Design→Preview to preview the report.

The report shows the number of events of each type of music that have appeared in each city.

11. Choose Design→Preview to go back to design mode.
12. Choose File→ Save. Save the file as *city_type*.
13. Choose File→Close.

You can add highlighting to a report such as this to make it easier to analyze it at a glance. For example, you could add color to all cells where the value is greater than 20. For information on highlighting data, see the *InfoMaker User's Guide*.

Project Three: Enhancing the Revenues Report

The third report, *revenues*, includes data about the price and attendance of the event and the capacity of the club in which it was held. It also includes two client-side computed fields, *revenue* (price * attendance) and percent *filled* (attendance /capacity). It groups data by type of music and summarizes the data by group.

In “Project One: Developing the Event Listing Report” on page 5-2, you created a client-side filter for data. In this project, you will define a retrieval argument for the report to create a server-side filter.

This project involves the following tasks:

- Task 1: Open the Predefined Report
- Task 2: Add a Retrieval Argument

Task 1: Open the Predefined Report

The basic structure and layout of the *revenues* report has been defined already.

1. Click the Report button in the PowerBar to open the Report painter.
2. Select Revenues from the list of Reports, and click Preview.
3. Choose Design→Preview to go back to design mode.

Task 2: Add a Retrieval Argument

You can define a report to prompt you to specify information about which rows to retrieve when you preview the report. InfoMaker uses the values you supply when requesting the rows from Adaptive Server.

You define a retrieval argument when defining the data source. After you define the retrieval argument, you reference the retrieval argument in the Where or Having tab in the SQL painter.

In this task, you will define a week (numbered 1–52) as a retrieval argument. When you preview or run the report, InfoMaker will prompt for a value.

1. Choose Design → Edit Data Source.
2. Choose Design → Retrieval Arguments.
3. Type “week” in the Name box, and click OK.
4. Select the Where tab in the SQL toolbox.

5. In the Column box, type:

```
datepart(wk, the_date)
```

datepart(wk) is an Adaptive Server function that returns the week number of the specified date.

6. Press the Tab key.
7. In the Operator column, type “=”.
8. Press the Tab key.
9. In the Value column, type:

```
:week
```

You reference retrieval arguments with their name preceded by a colon.

10. Choose Design → Data Source.
11. Choose Design → Preview.

InfoMaker prompts you for a value for the retrieval argument.

12. Type “45” in the value box, and click OK.

Only rows for the 45th week of the year should be visible.

13. Choose Design → Preview to go back to design mode.
14. Choose File → Save.
15. Choose File → Close.

Project Four: Creating an InfoMaker Application

You can bundle completed reports and forms into an InfoMaker application, an executable file that you can distribute to other users, even if they do not have InfoMaker installed.

This project involves the following tasks:

- Task 1: Create an Application
- Task 2: Use the Application

You create applications in the Environment painter.

Task 1: Create an Application

1. Open the Environment painter by clicking the Environment button in the PowerBar.
2. Select all the reports and forms that appear in the tutorial library, as shown in Figure 5-6.

Control-click to select multiple items.

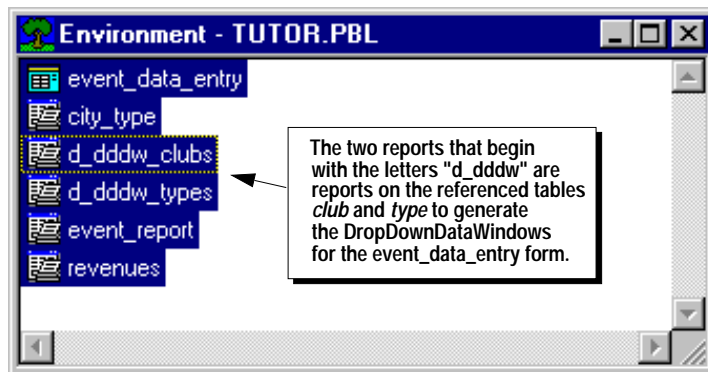


Figure 5-6: Select all reports and forms for the application.

3. Choose Entry→Create Executable.
The Create Executable dialog box appears.
4. Type "dbArts" as the Executable Title.
5. Type "dbarts.exe" as the Executable File name.
6. Click the Browse button next to the Executable File Name box, and navigate to the *tutorial* directory.

7. Click Save.
8. Click the Browse button next to the Executable Icon box.
9. Navigate to the *tutorial* directory, and select *dbarts.ico*.
10. Click Open.
11. Click Next.
12. Click Finish in the Executable Items window. You do not need to do anything with this window except click Finish.

InfoMaker creates two files:

- *dbarts.exe*
- *dbarts.ini*

Together, these two files make up the application.

Task 2: Use the Application

1. Quit InfoMaker.
2. Start the application from the command line or from the File Manager or from Windows Explorer.
A Connect dialog box appears.
3. Type the “student” user ID and password.
4. Click OK.
The application starts.
5. To run the form, click the Form button. To run a report, click Report.
6. Exit the application.

About Deploying Applications

You can distribute the dbArts application to other users, who do not need InfoMaker to run the application. You can use the Install Builder program, which automates the deployment process that creates an installation program. Install Builder is included with Adaptive Server Enterprise for Windows NT; it is located in `\pws\InBld32`. For more information, see the *Install Builder User's Guide*.

You can also distribute an application with the Powersoft Deployment Kit.

In order for users to be able to use the dbArts application, they need:

- An application directory
- Open Client installed and network information configured on their machines (see *Installing Sybase Adaptive Server Enterprise on Windows NT* for information)
- The Powersoft database driver, *pbsyc050.dll*, installed in the application directory
- The executable and initialization files—in this example, *dbarts.exe* and *dbarts.ini*, respectively—installed in the application directory
- InfoMaker DLLs installed in the application directory

The Powersoft Deployment Kit automatically installs the Powersoft database driver and the InfoMaker DLLs. You would have to explicitly give users the other items in the list. For more information about deploying applications, see the *InfoMaker User's Guide*.

What's Next

In the next lesson, “Building a Dynamic Web Site with NetImpact Dynamo,” you will work with NetImpact Dynamo to create a database-enabled Web application.

6

Building a Dynamic Web Site with NetImpact Dynamo

► **Note**

This lesson takes about 40 minutes.

If you skipped the lesson “Designing the Events Database with SQL Modeler,” run the script `design.sql`. Before you begin this chapter, run the script `update.sql`. For information on running scripts, see Appendix A, “Using isql to Run Scripts.”

Where You Are

In previous lessons, you accomplished some system administration tasks, designed and generated the *eventsdb* database, entered and imported data, developed reports, and then built and ran an InfoMaker application.

In this lesson, you will build templates, the foundation of a NetImpact Dynamo Web application, and learn about DynaScript, a scripting language that allows you to add application logic to a template. The lesson will build up to working with a Web application that allows the general public to view a calendar of events for a specific week and type of music and to enter an upcoming event into the database.

As indicated in the “Note” above, you need to run the script `update.sql` before doing this lesson.

This lesson includes the following concepts and projects:

- NetImpact Dynamo As an Application Server 6-2
- Project One: Getting Started with NetImpact Dynamo 6-4
- Project Two: Creating a Template 6-7
- Project Three: Improving the Template 6-16
- Project Four: Working with DynaScript 6-19
- Project Five: Working with Forms 6-26
- Project Six: Importing the Web Site 6-29
- How the Application Works 6-32

NetImpact Dynamo As an Application Server

This section discusses some concepts that are useful for understanding how NetImpact Dynamo works as an application server in an Adaptive Server environment.

Beyond Client/Server: The Internet

As discussed in Chapter 5, “Creating Reports and Applications with InfoMaker,” to run an InfoMaker application, you need the InfoMaker executable, the Powersoft DLL, and the Sybase libraries to connect to Adaptive Server.

In this chapter, you will get a brief introduction to making a **thin-client** World Wide Web application. A thin-client application handles only presentation, and leaves logic and connectivity to an application server. For a NetImpact Dynamo Web application, a Web browser (and access to the Internet or your intranet) is all that users need.

NetImpact Dynamo is a client application that acts as a middle-tier **application server**, handling all connections to Adaptive Server through the ODBC interface and providing the application logic. Web clients do not need a local executable or local API, as illustrated in Figure 6-1. (For more information about APIs and the ODBC interface, see “About APIs” on page 1-9.)

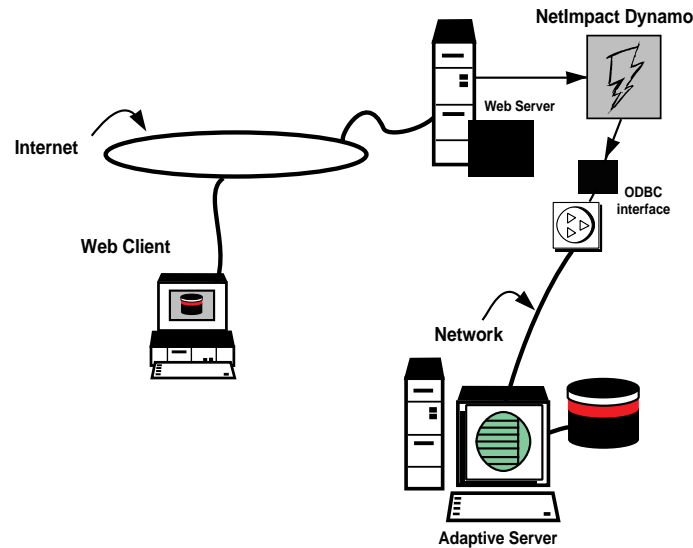


Figure 6-1: A NetImpact Dynamo Web site configuration

Beyond HTML: Dynamic Content

Ordinary Web sites consist of static HTML files. **HTML**, or HyperText Markup Language, is the language with which World Wide Web text documents are formatted. It defines the basic details of a document such as paragraph tags, graphics, and hypertext links. HTML files typically end with *.html* or *.htm*. File-based Web sites require constant updating.

The foundation of NetImpact Dynamo Web site is the **template**. A template, which has an *.stm* extension, is an object that includes the dynamic content embedded in static HTML. This dynamic content consists of:

- Results of queries sent to Adaptive Server wrapped in HTML
- DynaScript, an interpreted scripting language you can use to add application logic

You will build and learn more about templates in “Project Two: Creating a Template” on page 6-7.

Project One: Getting Started with NetImpact Dynamo

Before you start working with NetImpact Dynamo, you need to prepare an ODBC data source and a connection profile, which you will do in this project. Then you will connect to the database and automatically generate a Web site.

This project involves the following tasks:

- Task 1: Configure an ODBC Data Source for the Web Site
- Task 2: Create a Connection Profile
- Task 3: Connect to eventsdb
- Task 4: Examine the Web Site Tables

Task 1: Configure an ODBC Data Source for the Web Site

1. Start the ODBC Administrator.
2. Click Add.
3. Select the Sybase System 11 ODBC Driver.
4. Click OK.
5. Type “web_events” as the Data Source name.
6. Type the name of your Adaptive Server.
7. Type “eventsdb” as the Database Name.
8. Click Connection.
9. Type “student” as the Default Logon ID.
10. Click Performance.
11. Check for the following two settings:
 - Prepare Method should be 2-Full.
 - Select Method should be 0-Cursor.

An incorrect setting for either method can prevent you from being able to build a Web site.

12. Click OK.
13. Click Close.

Task 2: Create a Connection Profile

To create a Web site, you need to define a NetImpact Dynamo connection profile for the specific database you want to connect to. A connection profile is associated with an ODBC data source. You create connection profiles and do most Web site generation and administration in Sybase Central.

1. Start Sybase Central.
2. Select Tools→Connection Profiles.
The Connection Profiles dialog box is displayed.
3. Click New.
The Create New Profile window appears.
4. Type “website_events” as the profile name, and select NetImpactDynamo from the list of available types. Click OK.
The connection window appears.
5. Select the *web_events* ODBC data source from the list of available sources.
6. Type the “student” user ID and password. Click OK.

Task 3: Connect to *eventsdb*

1. Select the website_events profile in the Connection Profiles box.
2. Click Connect.
Sybase Central asks if you want to create a Web site on this connection.
3. Click Yes.
The first time NetImpact Dynamo connects to a database, it generates a site, creates a set of tables to hold the Web objects, and displays the objects in the Sybase Central interface.
After you click Yes, NetImpact Dynamo asks you for the name of the main folder for the Web site.
4. Type “MusicSite” as the Web site name.
NetImpact Dynamo asks if you want to install the NetImpact Dynamo documentation and sample application.

Do not select either of these options. Both are designed for the *psdemodb* sample database. For more information, see *Installing Sybase Adaptive Server Enterprise on Windows NT*.

5. Click OK.

After site generation, two folders appear in the work space, Connections and MusicSite. You may need to scroll down in the Sybase Central window and double-click NetImpact Dynamo to display the folders.

Task 4: Examine the Web Site Tables

There are now new site-related tables in the *eventsdb* database. You can examine these tables from the Sybase Central interface. To examine these tables, you need to make an Adaptive Server connection to your server (as opposed to a NetImpact Dynamo connection), if not already connected. You can have both connections active simultaneously, as shown in Figure 6-2. To connect to Adaptive Server, see Chapter 2, “Sybase Central and Adaptive Server System Administration Basics.”

► **Note**

Do not alter these site-related tables directly.

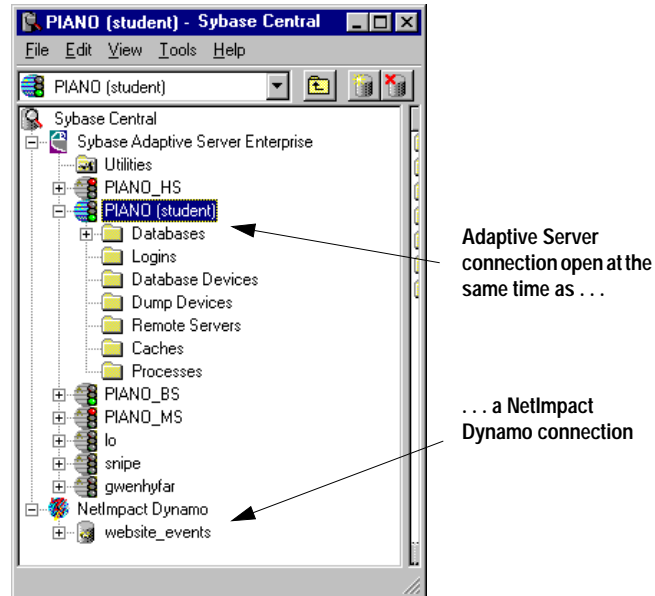


Figure 6-2: Working with Adaptive Server and NetImpact Dynamo connections

1. Double-click the server icon in Sybase Central.
2. Open the Databases folder.
3. Double-click *eventsdb*.
4. Double-click the User Tables folder.
5. Choose View→Refresh Folder.

The following new tables appear in the list:

- *WebConnection*
- *WebData*
- *WebDocumentType*
- *WebSynchronize*
- *WebTemplate*

Project Two: Creating a Template

The basic Dynamo template includes queries as well as formatting and static HTML. NetImpact Dynamo sends embedded queries to

Adaptive Server, formats the results as defined by the formatting, and sends the formatted results along with any static HTML and text to the Web server, which sends it to the Web client that requested it.

In this project you will build a template that includes a query and HTML formatting. It involves the following tasks:

- Task 1: Create the Calendar Template with an Embedded Query
- Task 2: Configure the NetImpact Dynamo Personal Web Server
- Task 3: Start the Personal Web Server and a Web Browser
- Task 4: Map MusicSite to a Port Number (Optional)
- Task 5: View the Output

About Embedded Queries and Formatting

Embedded queries are stored in the template and are delimited by special HTML comment tags, as shown in Figure 6-3. You do not have to type either the Transact-SQL code or the HTML comment tags—you can use a tool (or begin with the tool and refine the default code).

It is a good idea to become familiar with the different parts of a template.

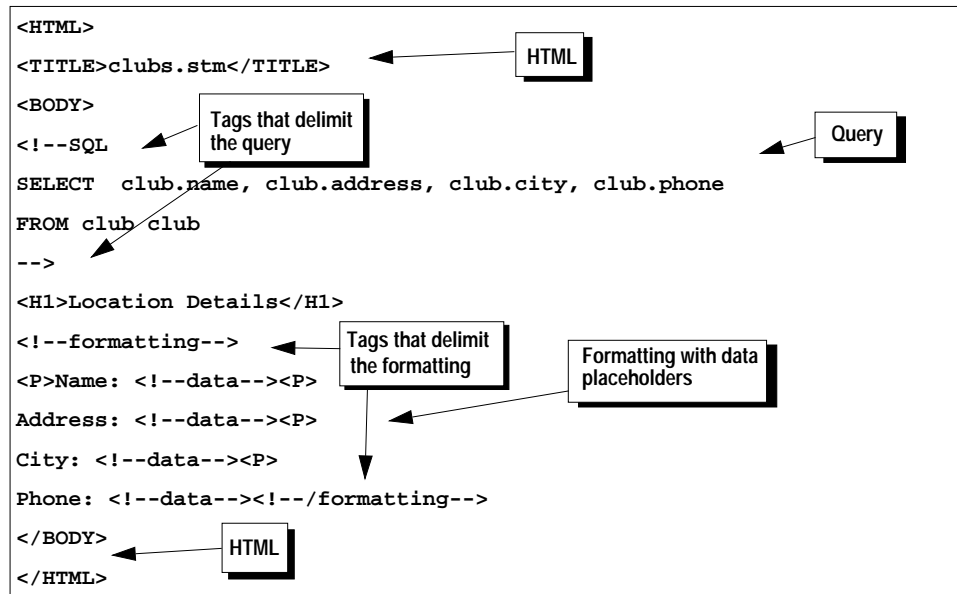


Figure 6-3: A basic Dynamo template

The HTML formatting for data output sits next to data placeholders that are delimited by a formatting comment tag. Only one row of placeholders is included in the template. When it processes the template, NetImpact Dynamo duplicates the formatting for all rows in the result set before sending the results back to the user.

Task 1: Create the Calendar Template with an Embedded Query

NetImpact Dynamo provides an easy way to build a query and to define formatting for data output.

In this task, you will create a template called *Calendar* that displays basic event data.

1. Double-click NetImpact Dynamo in the Sybase Central window on the left.
2. Double-click `website_events`.
3. Double-click the MusicSite folder.
4. Double-click Add Template.
5. Type "Calendar" as the name of the Template, and click Next.

6. Enter a description of the template, such as “A calendar of musical events,” and click Next.
Descriptions are displayed in the Sybase Central interface and as a level-one head in the page.
You can either type a SQL statement or use a tool to build one. In this task, you will use the tool, which is called the SQL Query Editor.
7. Click Select.
If a logon box appears, click OK.
The Edit Query dialog box appears.
8. Select the Tables tab if it is not already displayed.
9. Select View in the Table Type box.
10. Select the *artist_club_date* view.
11. Click Add.
12. Select the Compute tab to add a computed field.
13. In the New Computed Column field, type:

```
datename(dw, the_date)
```


datename is an Adaptive Server function that returns the day of the week of the date, passed as a parameter.
14. Click Add.
15. Select the Columns tab.
16. Click the plus sign next to the table name to display the available columns.
17. Select the *name* column, and click Add to add it to the query.
Repeat for the *artist* and *descrip* columns.
The columns appear in the Selected Columns list.
18. Select the Test tab.
19. Clear the “Quote Table and Column Names” check box. Quoting the table and column names is not correct syntax for Adaptive Server.
20. Clear the “Qualify Table Names with Owner” check box.
This is an optional setting.
21. Click Test to see if the query returns data.

Data appears in the data window, verifying the validity of the query.

22. Click OK.
23. Click Next.
24. Choose Table with Headings as the default HTML format for the data.
25. Click Finish.

A new template, *Calendar.stm*, appears in the Sybase Central workspace.

You will view the resulting HTML document from a Web browser in a later task.

Task 2: Configure the NetImpact Dynamo Personal Web Server

To connect to the Web site from a local Web browser, you have to configure NetImpact Dynamo's Personal Web Server. To use the Personal Web Server, you must have TCP/IP installed.

1. Start the program Dynamo Configuration.

A message appears warning you that changes will not take effect until NetImpact Dynamo applications are restarted. Click OK.
2. Click Add.
3. Type “/MusicSite” as the URL prefix that Web browsers will reference.
4. Select DB as the access method.
5. Click Browse to enter the local path to the resource on your machine.
6. Choose the website_events profile.
7. Click OK.

The new URL Prefix appears in the Configurator Window, as shown in Figure 6-4.

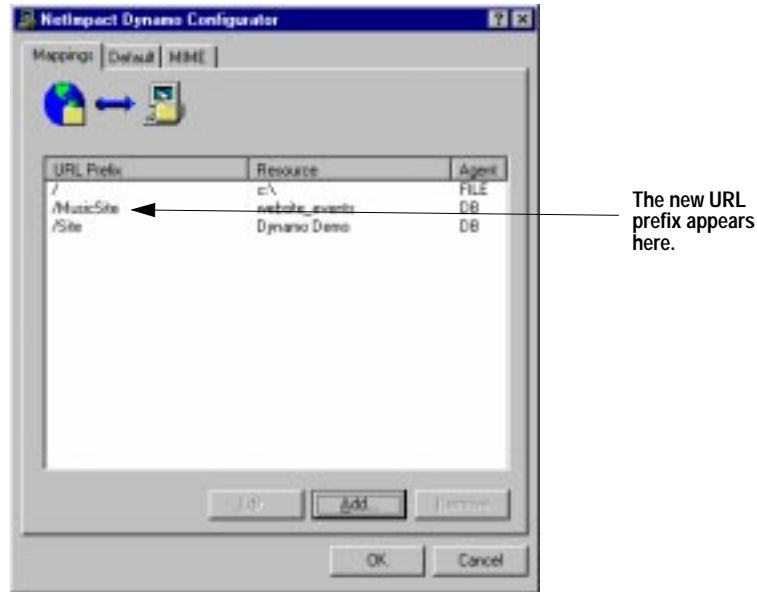


Figure 6-4: Configuring the Personal Web Server

8. Click OK again.

Task 3: Start the Personal Web Server and a Web Browser

1. Start the program Personal Web Server.

If you are running Windows NT 4.0 or Windows 95, an icon appears in the Start bar. If you are running Windows NT 3.51, the Configurator window displays. Minimize this window.

2. Start a Web browser.
3. Open the following URL:

`http://localhost/MusicSit/Calendar.stm`

- In Netscape Navigator, you open a URL by choosing File→Open Location, typing the URL, and clicking Open.
- In Microsoft Internet Explorer, you open a URL by choosing File→Open, typing the URL, and clicking OK.

If you get an error message in the browser window, double-check the Dynamo configuration. Be sure the MusicSite URL

Prefix appears in the Configurator Window, as shown in Figure 6-4.

Another reason for an error may be that you have another Web server loaded on your computer that uses the default port number, 80. You may have to map the MusicSite to a port number other than the default. You will do that in the next task. Many users have Web servers on their computers without knowing it.

Task 4: Map MusicSite to a Port Number (Optional)

1. Start the program Dynamo Configuration.
2. Select the Default tab.
3. Type a new Port number such as 90.
4. Click OK.

Changes do not take effect until you restart Personal Web Server.

5. Exit Personal Web Server.
 - In Windows NT 4.0 and Windows 95, you can exit the Personal Web Server by right-clicking on the Personal Web Server icon in the Start panel and choosing Exit.
 - In Windows NT 3.51, you can exit the Personal Web Server by restoring the icon that you minimized and then exiting.
6. Start the Personal Web Server again.
7. Request the URL again. When requesting the URL from a Web browser, insert a colon and the new port number after the word "localhost," like this:

```
http://localhost:90/MusicSite/Calendar.stm
```

► **Note**

If you mapped MusicSite to a new port number, add the colon and the new port number after the word "localhost" throughout this lesson when requesting a URL. Another option is to map the other Web server to an alternate port number and to use the default port number, 80, for the Personal Web Server.

Task 5: View the Output

The document appears in the Web browser window. It displays data in a table format, since you chose Table with Headings as the HTML display format in “Task 1: Create the Calendar Template with an Embedded Query” on page 6-9. Figure 6-5 and Figure 6-6 illustrate how NetImpact Dynamo and a Web browser turn the Dynamo code into a Web document.

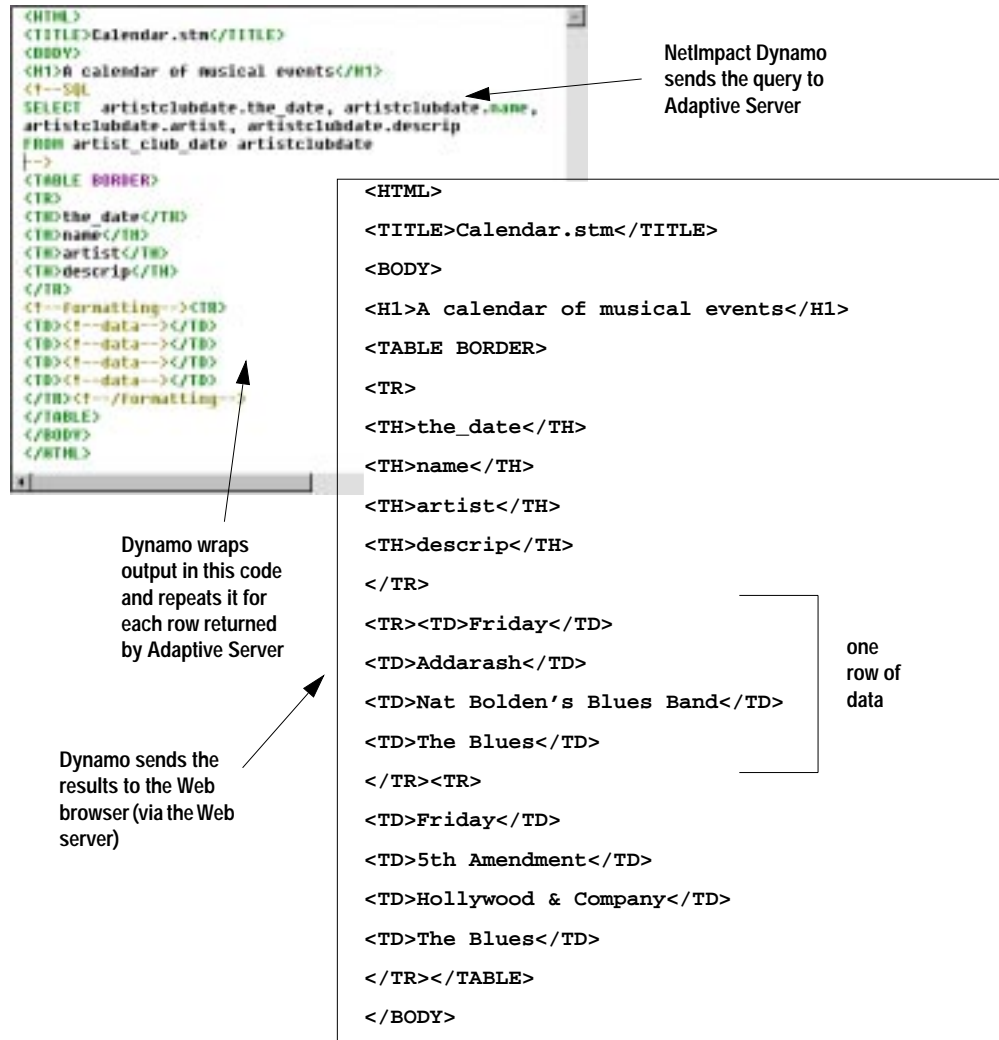
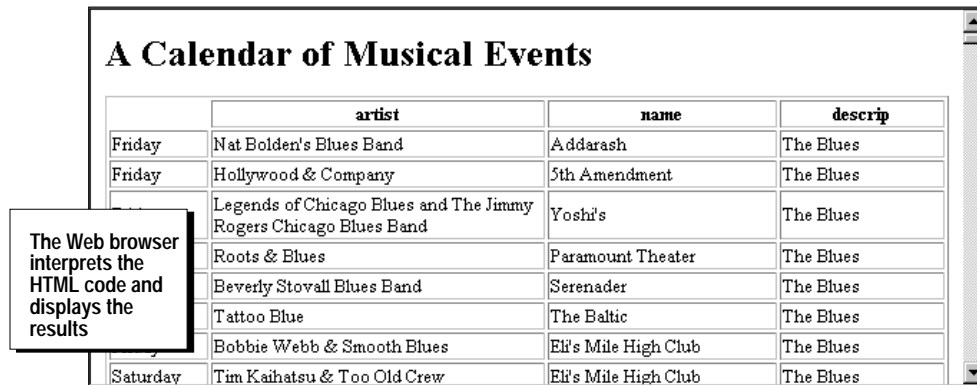


Figure 6-5: Template source code and the resulting HTML code



The Web browser interprets the HTML code and displays the results

	artist	name	descrip
Friday	Nat Bolden's Blues Band	Addarash	The Blues
Friday	Hollywood & Company	5th Amendment	The Blues
	Legends of Chicago Blues and The Jimmy Rogers Chicago Blues Band	Yoshi's	The Blues
	Roots & Blues	Paramount Theater	The Blues
	Beverly Stovall Blues Band	Serenader	The Blues
	Tattoo Blue	The Baltic	The Blues
	Bobbie Webb & Smooth Blues	El's Mile High Club	The Blues
Saturday	Tim Kaihatsu & Too Old Crew	El's Mile High Club	The Blues

Figure 6-6: Final output in a browser

Project Three: Improving the Template

You can make changes in the template from within Sybase Central and view the results from the open Web browser. When you change the template, you will save the changes to the database. Then, you will reload or refresh the document in the Web browser to view the changed document.

This project involves the following tasks:

- Task 1: Refine the Formatting and the Query
- Task 2: Save Changes to the Database and Reload the URL

Task 1: Refine the Formatting and the Query

You may not be happy with the default HTML formatting inserted by NetImpact Dynamo for the table. To change formatting, you can refine the code as desired. In this task, you will delete the border from the table and change the heading. In addition, you will alter the query so that the data appears in chronological order.

NetImpact Dynamo's script editor displays coding of different types in specific colors. You can use this feature to avoid typographical errors that cause errors in script interpretation. Refer to the illustrations throughout the rest of this chapter for correct color coding.

1. Double-click *Calendar.stm* in the Sybase Central window.
The text of the template appears.
Figure 6-7 illustrates the changes you will make in steps 2 and 3.
2. The text delimited by the HTML tags <H1> and </H1> is a top-level, or level-1, head. Change this text to “East Bay Musical Events.”
3. Delete the word “BORDER” and the space preceding it from the <TABLE BORDER> tag so that it reads “<TABLE>”.

```
<HTML>
<TITLE>Calendar.stm</TITLE>
<BODY>
<H1>A calendar of musical events</H1>
<!--SQL
SELECT datename(dw, the_date), artistclubdate.name,
artistclubdate.artist, artistclubdate.descrip
FROM artist_club_date artistclubdate
-->
<TABLE BORDER>
```

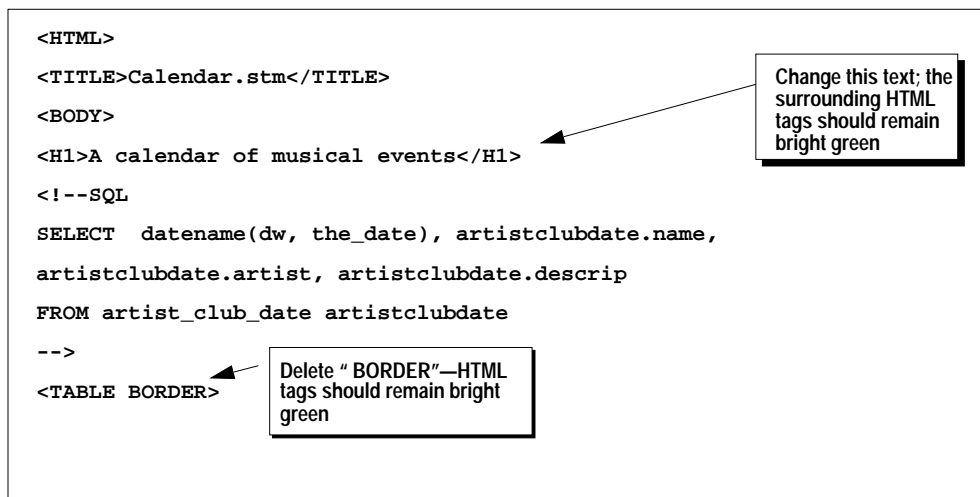


Figure 6-7: Altering the template

4. Place the cursor below the final line of the embedded SQL statement.
5. Type the following, ending with a carriage return:
`order by the_date`

Figure 6-8 illustrates steps 4 and 5.

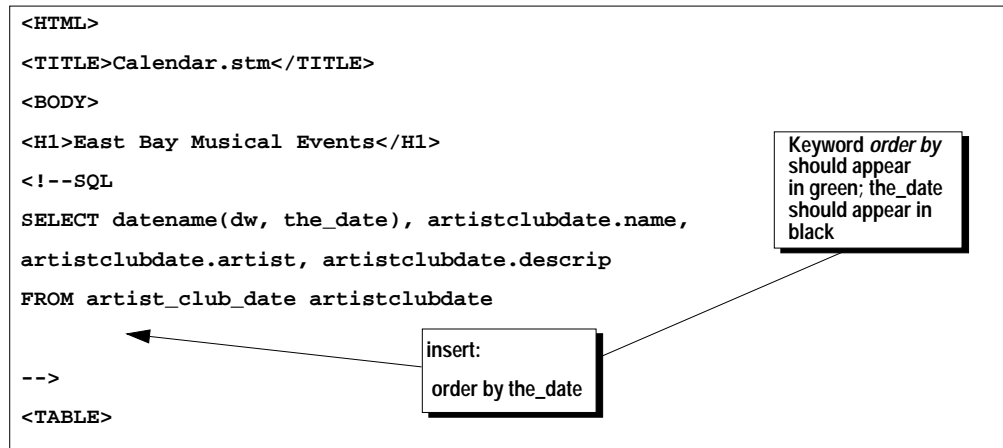


Figure 6-8: Refining the query

Task 2: Save Changes to the Database and Reload the URL

1. Choose File→Save to Database.
2. Close *Calendar.stm*.
3. Reload or refresh the document in the Web browser.
 - In Netscape Navigator, you **reload** a document by clicking the Reload button or choosing View→Reload.
 - In Internet Explorer, you **refresh** a document by clicking the Refresh button.

The document appears with the changes you just made and with the data in chronological order.

Some typographical errors in the SQL statement are undetected by the script editor. For example, it is possible to mistype “the_date” as “the_dates.” In this situation, you will get the following error message:

```

An error has occurred expanding the template. Line
7: Column 1 SQL State s00022 [INTERSOLV] [ODBC SQL
Server driver] [SQL Server] Invalid column name
'the_dates'.

```

If you get a script or SQL error message, fix the error in the template, save the changes to the database, and reload the URL in the Web browser.

Project Four: Working with DynaScript

In addition to embedded queries, formatting, and static HTML, you can add application logic to templates by including NetImpact DynaScript. You can also equip templates to process parameters passed from a Web browser appended to the URL or submitted via an HTML form.

This project involves the following tasks:

- Task 1: Add DynaScript to the Template
- Task 2: Pass an Argument to a Template
- Task 3: Open a Template That References a Query with DynaScript

About DynaScript

Figure 6-9 shows a complex NetImpact Dynamo template that includes DynaScript as well as JavaScript. (For more information about using JavaScript in the NetImpact Dynamo templates, see “Client-Side Data Checking with JavaScript” on page 6-33.)

As with queries, NetImpact Dynamo interprets DynaScript before returning results to the user.

DynaScript is a JavaScript-compatible scripting language; both are object-based scripting languages. As in object-oriented programming, an **object** in these scripting languages is a unique instance of a data structure that hold pieces of information and functions that operate on this information.

DynaScript provides many capabilities that help you work with the Web site and includes some built-in objects that will be useful. For example, it includes a predefined query object and a document object, among others. You reference the results of embedded SQL queries as query objects.

For more information on the DynaScript and DynaScript objects, see the NetImpact Dynamo documentation. For more information about JavaScript and JavaScript’s support for objects, see one of the many commercial books available, or see the Netscape online tutorial and guide on the World Wide Web.

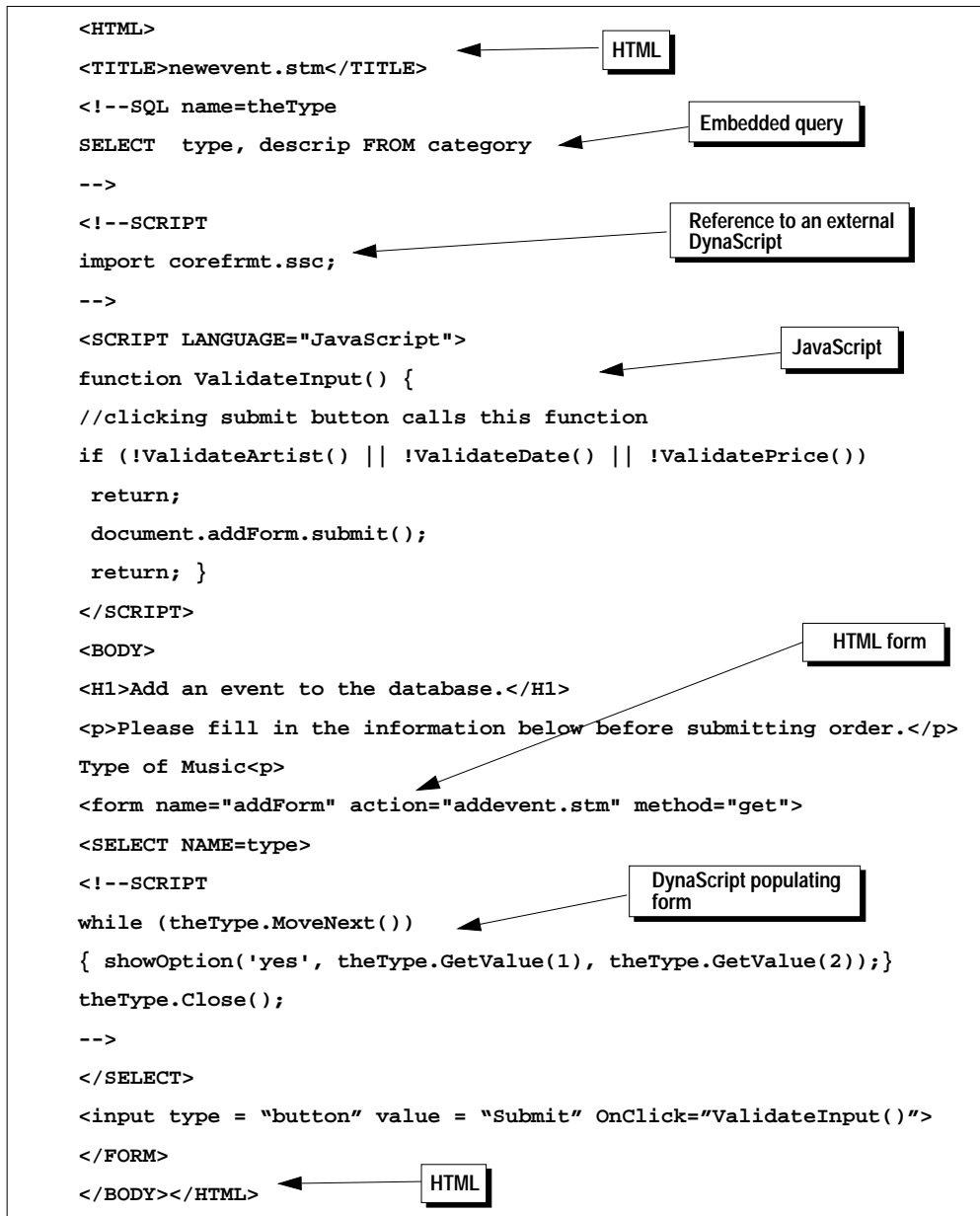


Figure 6-9: A complex DynaScript template

Task 1: Add DynaScript to the Template

1. Open the *Calendar.stm* template in Sybase Central.
2. Place the cursor between the HTML close comment tag that delimits the SQL query and the <TABLE> start table command.
3. Insert a carriage return, and type these two lines of code, each followed by a carriage return:

```
<!--SCRIPT
-->
```

These tags delimit DynaScript.

When you insert code between these two tags, insert a carriage return after the code so that the final tag appears on a line by itself. The carriage return after the final tag is for readability only.

Figure 6-10 illustrates steps 2 and 3.

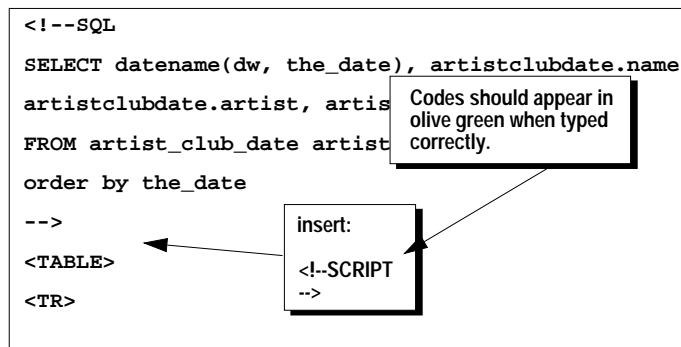


Figure 6-10: Adding DynaScript

4. Type the following lines of code within the script delimiters, each followed by a carriage return:

```
mynum = 2 + 2;
document.writeln("Hello, World! <P>");
document.writeln("2 + 2 " + mynum + "<P>");
```

<P> is an HTML tag that inserts a new line.

You can include HTML formatting tags alongside the text as arguments to the DynaScript `document.writeln()` function if you treat them as a string.

Figure 6-11 illustrates step 4.

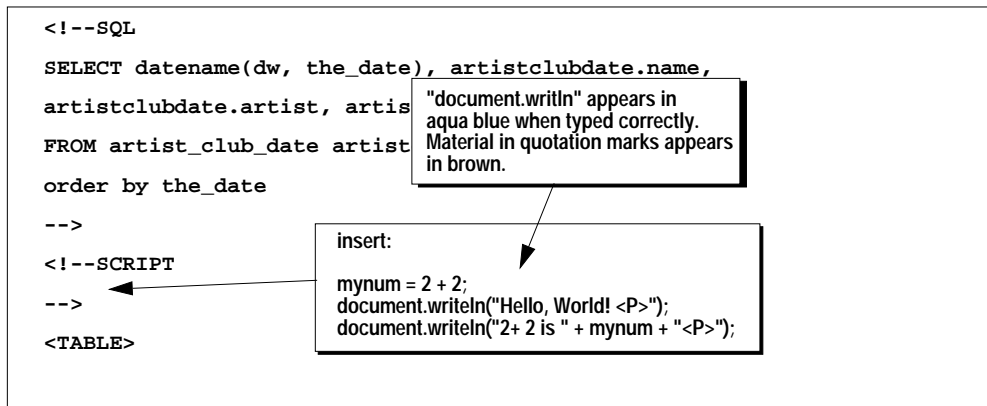


Figure 6-11: Adding DynaScript

5. Choose File→Save to Database, and reload or refresh the document in the Web browser.

About Script Error Messages

If you receive an error warning, return to the open template and fix the error. Refer to Figure 6-11 if you need to. Save changes to the database, as in step 5, above, and reload or refresh the document in the Web browser.

Task 2: Pass an Argument to a Template

Users can supply an argument when they request a document, which will be customized for the value they supply. The usual way to supply an argument is to fill in a form. (You will work with forms in the next task, “Task 1: Work with a Form Page” on page 6-26.)

The simplest way to supply an argument is to add it at the end of the URL, as in the following string:

```
?argumentname=value
```

In this task, you will create an argument called “name,” reference it from within HTML and DynaScript, and supply the value appended to the URL that you request.

1. Return to the open *Calendar.stm* template.
2. Place the cursor below the script element that you added in “Project Four: Working with DynaScript” on page 6-19. The cursor should be above the <TABLE> tag.
3. Add a carriage return, and type the following:

```
Hello, $name!<P>
```

You can refer to an argument directly in the HTML and query portions of a template, where its name is preceded with the “\$” (dollar sign). This is called a **text replacement macro**. NetImpact Dynamo substitutes the text replacement macro with the value passed in as an argument before sending the query.

Figure 6-12 illustrates steps 2 and 3.

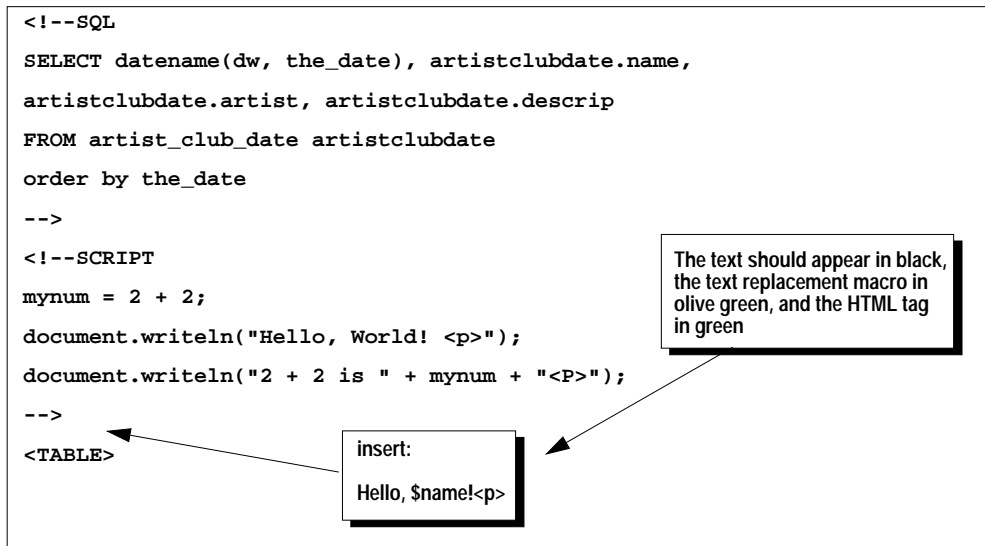


Figure 6-12: Using a text replacement macro

4. Request the following URL in the Web browser, substituting your name for *yourname*:

```
http://localhost/MusicSit/Calendar.stm?name=yourname
```

The document should display your name.

Task 3: Open a Template That References a Query with DynaScript

Instead of using the formatting repeated for each row of output, which is delimited by the <!--Formatting--> tags, you can access and program the results of queries from within DynaScript. To do so, you use the query object, for which NetImpact Dynamo provides predefined methods. Along with objects, object-oriented programming includes properties and methods. **Methods** are functions that specify an action for an object.

The query object methods move you through the rows of the result set. You work with the results of a query using the following query object methods:

- **MoveNext()**
Changes the current row to the next row in the result set.
- **GetValue()**
Returns the value of the specified column of the current row of the result set.
- **Close()**
Closes the active query. If you do not close an active query, the results of the query remain in *tempdb*.

In this task you will open a template that accesses the results of queries using DynaScript's query object methods and programs the output with a user-defined function.

1. Close the template *Calendar.stm*.
2. Delete the template *Calendar.stm* from the Sybase Central workspace. To delete a file, select it, and press the Delete key.
3. Choose File→Import.
4. Click File Browse.
5. From the *tutorial* directory select both *Calendar.stm* and *Form.stm*. Control-click to select multiple files.
6. Click Import.
7. Click OK.
8. Open the newly imported *Calendar.stm* template.
9. Examine the template. Figure 6-13 identifies new elements and concepts.

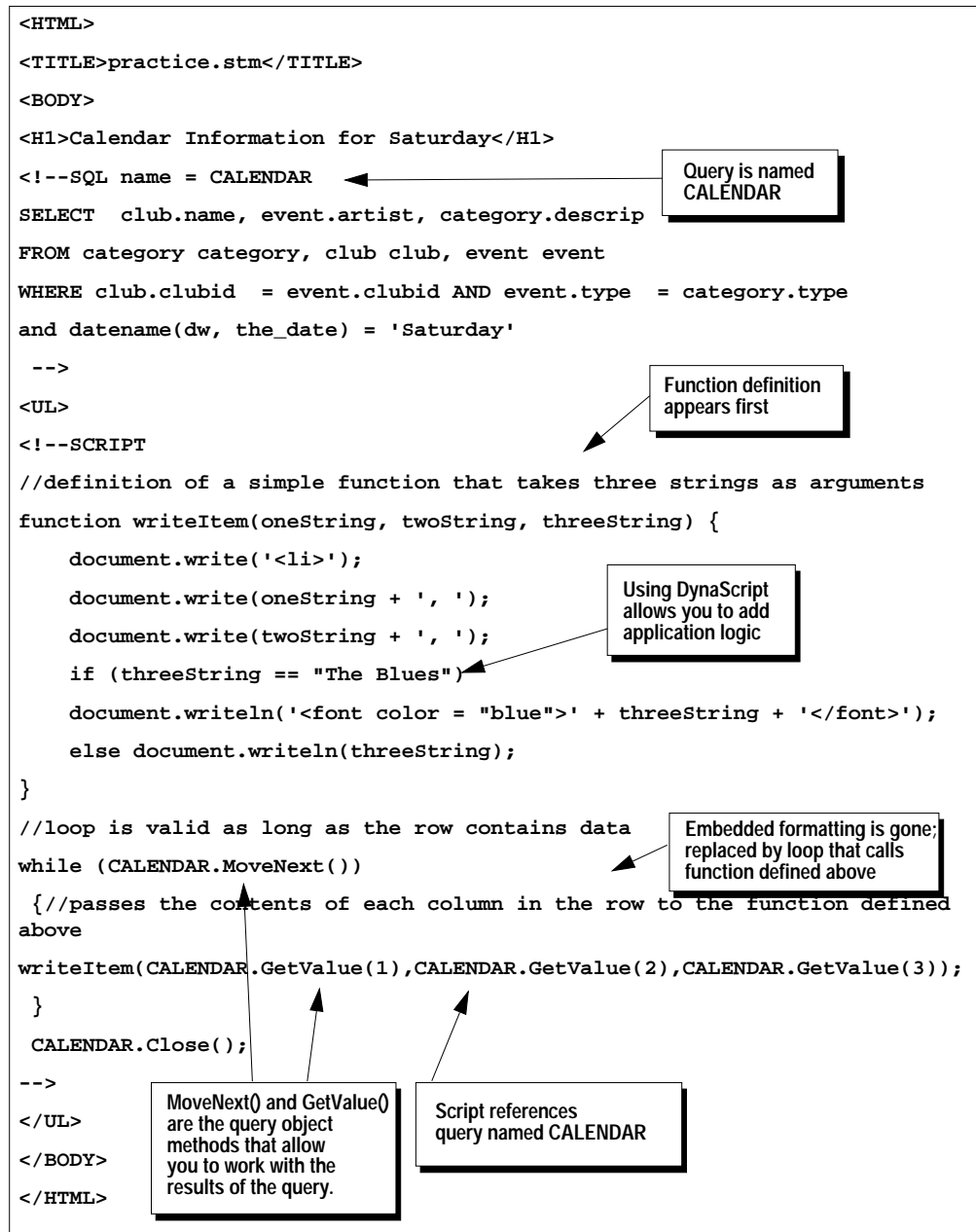


Figure 6-13: Advanced elements of the calendar template

10. Request the following URL from the Web browser:

```
http://localhost/MusicSit/Calendar.stm
```

11. Close the open *Calendar.stm* template in Sybase Central.

Project Five: Working with Forms

In “Task 2: Pass an Argument to a Template” on page 6-22, you passed an argument to a template by appending it to the URL. The usual way of passing an argument to a template is to enter a value in a form. In this project, you will work with a template that includes a form and pass an argument to the *Calendar.stm* template via the form page.

This project involves the following tasks:

- Task 1: Work with a Form Page
- Task 2: Edit the Calendar Template to Receive the Argument
- Task 3: Pass the Argument to the Template via the Form

Task 1: Work with a Form Page

1. Open *Form.stm* in Sybase Central.
2. Request the following URL from the Web browser:

```
http://localhost/MusicSit/Form.stm
```

The template includes an HTML form with a field where the user can specify the day for which he wants to view events. This field is an HTML <SELECT> element that shows a list of options—in this case, days of the week.

As shown in Figure 6-14, the name of the field is *datename*, which also becomes the name of the argument that will be passed by the form.


```

<HTML>
<TITLE>Form.stm</TITLE>
<BODY>
<H1>View events for a specific day.</H1>
Choose a Day. <p>
<FORM ACTION = "Calendar.stm" method="get">
<SELECT NAME=datename>
<OPTION>"Sunday"</OPTION>
<OPTION>"Monday"</OPTION>
<OPTION>"Tuesday"</OPTION>
<OPTION>"Wednesday"</OPTION>
<OPTION>"Thursday"</OPTION>
<OPTION>"Friday"</OPTION>
<OPTION>"Saturday"</OPTION>
</SELECT><P>
<INPUT TYPE="submit" value = "Proceed to Calendar"><P></FORM>
</BODY>

```

The form's "action" is another template—clicking the Submit button calls up the specified template and passes the argument, which is the value of the <SELECT> element field.

The field name is *datename*.

The HTML <SELECT> element generates a dropdown list box populated with the options specified here.

Drop-down list box

In this example, the value of *datename* is "Sunday".

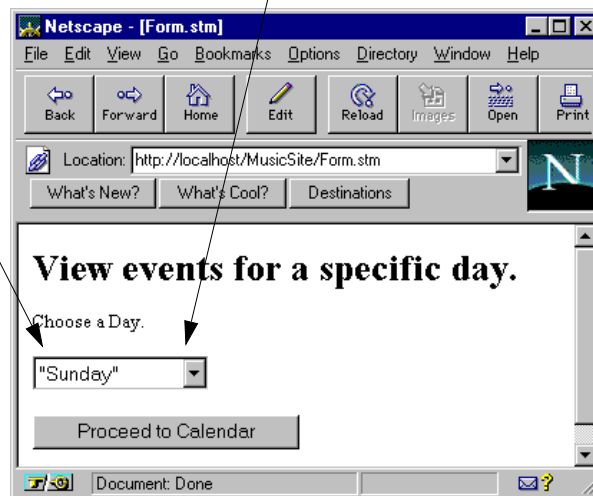


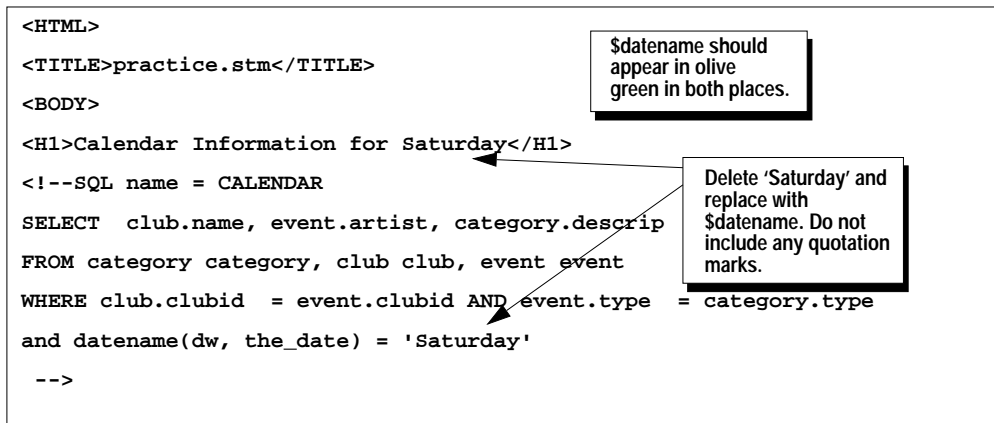
Figure 6-14: An HTML form

Task 2: Edit the Calendar Template to Receive the Argument

Calendar.stm originally showed events for Saturday. You can alter the existing *Calendar.stm* template to receive the user input and to show a list of events for the day the user specifies in the form.

1. Open the template *Calendar.stm* in Sybase Central.
You are going to replace the word “Saturday” with the text replacement macro `$datetime`.
2. Delete the word “Saturday” from the `<H1>` title, and replace it with `$datetime`.
3. Delete ‘Saturday’ from the SQL query named “CALENDAR” and replace it with `$datetime`. Do not include any quotation marks.

Figure 6-15 illustrates steps 2 and 3.



```

<HTML>
<TITLE>practice.stm</TITLE>
<BODY>
<H1>Calendar Information for Saturday</H1>
<!--SQL name = CALENDAR
SELECT club.name, event.artist, category.descrip
FROM category category, club club, event event
WHERE club.clubid = event.clubid AND event.type = category.type
and datetime(dw, the_date) = 'Saturday'
-->

```

Figure 6-15: Referencing an argument from a form

4. Choose File→Save To Database.

Task 3: Pass the Argument to the Template via the Form

1. Request the following URL in the browser:
`http://localhost/MusicSit/Form.stm`
2. Select a day in the form field.
3. Click Submit.

The Web browser displays the events for the day you specified. The Sybase Press book *Sybase SQL Server on the World Wide Web* (1996, International Thompson Computer Press) provides a good discussion of forms and how data is passed over the Internet.

Project Six: Importing the Web Site

In this project, you will import the dbArts Web application and examine its templates and scripts.

This project involves the following tasks:

- Task 1: Import the Predefined Site
- Task 2: Try Out the Application from a Web Browser

About the Application

The dbArts Web application is simple. The top tier of the application is an entry page. From this entry page, the user can view a calendar or submit an event.

The second tier of the application requests information from the user. To view a calendar, the user must provide the date and type of music. To submit an event, the user must provide the date, artist name, club name, type of music, and the cost of the event.

The third—and bottom—tier of the application processes the information provided by the user to produce the desired results—the calendar of the user's choice or the submission (and confirmation) of the event. If the user has chosen a calendar, he or she can also branch to specific information about a club where an event will be held.

Figure 6-16 illustrates the application.

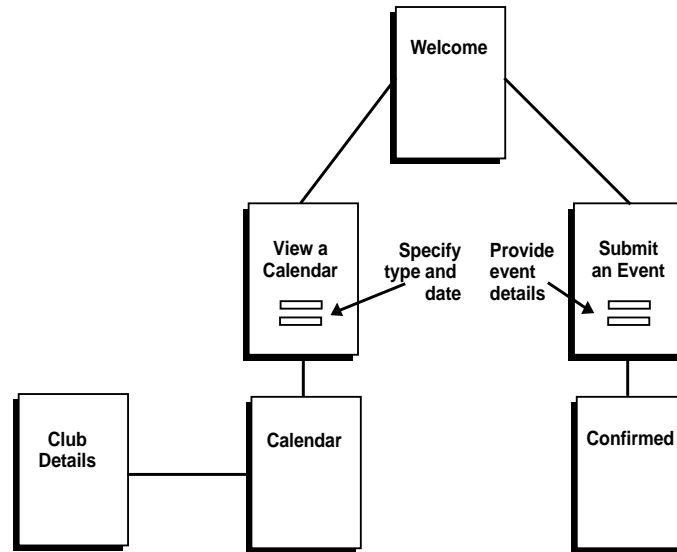


Figure 6-16: The dbArts Web application

The application queries the database at its second and third tiers. At the second tier, the application requests information about valid user choices for some of the form entries; at the third tier, the application requests data from or submits information to the database, based on the user's input.

Figure 6-17 illustrates how the application interacts with the database.

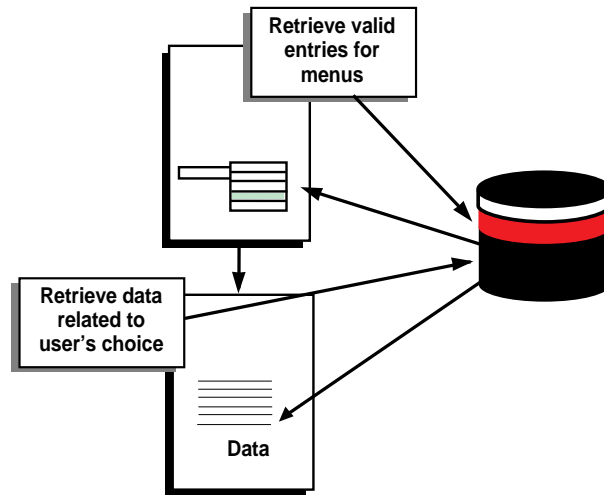


Figure 6-17: How the dbArts Web application interacts with the database

Task 1: Import the Predefined Site

1. Delete *Calendar.stm* and *Form.stm* from the Sybase Central interface.
2. Choose File→Import.
3. Click Folder Browse.
4. Navigate to the *tutorial* folder, and open the MusicSite folder.
5. Select the Images folder, and click OK.
6. Click File Browse.
7. Navigate to the *tutorial* folder, and open the MusicSite folder.
8. Select all files in the folder. Control-click to select multiple files (12 files should be selected).
9. Click Import.
10. Click OK.

Task 2: Try Out the Application from a Web Browser

Since the application includes JavaScript, you need a JavaScript-enabled browser to use the application. JavaScript is an optional

element that you can include in the NetImpact Dynamo templates. It is interpreted by the Web browser, not by NetImpact Dynamo. For more information on JavaScript, see “Client-Side Data Checking with JavaScript” on page 6-33.

Connect to the new site from a Web browser, and try out the application.

1. Start the Personal Web Server, if it is not already running.

2. Request the following URL from the Web browser:

```
http://localhost/MusicSit/welcome.stm
```

3. Experiment with the application. Try the following:

- View a calendar for a week and type of your choice.
- View information about a specific club listed in the calendar.
- Submit a new event.
- Try submitting an event without completing all the fields on the form.

How the Application Works

Look at the site from the Sybase Central interface. This section is a brief glimpse at how some parts of the application work. You may want to open the files in Sybase Central for an insider’s view of how the application was created. For complete instructions on working with NetImpact Dynamo, see the *NetImpact Dynamo User’s Guide*.

About the Script Files

The first thing you might notice when viewing the Web site in Sybase Central is that there are five objects with the .ssc extension. These are script files, which are similar to templates, but include only DynaScript, not queries or static HTML. This application uses the script files exclusively for defining objects, methods, and functions.

The template files reference the script files so that they can call the functions and methods. To reference a script file, you **import** it near the top of the template. The script files are organized by topic. Examine the files if you want some simple but concrete examples. The next two sections discuss features defined in two of the script files.

Navigation Bar Generation in *navbar.ssc*

Each page includes a navigation bar with buttons that link to the main pages—the home page (*Welcome.stm*), the submit page (*newevent.stm*), and the calendar page (*startcal.stm*)—plus a button that goes back to the previous page.

The objects, methods, and functions for creating the navigation bar are defined in the script file *navbar.ssc*. The templates call the function `navBar()`, which dynamically creates the navigation bar based a value passed to it by the calling page. The function omits a button for a template if that template is the one calling the function—for example, the function omits the Home button on the home page.

The individual pieces of information that make up an object are called **properties**. Location is a property of the DynaScript's predefined document object that returns the path and name of the current document. So instead of having to hard code the path and name of the current document into each template, the templates pass the value of `document.location`. For example, on the home page, `document.location` returns the value *Welcome.stm*, so this is the value of the parameter passed to `navBar()`.

HTML Formatting in *corefmt.ssc*

The script file *corefmt.ssc* includes formats for generating HTML. Instead of typing and retyping HTML code for hypertext links and the like, you can create formats that automate the process and provide consistency in your code. For example, `makeLink()`, one of the functions in this script file, writes the HTML codes for a hyperlink based on the value of the path and text string passed to it. You can look in the script file *corefmt.ssc* in Sybase Central for further details.

Client-Side Data Checking with JavaScript

You can off-load some of the programming functionality to the client by embedding JavaScript into the template, as shown in Figure 6-9 on page 6-20. The *newevent.stm* template off-loads some of the data-checking to the client, using JavaScript functions. NetImpact Dynamo sends the JavaScript to the browser, untouched.

JavaScript-enabled Web browsers interpret the JavaScript functions. (Current releases of both Netscape Navigator and Internet Explorer support JavaScript.).

What's Next

In the next lesson, “When You Have Completed the Tutorial,” you will perform a few tasks to undo the work you did in the first six lessons.

7

When You Have Completed the Tutorial

Where You Are

Now that you have finished with the tutorial, you need to undo the work you have done, for the sake of Adaptive Server security and to leave a “clean slate” for the next student.

Project: Undoing the Tutorial Activities

You can do most of the cleanup by running a script name `undo.sql`. There are also some additional steps involved. Follow the steps below:

1. Start Sybase Central, if it is not already running.
2. Exit your Web browser and the Personal Web Server, if they are running.
3. Disconnect the `website_events` connection, if it is still connected:
 - Choose Tools→Disconnect.
 - Select `website_events`.
 - Click Disconnect.
4. Choose Tools→Connection Profiles.
5. Select `website_events`.
6. Click Remove.
7. Click OK when asked to confirm.
8. Click Close to exit Connection Profiles.
9. Run the script `undo.sql` using the “sa” login and password or the login and password given to you by the System Administrator gave you (see “About the “sa” Login” on page 2-5).

For complete information on running scripts, see Appendix A, “Using isql to Run Scripts.”

10. If your system is a pre-production environment, shut down and then restart Adaptive Server to free up the `tutorial_dev` device. You can stop and start a local server in the Sybase Central interface by right-clicking the server icon and selecting Stop or Start. For more instructions on starting or stopping an Adaptive Server, see *Configuring Adaptive Server Enterprise for Windows NT*.

If your system is a production environment, tell the System Administrator you have dropped *tutorial_dev*.

11. Delete the file *tutor.dat* from `\sybase\dat`.
You will not be able to delete these files until you restart Adaptive Server, as instructed in step 10.
12. Start the ODBC Administrator, select *create_events*, and click Delete. Click Yes when asked to confirm.
13. Select *web_events* and click Delete. Click Yes when asked to confirm.
14. Select *msclub*, if it appears, and click Delete. Click Yes when asked to confirm.
15. Exit the ODBC Administrator.
16. Start InfoMaker.
17. Click the Database Profiles button.
18. Select *eventsdb* and click Delete.
19. Select *msclub*, if it appears, and Click Delete.
20. Click Cancel, and exit InfoMaker.

Thank you for taking this hands-on tour of Adaptive Server Enterprise for Windows NT.

A

Using *isql* to Run Scripts

Getting a Command Prompt

You run *isql* from the command line. To access the command line:

- Windows NT 3.51
Double-click the MSDOS icon in the Main program group in the Application Manager.
- Windows NT 4.0 and Windows 95
Choose Start→Command Prompt.

Tips

Some tips on running scripts:

- At the command line, run a script from the directory in which the script is located. Most of the scripts in this tutorial are located in the *tutorial* directory that you copied to your hard disk from the product CD or that you downloaded from the Web. Before you run the script, change directories to the correct directory with the `cd` command.
- It is easiest to run scripts if the DSQUERY environment variable is set to the name of the Adaptive Server you will be using. To find out if your DSQUERY environment variable is set correctly, type the following at a command line prompt:

```
echo %dsquery%
```

If the DSQUERY environment variable is not set to the Adaptive Server that you will be using, you can either set the environment variable to the value that you want or specify the server name as a parameter each time you run a script. For information on setting environment variables, see your operating system documentation.

How to Run Scripts

To run a script at the command line from the *tutorial* directory, type:

```
isql -Ulogin -Pstudentpassword -iscriptname -Sservername
```

where:

- *login* is the “student” login you create in this tutorial
- *studentpassword* is the password you create for the login “student”
- *scriptname* is the name of the script
- *servername* is the name of your Adaptive Server (an optional parameter)

Example

If you created the password “Ilearn” for the student login, and you want to run the script *tutor.sql*, follow these steps:

1. Go to the command line.
2. Change directories to the *tutorial* directory.
3. Type the following:

```
isql -Ustudent -PIlearn -itutor.sql
```

If you need to specify the name of your Adaptive Server, which is *PIANO*, type:

```
isql -Ustudent -PIlearn -SPIANO -itutor.sql
```

4. Press Enter.

If the script runs successfully, you will see messages about the number of rows affected.

For complete information about *isql*, see the *Utility Programs for Windows and Windows NT*.

Index

A

Adaptive Server
 environment 2-2
 starting 2-4
Aggregate column
 in InfoMaker crosstab 5-8
Application logic
 adding to a NetImpact Dynamo
 template 6-19
Application Programming Interfaces
 (APIs) 1-9
Applications (Adaptive Server
 Professional for Windows NT)
 starting 2-2
 where to find 2-2
Application server
 NetImpact Dynamo as 6-2
Arguments (NetImpact Dynamo)
 referring to in template 6-23
 supplying 6-22
Auto-Migrate (SQL Modeler) 3-14

B

Business rules
 associating with a database object in
 SQL Modeler 3-12
 associating with references in SQL
 Modeler 3-15
 cataloging in SQL Modeler 3-14
 enforcing 3-6

C

Cartesian products 3-5
Check constraints 3-7
 See also Domain checking
City-Type report
 developing in InfoMaker 5-6
clb_cat.sql script 4-15

Client/server, Sybase's
 configuration 1-8
Client-Library
 about 1-9
 called by *pbsync050.dll* (InfoMaker) 4-2
 used by ODBC interface 1-10
Client-side filters
 creating in InfoMaker 5-4
Columns
 defining in SQL Modeler 3-10
Component Integration Services 1-4
Components of Adaptive Server
 Enterprise for Windows NT 1-3
Connecting to Adaptive Server
 from InfoMaker 4-3
 from NetImpact Dynamo 6-5
 from SQL Modeler 3-9, 3-20
 from Sybase Central 2-6
Connecting to the MusicSite Web
 site 6-11
Connection Profiles
 creating for NetImpact
 Dynamo-Adaptive Server
 connection 6-5
 corefmt.ssc script (dbArts Web
 application) 6-33
Creating an InfoMaker application 5-11
Creating forms (InfoMaker) 4-15
Crosstab report, defined 5-6

D

Database devices
 creating with Sybase Central 2-9 to
 2-13
 logical names 2-9
 physical names 2-9
Database option truncate log on checkpoint
 setting for *eventsdb* in SQL
 Modeler 3-13
 setting for *tempdb* 2-14

- Database profile, creating in
 - InfoMaker 4-3
 - Databases, generating in SQL
 - Modeler 3-18
 - Data entry with InfoMaker 4-6
 - with Data Manipulation painter (grid) 4-8 to 4-10
 - with forms 4-15 to 4-18
 - Data source
 - definition in InfoMaker 4-11
 - dbArts, business problem of 1-2
 - dbArts Web application 6-29
 - how it works 6-32
 - using 6-31
 - Declarative referential constraints 3-6
 - as opposed to triggers 3-19
 - Display formats (extended attributes) 4-13
 - Displaying properties, in Sybase
 - Central 2-4
 - Domain checking 3-7
 - with a client-side validation style 4-13
 - DropDownDataWindow edit style (InfoMaker) 4-15
 - Dynamo script
 - about 6-19
 - codes delimiting 6-21
- E**
- Edit styles (extended attributes) 4-14
 - Event Listing report
 - creating in InfoMaker 5-2
 - eventsdb* database
 - generating in SQL Modeler 3-18
 - Extended attributes
 - display format 4-13
 - edit style 4-14
 - in InfoMaker 4-13
 - validation style 4-13
- F**
- Filters
 - client-side 5-4
 - server-side 5-9
- Foreign keys** 3-3
 - function of 3-3
 - migrating from primary keys in SQL Modeler 3-14
 - of *eventsdb* 3-3
- Forms**
 - creating in InfoMaker 4-15
- H**
- HTML (HyperText Markup Language)
 - defined 6-3
 - forms 6-26
 - NetImpact Dynamo generation of 6-33
- I**
- Importing data
 - in InfoMaker 4-9
 - using a data pipeline (InfoMaker) 4-10
 - Indexes
 - automatic creation of 3-7
 - defined 3-7
 - defining in SQL Modeler 3-16
 - InfoMaker
 - connecting to Adaptive Server 4-2
 - creating applications 5-11
 - deploying applications 5-12
 - functionality 1-4
 - library files 4-7
 - Install Builder program
 - for deploying InfoMaker applications 5-12
 - isql, using to run scripts A-1 to A-2
- J**
- JavaScript
 - browser support of 6-33
 - for client-side data checking 6-33

Joins 3-4

K

Key columns. *See* Keys.

Keys 3-3

 automatically migrating (SQL
 Modeler) 3-14

 foreign keys 3-3

 primary keys 3-3

L

Logging into Adaptive Server from
 Sybase Central 2-5

Logins

 “sa” login 2-5

 adding to Adaptive Server in Sybase
 Central 2-5

 creating “student” login for
 tutorial 2-8

 defined 2-5

M

master database 2-7

master device, checking size of 2-12

Methods (NetImpact Dynamo),
 defined 6-24

Model properties (SQL Modeler) 3-12

Monitoring the server 1-7

N

navbar.ssc script file (dbArts Web
 application) 6-33

Navigation bar in dbArts Web
 application 6-33

NetImpact Dynamo

 as application server 6-2

 connecting to Adaptive Server 6-5

 functionality 1-4, 6-2

O

Objects (NetImpact Dynamo) 6-19

ODBC (Open Database
 Connectivity) 1-9

ODBC data sources

 advanced settings for NetImpact
 Dynamo 3-9, 6-4

 defining for data pipeline 4-10

 defining for NetImpact Dynamo 6-4

 defining for SQL Modeler 3-9

 implications of incorrect settings 3-9,
 6-4

ODBC drivers

 installed with Open Client
 software 1-10

Open Client 1-9

Open Client libraries 1-9

P

Painter, in InfoMaker 4-6

pbsyc050.dll (Powersoft Dynamic Link
 Library)

 automatic installation of 4-2

 for deploying InfoMaker
 application 5-13

Personal Web Server

 configuring 6-11

 restarting 6-13

Physical data model (PDM) 3-8

 and SQL Modeler 3-8

 for *eventsdb* 3-11

Piping data, in InfoMaker 4-10

Powersoft Deployment Kit

 for deploying InfoMaker
 applications 5-12

Powersoft Repository

 about 4-2

 automatically created by
 InfoMaker 4-5

 populating 4-6

Powersoft stored procedures

 installing 4-3

 verifying installation of 4-2

Primary keys 3-3
 function of 3-3
 migrating to foreign keys in SQL
 Modeler 3-14
 of *eventsdb* 3-3
 Processes
 Adaptive Server 2-7
 examining in Sybase Central 2-7
 Properties (NetImpact Dynamo)
 defined 6-33

Q

Query-building tools
 SQL Query Editor (NetImpact
 Dynamo) 6-10 to 6-11
 SQL toolbox (InfoMaker) 5-7 to 5-9
 Query objects (NetImpact Dynamo)
 about 6-19
 using 6-24

R

RDBMS (relational database
 management system)
 advantages of 1-9
 References
 associating with business rules in SQL
 Modeler 3-15
 defining in SQL Modeler 3-14
 Referential integrity
 defined 3-6
 option in SQL Modeler 3-6
 Relationships, and data retrieval 3-4
 Reports
 creating in InfoMaker 5-2
 previewing 5-3
repos.sql script 4-6
 Retrieval arguments
 defining (InfoMaker) 5-10
 Revenues report, creating in
 InfoMaker 5-9
 Roles

assigning to “student” login for
 tutorial 2-9
 in Adaptive Server 2-5

Rows

describing one instance of entity 3-2
 inserting in InfoMaker 4-9

S

Scripts

See also individual script names.
 how to run A-1 to A-2
 type of NetImpact Dynamo
 object 6-32

select command 3-4

Servers, visible in Sybase Central
 interface 2-4

Server-side filters, creating in
 InfoMaker 5-9

SQL (Structured Query Language)
 defined 1-9

See also Transact-SQL

SQL Query Editor (NetImpact Dynamo)
 using to build queries 6-10

SQL syntax, examining in
 InfoMaker 5-7

SQL toolbox (InfoMaker) 5-7

Storage. See Database devices.

Stored procedures

defined 3-20

generating in SQL Modeler 3-20

Sybase Central

using for database administration 2-1

using for Web site generation and
 administration 6-5

Sybase client/server environment 1-8

sybssystemprocs database 4-3

T

Tables

creating in SQL Modeler 3-9

defined 3-2

tempdb

- defined 2-10
- enlarging 2-11
- increased activity in 2-10
- Templates
 - altering to receive parameters 6-28
 - foundation of NetImpact Dynamo
 - Web application 6-3
- Thin-client applications 6-2
- Transact-SQL 1-9
- Troubleshooting Web site errors 6-13
- truncate log on checkpoint 2-14
- tutor.pbl*, opening 4-7
- Tutorial, purpose of 1-1
- tutorial_dev* database device
 - allocating space on in SQL Modeler 3-13
 - creating 2-9
- tutorial* directory
 - directory containing tutorial scripts and files A-1
 - installing 2-3

- U**
 - undo.sql* script 7-1
 - Unique constraints 3-7

- V**
 - Validation styles (extended attributes) 4-13
 - Views
 - creating in SQL Modeler 3-16
 - defined 3-16
 - examining SQL for in SQL Modeler 3-18
 - for *eventsdb* 3-16

- W**
 - Web sites
 - generation of 6-5
 - importing 6-29
 - Web site tables stored in database 6-6

- Wizards, in Sybase Central 2-4

