

SQL Anywhere Integration with Visual Studio 2005

A whitepaper from Sybase iAnywhere

CONTENTS

Introduction	3
Server Explorer plug-in	3
Opening the Server Explorer	3
Visual Studio data sources.....	5
SQL Anywhere integration components.....	10
How applications can use the data adapter component	16
SQL Anywhere Data Adapter properties.....	18
Generating DataSets	20
Preview data.....	23

INTRODUCTION

SQL Anywhere 10 contains a number of integration features with Microsoft Visual Studio .NET (both 2003 and 2005). These features are designed to make it easier to work with a SQL Anywhere database while developing an application using Visual Studio. This whitepaper outlines the integration features that are present with Visual Studio 2005, and contains tutorials demonstrating how the integration features can be used to ease application development while working with a database.

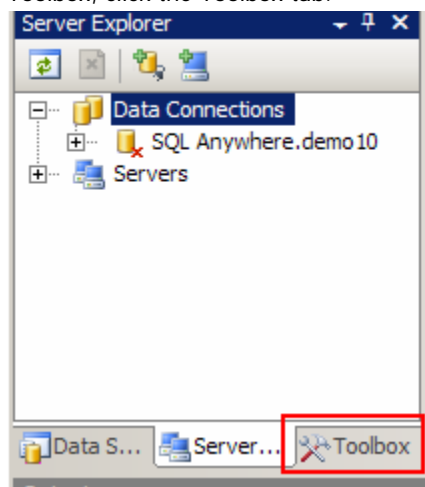
SERVER EXPLORER PLUG-IN

The Visual Studio Server Explorer can be used to display information about databases, such as their schema, and the data they contain. You can also modify the data stored in database tables using the Server Explorer.

OPENING THE SERVER EXPLORER

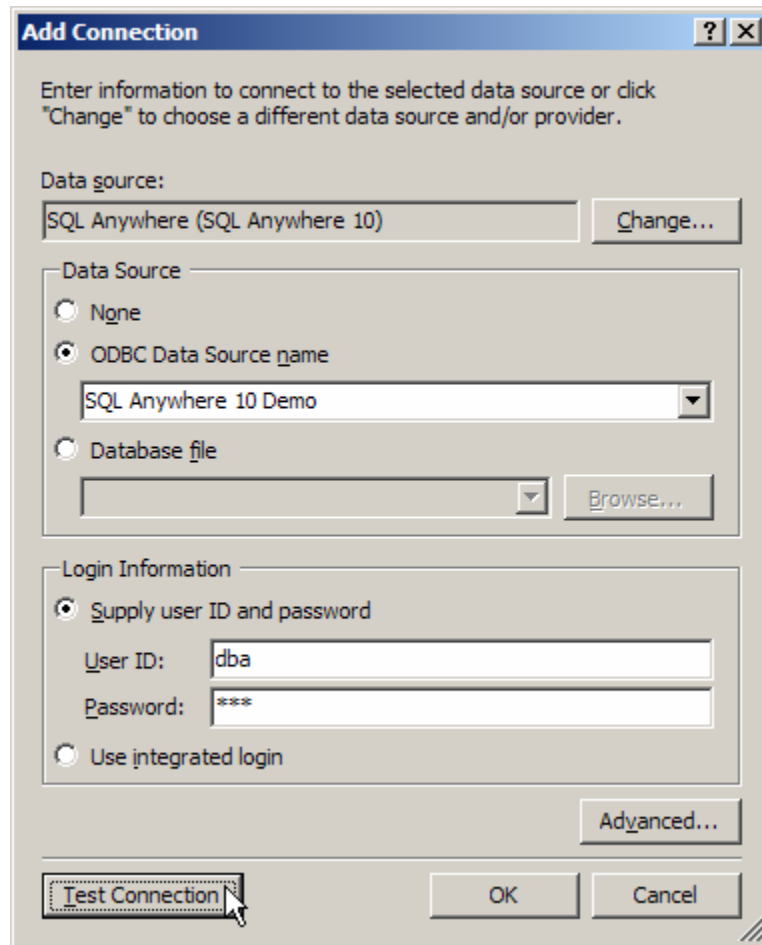
1. In Visual Studio, choose View > Other Windows > Server Explorer.

The Server Explorer appears on the screen, replacing the Toolbox. To switch back to the Toolbox, click the Toolbox tab.

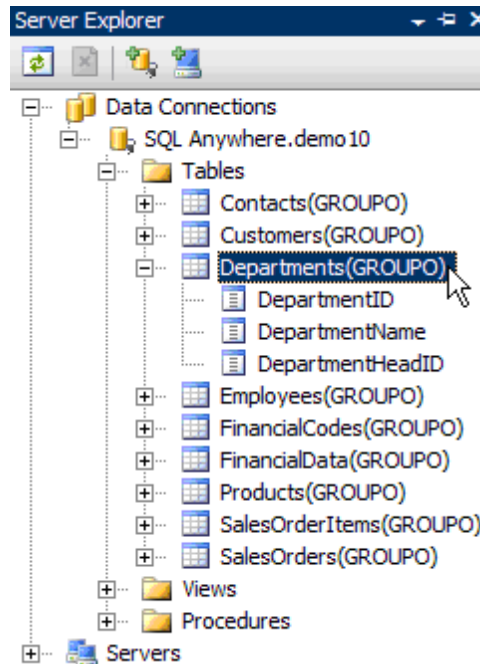


If the Server Explorer is docked to the side of your screen, tabs appear along the side.

2. Right-click Data Connections, and then choose Add Connection.
The Add Connection dialog appears.
3. If the Data Source is not set to SQL Anywhere (SQL Anywhere 10), click the Change button and select SQL Anywhere from the list.
4. Type **SQL Anywhere 10 Demo** in the ODBC Data Source name field.
5. In the User ID field, type **DBA**, and in the Password field, type **sql**.



6. Click Test Connection to test the supplied parameters.
A message box alerts you to a successful connection or any problems.
7. Click OK to add the connection.
The Server Explorer now displays the new connection—**SQL Anywhere.demo10**.
8. Expand the connection and the Tables entry below it.
The Server Explorer shows you all the tables that are in your database. To view the schema for one of the tables, click the + beside its name. For example, expand the Departments table to look at its schema:



The department table contains three columns: DepartmentID, DepartmentName, and DepartmentHeadID.

9. To view the data stored in this table, right-click the table name, and choose Show Table Data. The main area in Visual Studio displays the data. Click in a cell to edit the data in that cell. Any changes that you make are automatically committed to the database.
10. To add a new row of data to the table, enter the data in the row that has * (asterisk) to the left of it.

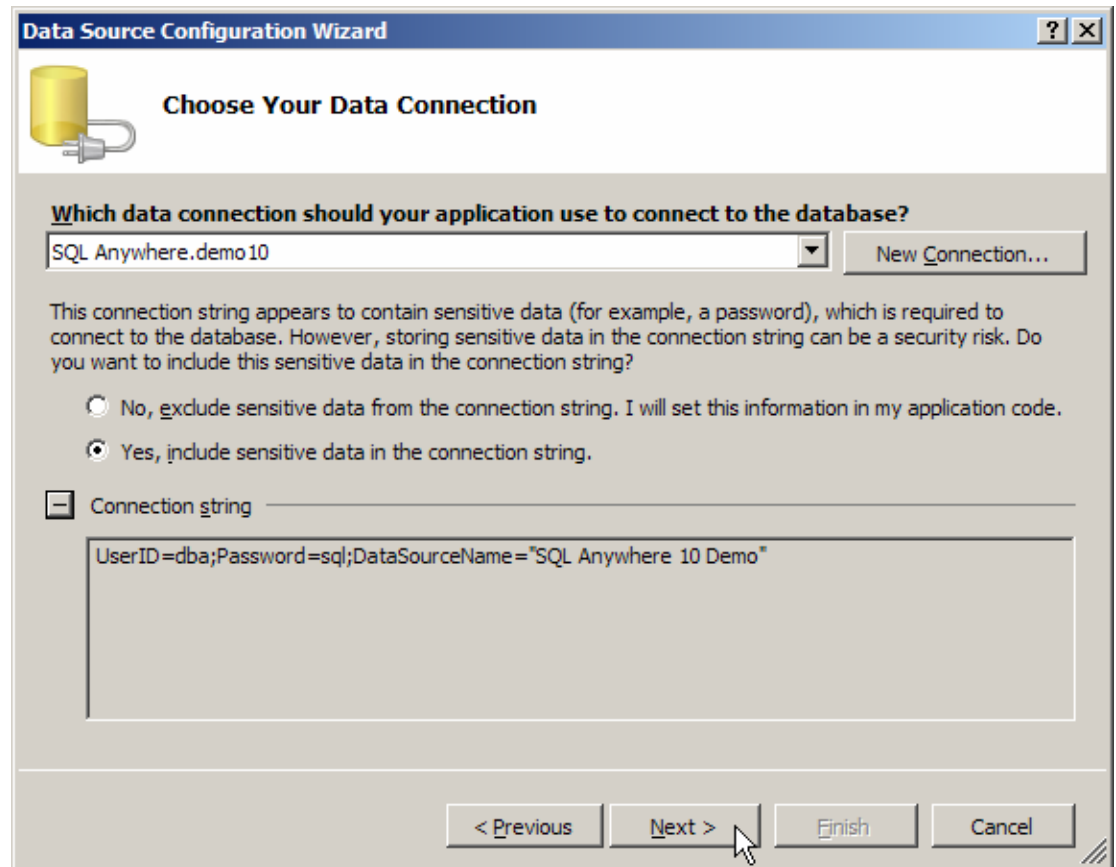
	DepartmentID	DepartmentName	DepartmentHe...
▶	100	R & D	501
	200	Sales	902
	300	Finance	1293
	400	Marketing	1576
	500	Shipping	703
*	NULL	NULL	NULL

VISUAL STUDIO DATA SOURCES

Visual Studio can maintain a list of data sources for your application.

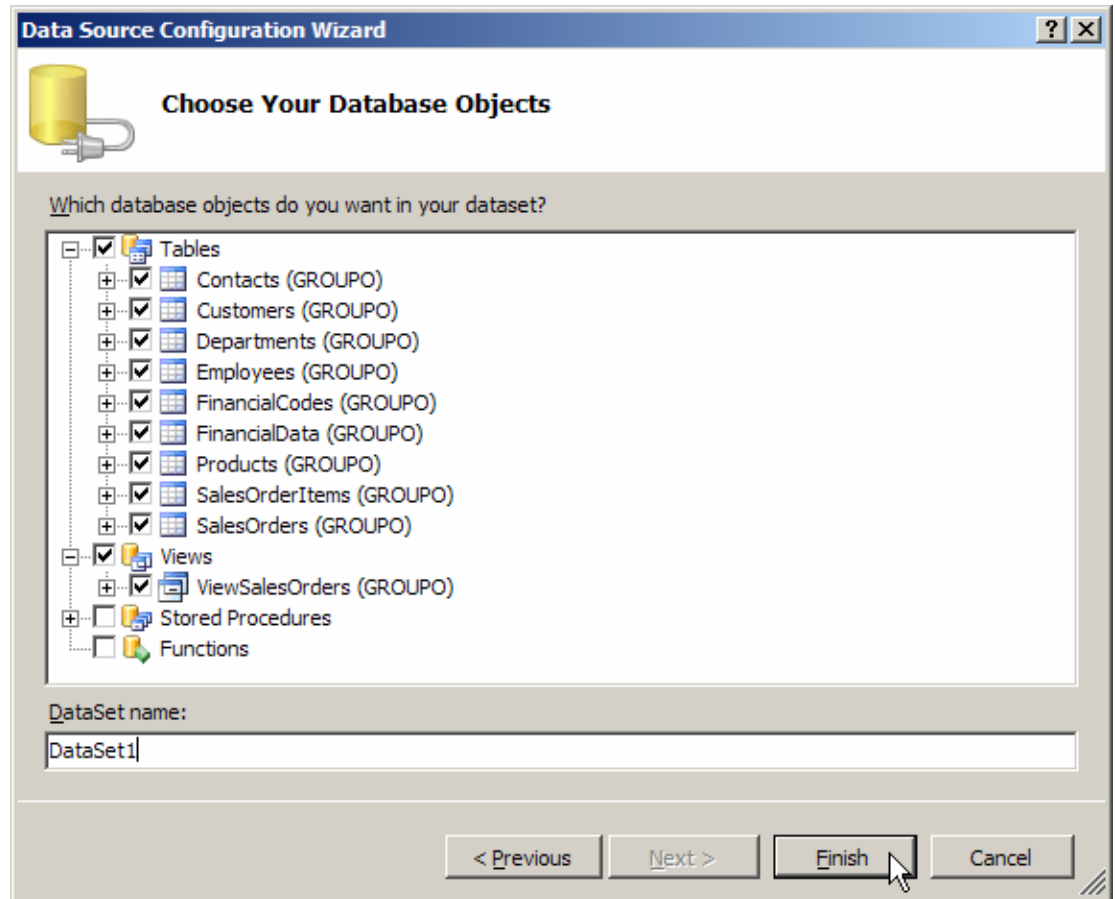
1. Create a new project:
 - a. Choose File > New Project.
 - b. Under Visual C# or Visual Basic, select Windows Application.
 - c. Click Form1.
2. To open the list of data sources, or connect to a new data source, choose Data > Show Data Sources.
3. Click Add New Data Source.
The Data Source Configuration wizard appears.
4. Select the Database option, and then click Next.

5. The SQL Anywhere data connection you created in the Server Explorer is available. The wizard informs you that the connection string contains sensitive data. Click the + beside Connection String to view the connection string. You should see that it includes a user ID and password. Since these are the default settings for the database, you can store them in the connection string. Select Yes, Include Sensitive Data In The Connection String. Click Next.



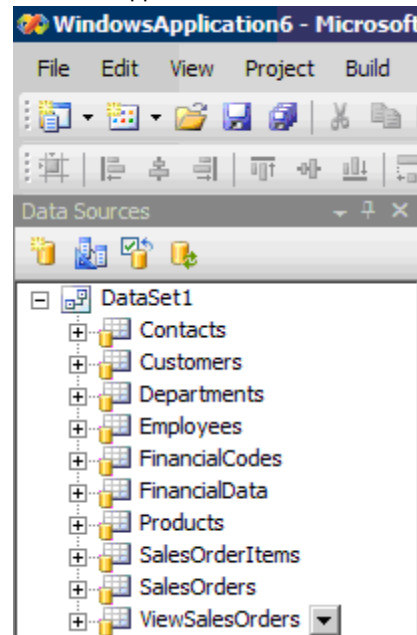
6. Use the default name **ConnectionString**. Click Next.

7. Include all of the tables and views in the dataset.

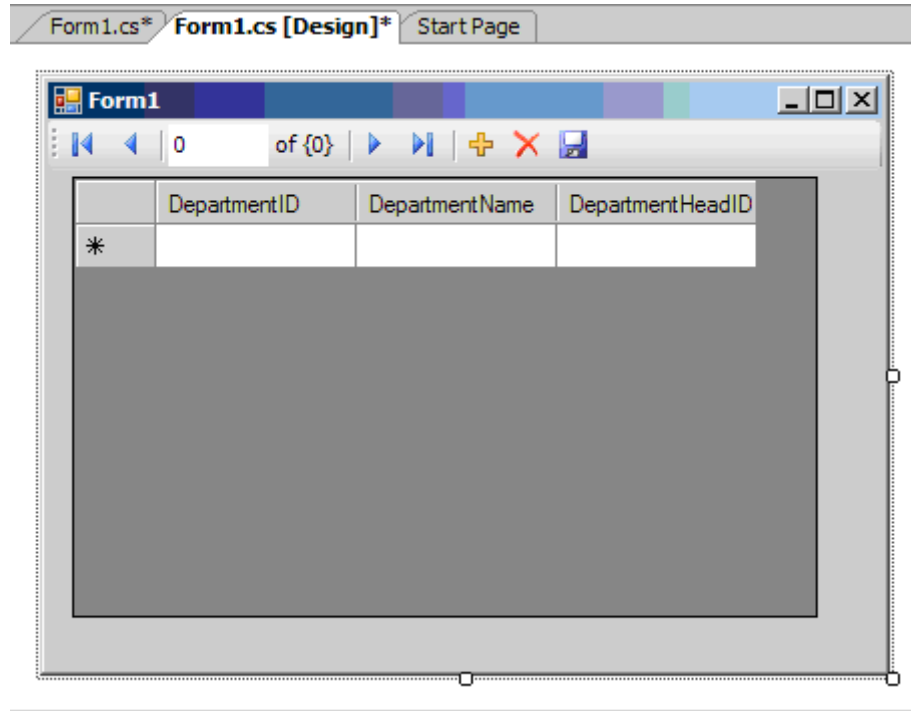


8. Name the DataSet. Use the default name **DataSet1**. Click Finish to close the wizard and create the new data source.

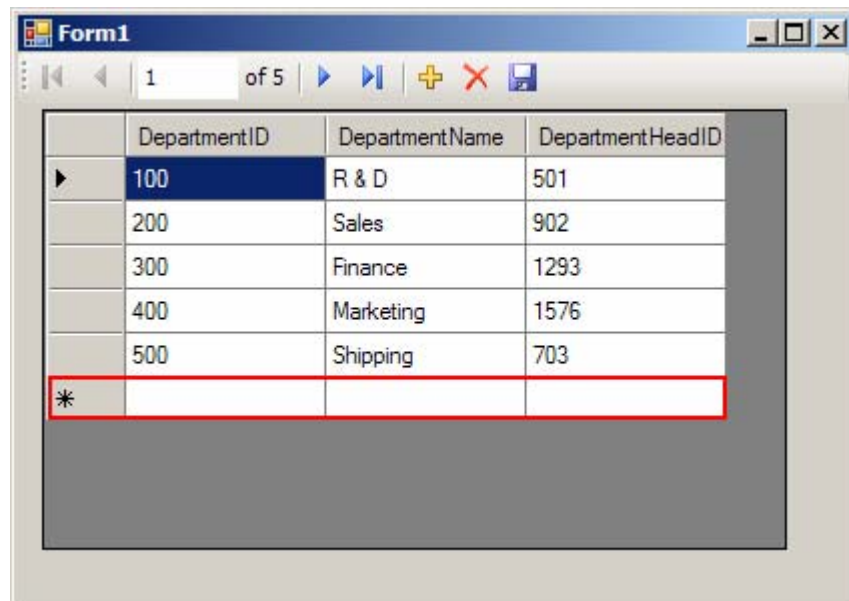
DataSet1 appears in the list of Data Sources:



- Elements from the DataSet can be dragged and dropped onto your form. For example, drag the Departments table onto your form. Visual Studio automatically creates the necessary bindings and table adapters, and supplies you with a graphical control that you can use to interact with the Departments table.




- Compile and run the application by choosing Debug > Start Debugging. The table is filled with data from the database.



- Add a new row by typing in the row with the asterisk beside it, or by clicking the + icon in the toolbar. By default, all of the data can be edited and saved to the database.
- To change the behavior of this control, stop the application, and view the Properties in Visual Studio for the data grid. To disallow a user from editing the content, set the ReadOnly property to True.

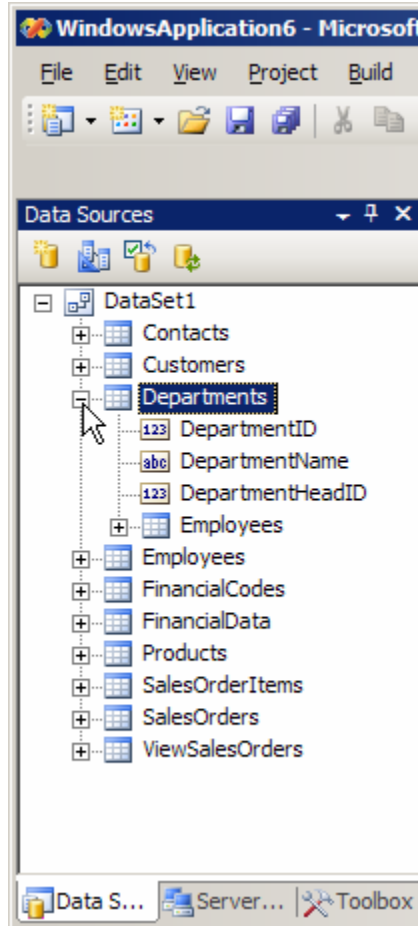
Properties

departmentsDataGridView System.Windows

GenerateMember	True
GridColor	 ControlDark
ImeMode	NoControl
Location	12, 28
Locked	False
Margin	3, 3, 3, 3
MaximumSize	0, 0
MinimumSize	0, 0
Modifiers	Private
MultiSelect	True
ReadOnly	False
RightToLeft	No
RowHeadersBorderStyle	Raised
RowHeadersDefaultCellStyle	DataGridViewCellStyle
RowHeadersVisible	True

[Edit Columns...](#), [Add Column...](#)

13. The Data Sources tab also allows you to view the contents of your DataSet. For example, expand the Departments table by clicking the + beside it.



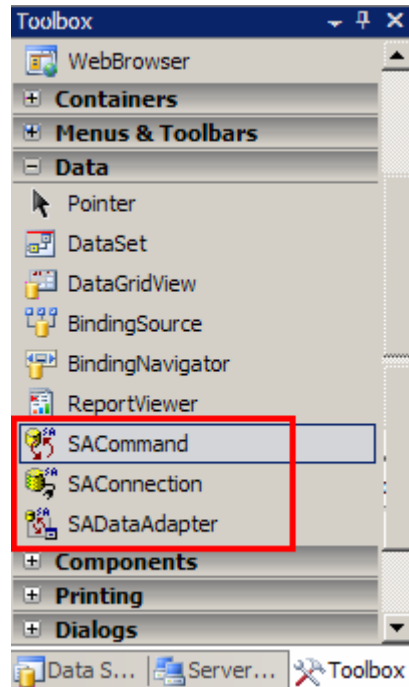
This view shows you the columns that are in that table. It provides another quick way that you can view the table schema from within Visual Studio. If more control is needed, you can also launch Sybase Central and Interactive SQL from within Visual Studio by choosing Sybase Central or Interactive SQL from the Tools menu.

SQL ANYWHERE INTEGRATION COMPONENTS

The SQL Anywhere integration components allow application designers to interact with a database both from within the Visual Studio development environment, as well as from within their code.

The SDataAdapter component is of particular interest. It allows an application designer to retrieve result sets from the database easily and store them in a DataSet object (DataSet). The data in the DataSet object can be edited, and the SDataAdapter automatically updates the database with the changes.

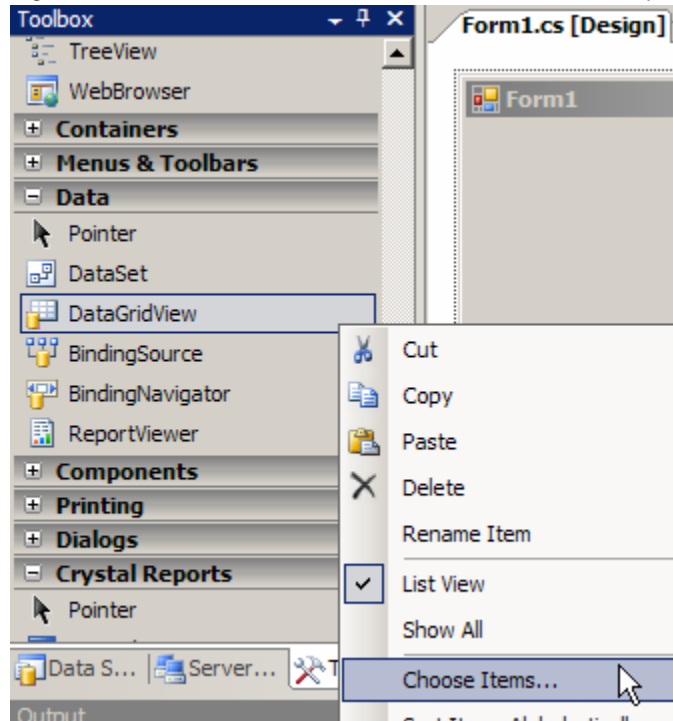
The SDataAdapter object is located in the Toolbox, along with the other SQL Anywhere integration components, SConnection and SCommand. Expand the Data section to display the data components:



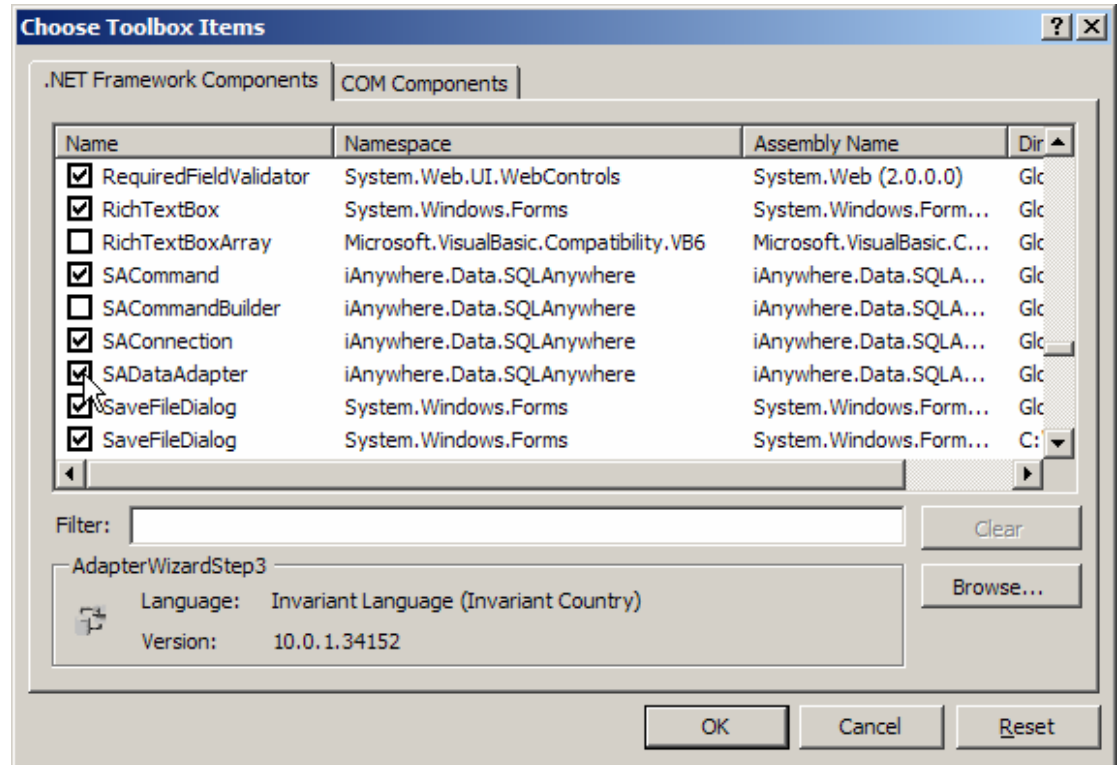
If you do not see the SQL Anywhere components, you must add them manually. You can skip the following procedure if you already have the SQL Anywhere components listed in the Data section of your Toolbox.

To add the SQL Anywhere Integration Components

1. Right-click the Toolbox, and choose Choose Items from the popup menu.



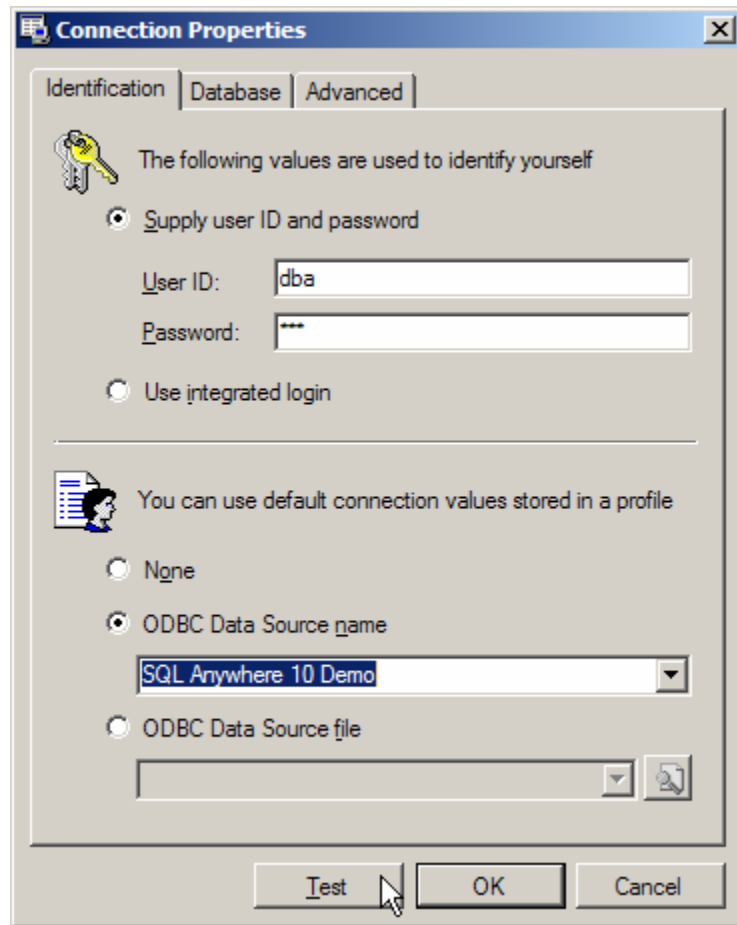
2. In the list of available components, select SACommand, SAConnection, and SADATAAdapter, and then click OK.



To use the SADATAAdapter component

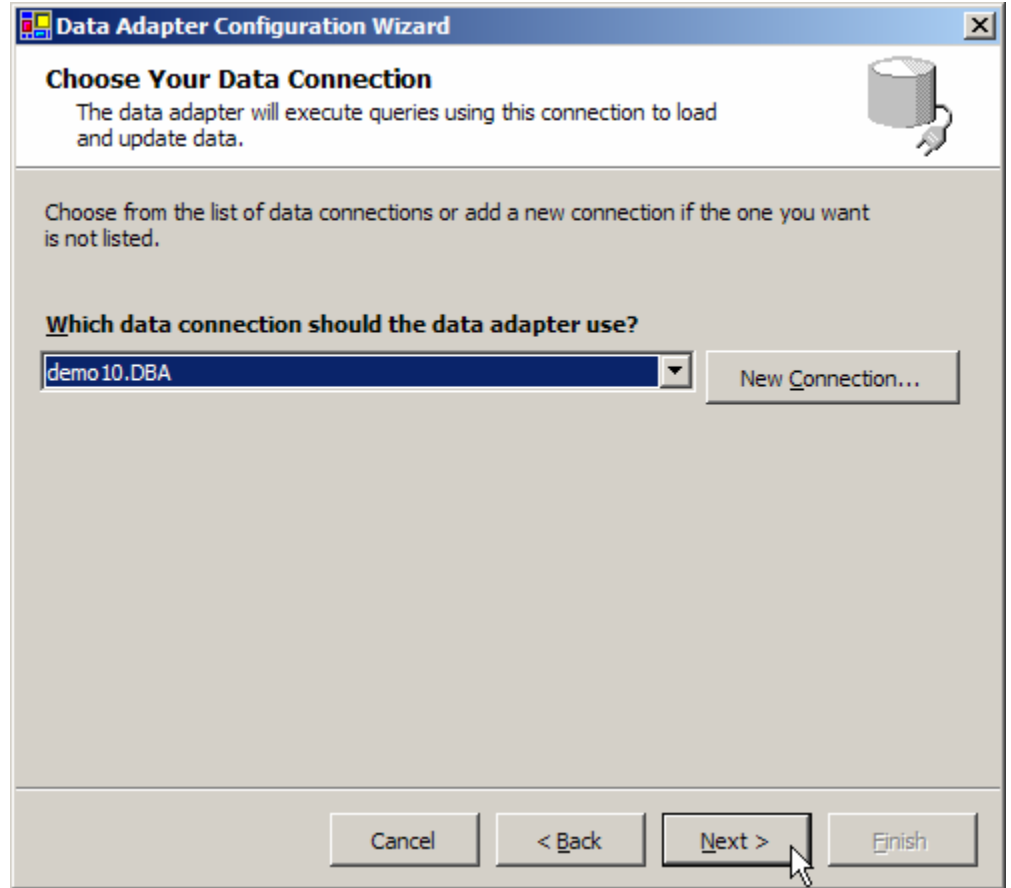
1. Click and hold the SADATAAdapter component in your Toolbox, and then drag and drop it onto your form.
An saDataAdapter1 object appears, and the Data Adapter Configuration wizard appears.
2. Click Next in the Data Adapter Configuration wizard.
3. Choose the connection you want the saDataAdapter to use. If you already have a connection available, you can use that connection, or you can click New Connection to set up a new connection.

4. Type **DBA** in the User ID field, **sql** in the Password field, and **SQL Anywhere 10 Demo** in the ODBC Data Source Name field.



5. Click Test to test your connection. If everything was entered correctly, the message Connection Successful appears.

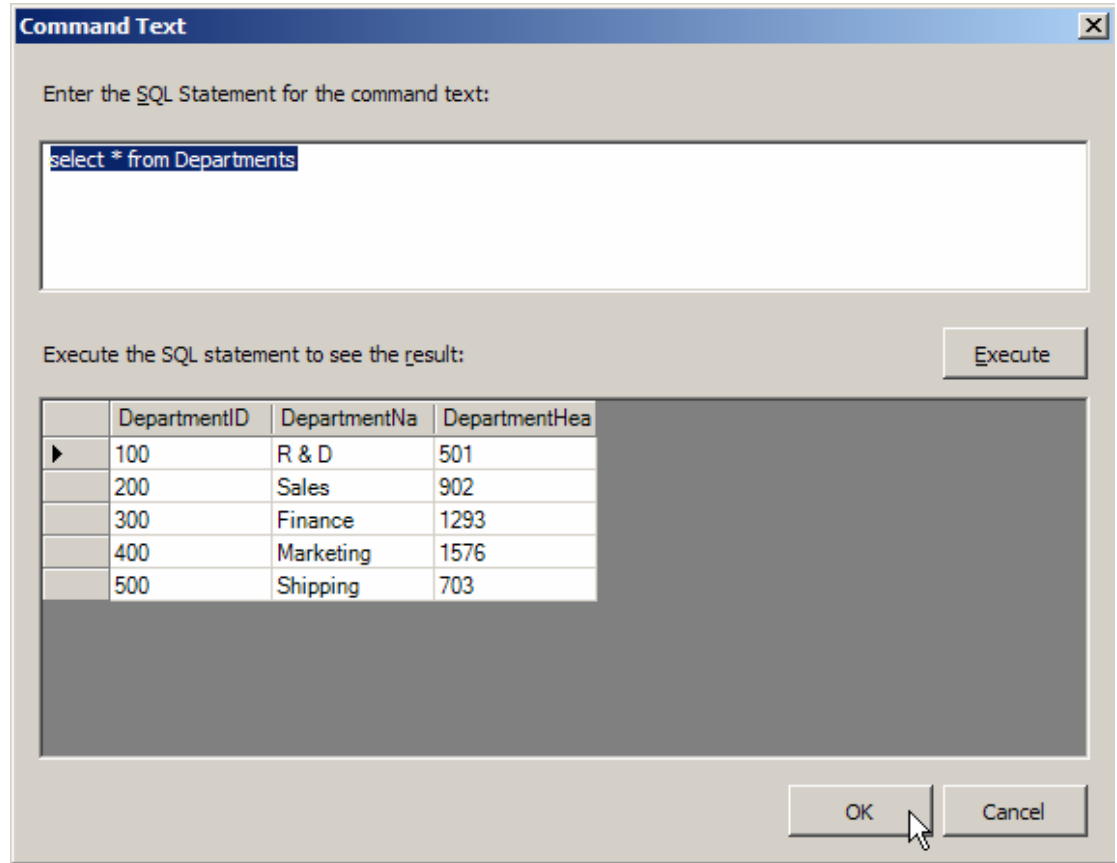
6. Click OK to confirm the connection and return to the Data Adapter Configuration wizard screen, where your new connection is selected. Click Next to continue.



7. The wizard asks how the saDataAdapter should access the database. Use the default option Use SQL Statements. Click Next.
8. On the next screen, specify the SQL statement to use to load data into the data adapter. Click Query Builder to access the Command Text dialog. This dialog helps you make sure your SQL statement is correct.
9. For this tutorial, you are loading the contents of a table called Departments into your DataSet. Enter the following statement, and then click Execute:

```
SELECT * FROM Departments
```

10. Query Builder displays the result of your query—the data from the Departments table.

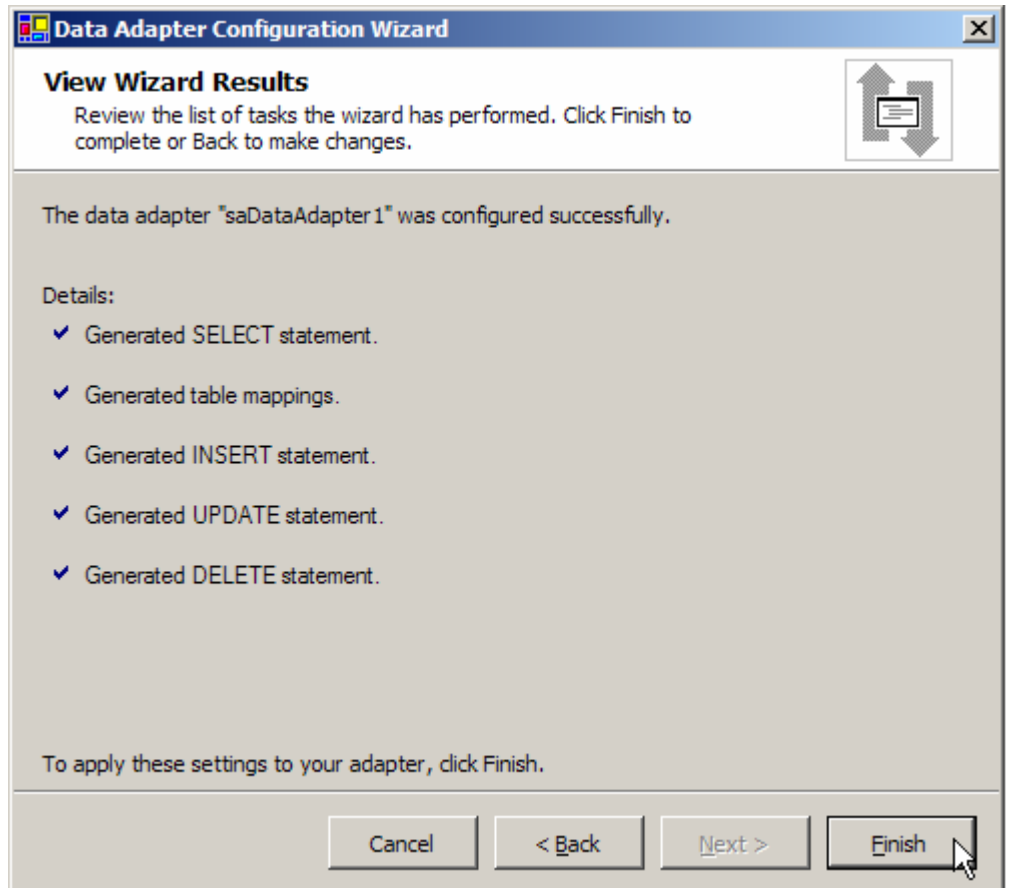


11. Click OK.

The query you wrote appears in the wizard.

12. Click Next.

The wizard presents you with a status screen, showing the results of the Data Adapter creation. If the creation process was successful, the dialog looks like the following example.



13. Click Finish to close the wizard.

HOW APPLICATIONS CAN USE THE DATA ADAPTER COMPONENT

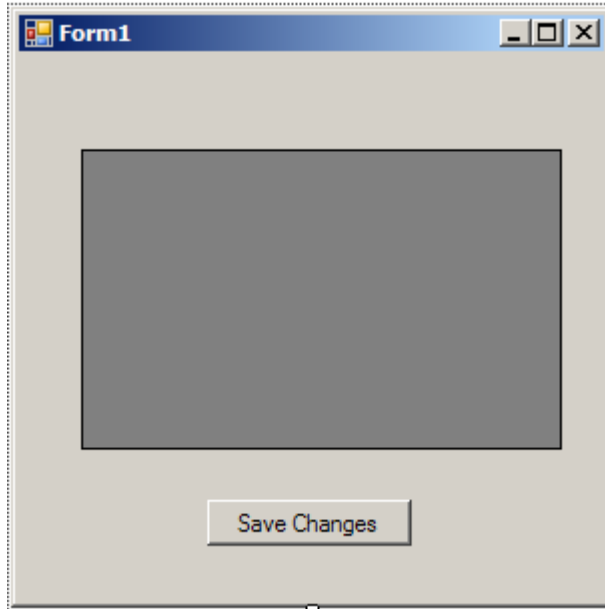
Now that you have created an `saDataAdapter` object, you can use it in your application. The easiest way to do this is with a `DataGridView` object.

Note: When you use the `saDataAdapter`, no locks are placed on the rows in the database. This means that there is the potential for conflicts to arise when you apply changes from the `DataSet` to the database. Your application should include logic to resolve or log conflicts that arise. You can find more information about this in the SQL Anywhere documentation.

To configure an application to use the `saDataAdapter` Component

1. In the Toolbox, click the Windows Forms section, and then click `DataGridView`.
2. Drag the `DataGridView` object onto your form.
3. Drag a Button to your form.
4. Right-click the button and choose Properties.

5. Change the button text to **Save Changes**.



6. Now that the saDataAdapter is loaded with data, and the DataGridView placed on your form, you can link the DataSet generated by the saDataAdapter to the contents of the DataGridView.

Place the following code in the `global` declarations section of your form to create a DataSet object:

```
[C#]
DataSet ds = new DataSet();
[Visual Basic.NET]
Dim ds As New DataSet
```

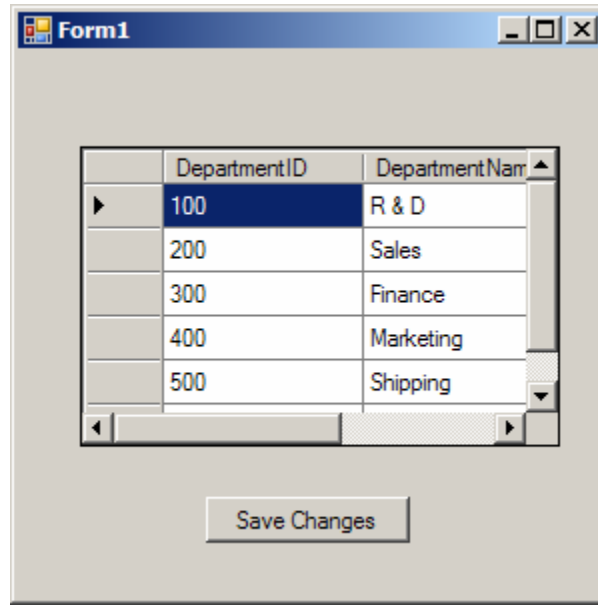
Place the following code in the `Load` method of your form:

```
[C#]
saDataAdapter1.Fill(ds);
dataGridView1.DataSource = ds;
dataGridView1.DataMember = "Departments";
[Visual Basic.NET]
DataAdapter1.Fill(ds)
DataGridView1.DataSource = ds
DataGridView1.DataMember = "Departments"
```

7. For the saDataAdapter to save its results, you must configure the button to save the updated data. Place the following code in the `Click` method of the button:

```
[C#]
saDataAdapter1.Update(ds);
[Visual Basic.NET]
saDataAdapter1.Update(ds)
```

8. Run the application.
You can edit the data on the form. Clicking Save Changes saves your changes back to the database.

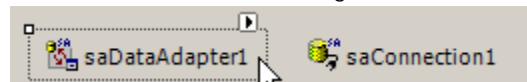


SQL ANYWHERE DATA ADAPTER PROPERTIES

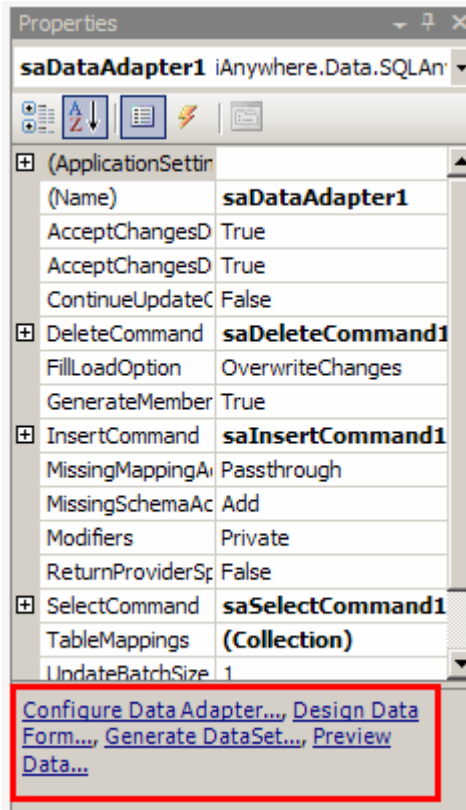
After an saDataAdapter is created, you can use built-in wizards to view its current configuration, or update it to alter its behavior. This section assumes you already have an saDataAdapter present in your application.

To view and modify data adapter properties

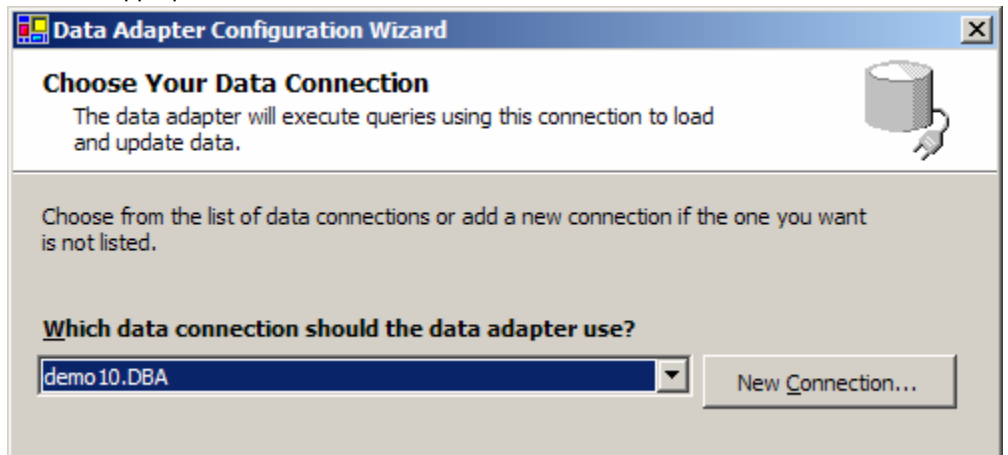
1. At the bottom of the form design window, click the saDataAdapter1 object.



Selecting the object causes its properties to appear in the Properties window. These properties can be configured manually. However, at the bottom of the Properties window, there are links to wizards that can help you modify the settings.

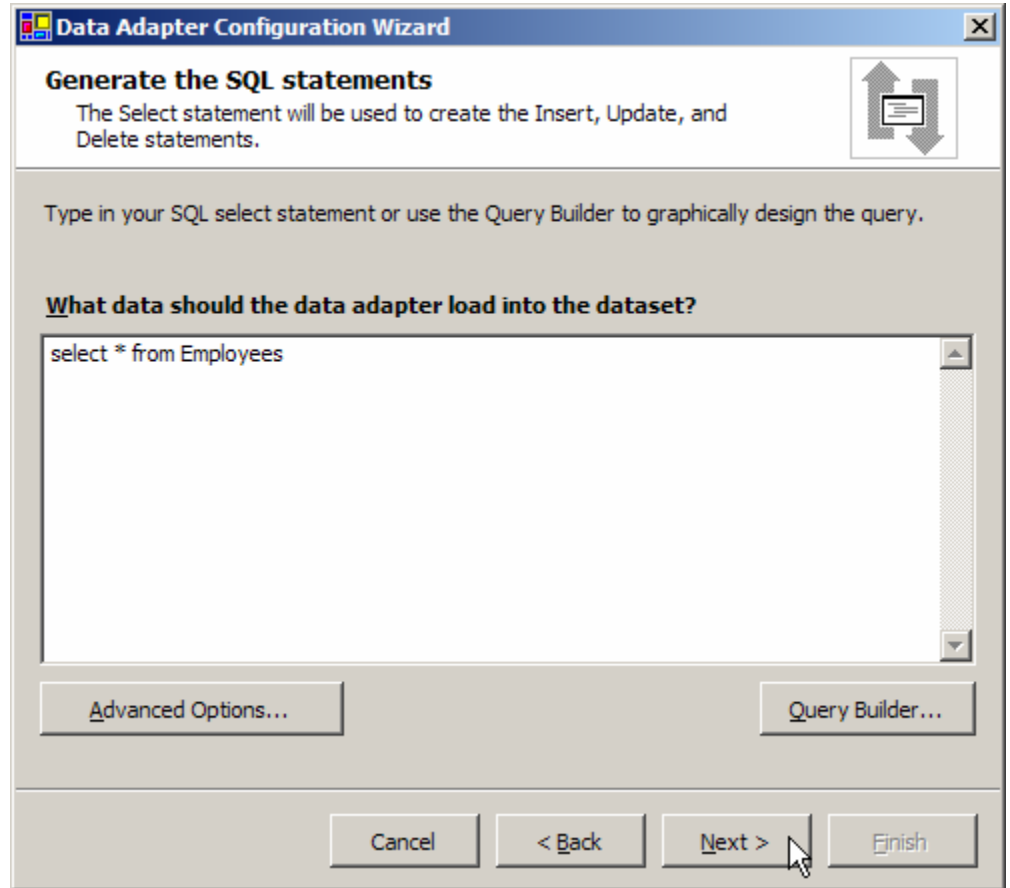


2. Select Configure Data Adapter in the Properties window. Alternatively, right-click saDataAdapter1, and choose Configure Data Adapter. This link allows you to modify all the settings that were originally set when you dropped the Data Adapter into your project. When you click this link, a Welcome dialog appears.
3. Click Next.
4. Since you already have a connection to the sample database, you do not need to choose another one. Click Next.
If you want to use another database, click New Connection to create a new connection, and enter the appropriate information.



5. You are asked whether you want to use SQL statements or existing stored procedures to access the database. Use the default option (SQL Statements). Click Next.

- You are asked which data the Data Adapter should select from the database. Your previous query, `SELECT * FROM Departments`, appears as the default. Change the statement to `SELECT * FROM Employees`.



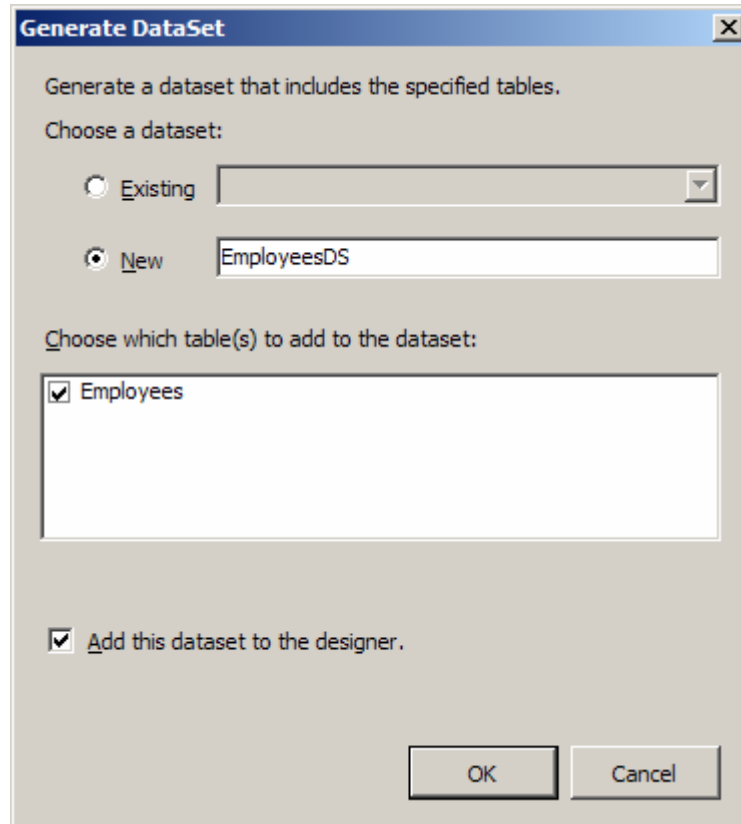
- You can use the Query Builder to verify that the query is correct. If the proper data is being returned, click Next, and then click Finish to confirm the changes.

GENERATING DATASETS

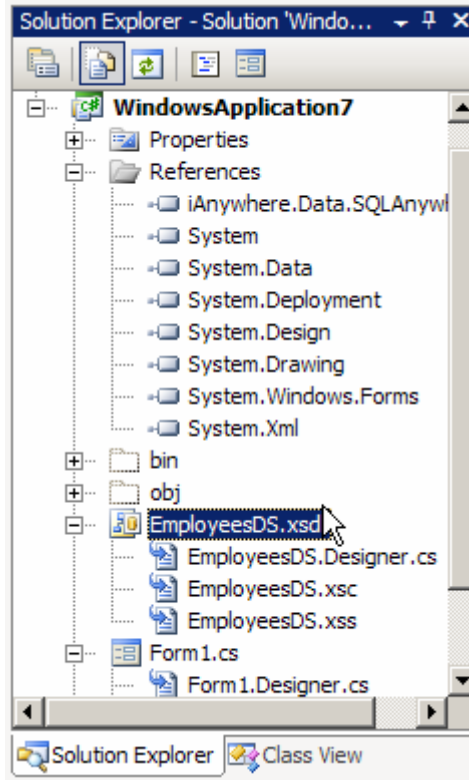
The Generate DataSet link generates an XML Schema Definition (.xsd) file. An XML Schema Definition file is a standard way to describe the structure of a database. This file can be easily deployed with your application, and used to create a DataSet object (DataSet).

To create an .xsd file

- At the bottom of Properties window for saDataAdapter1, click Generate DataSet. Alternatively, right-click saDataAdapter1, and choose Generate DataSet.
- You can specify either to generate an existing DataSet (Existing), or to generate a new one (New), in the Generate DataSet dialog. Since you have not created any DataSets yet, you must choose to generate a new one. Select New, and type **EmployeesDS** as the name for the DataSet.



3. In the Choose Which Table(s) To Add To The Dataset list, you see only the Employees table, since this is the only table that the saDataAdapter references. If you built a custom query for the saDataAdapter, then the list of tables you have access to may be different.
4. Click OK.
The *.xsd* file is generated, and appears in the Solution Explorer.



You can view the code that was automatically generated by double-clicking the *Employee.Designer.cs* file.

5. Your newly created DataSet object allows you to create datasets that are pre-configured with your database schema. For example, if you wanted to find out the type of the SocialSecurityNumber column, you could refer to it in this manner:

[C#]

```
EmployeesDS myEmployeeDataSet = new EmployeesDS();
MessageBox.Show(myEmployeeDataSet.Employees.SocialSecurityNumberColumn
.DataType.ToString());
```

[Visual Basic.NET]

```
Dim myEmployeeDataSet As New EmployeeDS
MessageBox.Show(myEmployeeDataSet.Employees.
SocialSecurityNumberColumn.DataType.ToString)
```

In the code sample above, myEmployeeDataSet is the DataSet name, Employees is the table you want to access, and SocialSecurityNumberColumn is the name of the column (followed by the word Column) that you want information about.

6. You can also use your DataSet object to hold data from the database. Using the DataGrid you created earlier, you can use the following code to fill the DataGrid:

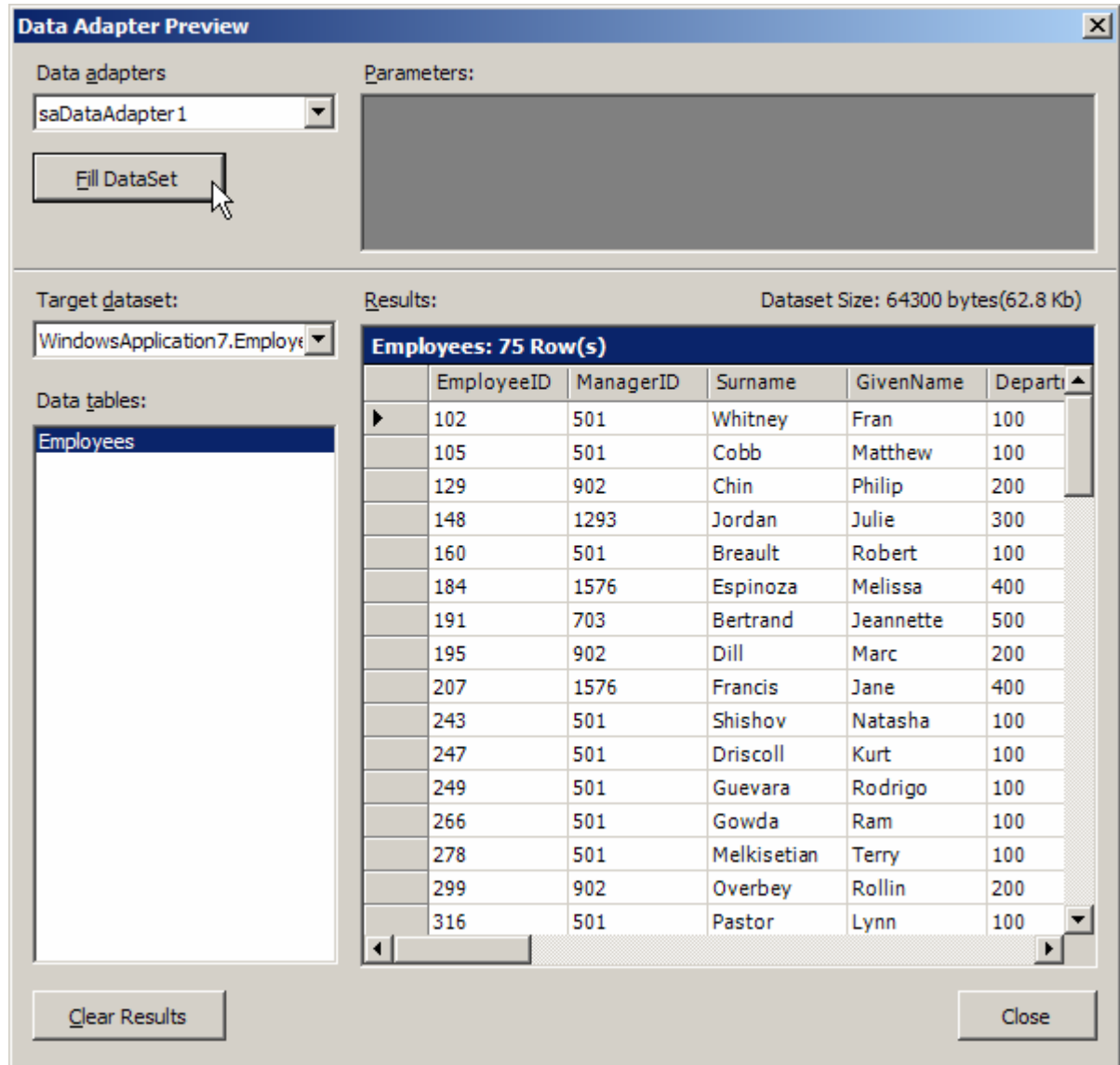
[C#]

```
EmployeesDS myEmployeeDataSet = new EmployeesDS();
saDataAdapter1.Fill(myEmployeeDataSet);
dataGridView1.DataSource = myEmployeeDataSet;
dataGridView1.DataMember = "Employees";
```

```
[Visual Basic.NET]
Dim myEmployeeDataSet As New EmployeesDS
saDataAdapter1.Fill(myEmployeeDataSet)
DataGridView1.DataSource = myEmployeeDataSet
DataGridView1.DataMember = "Employees"
```

PREVIEW DATA

The Preview Data link in the saDataAdapter1 properties area allows you to preview the data that the saDataAdapter will return. When you click the link, a dialog appears where you can select what data adapter you want to use, and what tables should be returned. Clicking Fill DataSet causes the main window to be populated with the data.



Previewing data is a good way to ensure that you are selecting the correct data for your application.

