CHAPTER 2

A Big-Picture Overview

Eschew the monumental. Shun the Epic. All the guys who can paint great big pictures can paint great small ones.

—Ernest Hemingway

Having a clear overview of how a BI solution is constructed can be one of the most important tasks to ensure a BI solution's success. To understand how a BI solution works, it is important not only to understand its individual projects but to comprehend how these projects integrate into a solution. Jumping into the intricate details without having a full understanding of where each piece of the puzzle fits is setting yourself up for failure. In other words, before starting work on any part of the solution, you need to see and comprehend the big picture.

The process of learning to create a BI solution is not much different. Therefore, to avoid the mistake of jumping in to create the individual projects that make up the solution, in this chapter we walk you through an entire BI solution from start to finish. You will see how each component is integrated and how they function together as a complete solution. Later, as you progress through the other chapters of this book, you will delve deeply into each of the component projects. This overview will help you understand the big picture.

The 10,000-Foot View

To start, let us list the steps that you will be performing in this solution. You begin building the solution by looking at the solution requirements and isolating the data you will be working with. You then move onto documenting the requirements and building your data warehouse. When the data warehouse is complete, you fill it up with data using a SQL Server Integration Service (SSIS) package. After filling the data warehouse, you create a cube and finally a report against the cube you have created.

Figure 2-1 shows a representation of these components. There are icons in the upper left of the figure representing the original source of the data. These original sources may be database tables or files, but in either case, you must review these objects in order to isolate the data you need for your particular BI solution. Afterward, move the data from its original source location into a data warehouse database.

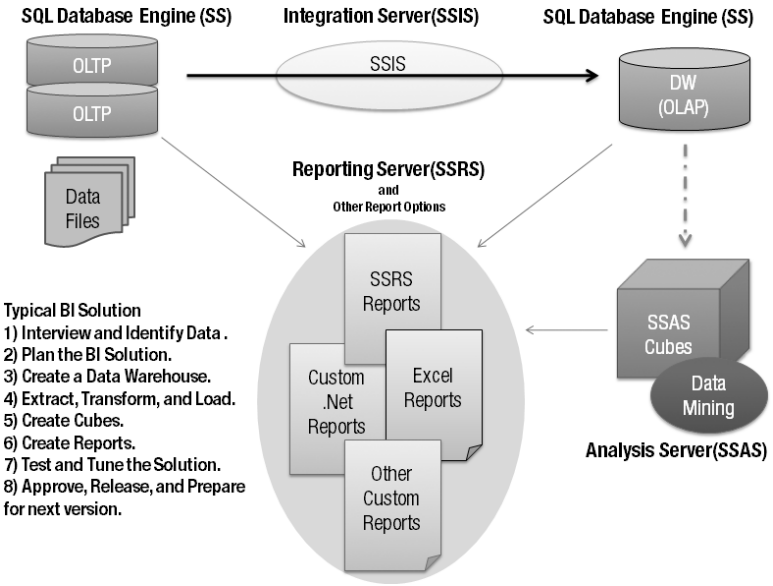


Figure 2-1. *A BI solution overview*

The data warehouse you create is designed on the principles of an online analytical processing (OLAP) style of database utilizing dimensional and fact tables. This is a different style of design than the online transactional processing (OLTP) databases that most developers are familiar with. The design differences are based upon their purpose. Databases that focus on gathering new data are designed around the OLTP format. The OLAP format focuses on providing information from existing data. As you will see later, you work with both OLTP and OLAP databases in a BI solution. OLAP databases come in two common forms: relational databases and cube databases. Relational databases use tables to contain the reporting data, while cube databases use cubes instead. This makes sense when you remember that the terms *tables* and *relations* are synonymous in database terminology.

In a BI solution, data warehouses are created using relational databases in an OLAP format. Nevertheless, you may also create an OLAP cube in addition to the data warehouse. Note that in Figure 2-1, we have displayed this connection between these two objects with a dotted line, indicating that the cube database represents an optional component.

Not all BI solutions need a cube database. In fact, many companies choose to create reports using the data warehouse alone. In Figure 2-1, the thin lines from the data warehouse to the reporting options represent this standard scenario. In addition, it is still possible to pull report data from the original online transaction-processing (OLTP) databases when needed, indicated in Figure 2-1.

The data warehouses and cubes provide additional options that make these structures desirable. For instance, because SSAS cubes host data mining capabilities, you can pull data mining results to your reports through your cubes. Another advantage of having a cube is that a variety of reporting applications are available designed to work with cubes alone.

Interviewing and Isolating Data

In any BI solution, the first course of action is interviewing the client or company owner that needs the solution. Because we do not have real life clients to interview, we describe the scenario here. Let's consider the following as our letter of engagement:

Dear Consultant,
I need reports that will give me information about weather patterns. Currently, I have been collecting data in the format shown in Table 2-1. I track the dates, the maximum and minimum temperatures, and the events of that day. Could you please create an example of what you do for customers like me?

Sincerely,
A Typical Client

And as the letter promises, Table 2-1 shows an example of the data.

Table 2-1. *The Data in the WeatherHistory.txt File*

Date	Max TemperatureF	Min TemperatureF	Events
1/23/2011	48	43	Rain
1/24/2011	51	46	Rain
1/25/2011	52	38	Rain/Sun
1/26/2011	53	41	Rain/Sun
1/27/2011	50	37	Fog

The client has not provided much detail, which is consistent with what you are likely to see in a real-life scenario. Yet once you review the data, you will find you have enough to get started. Besides, creating the prototype solution is often better than asking for more details when you are first trying to understand a client's needs. You are more likely to understand what questions to ask and be able to extract more information from the client in a second interview after you have created a simple prototype.

If you want to understand what is needed in a BI solution, start by understanding its data. For example, look at the range of values and data types noted in Table 2-1. You can see under the date column, for example, that the customer is using days, months, and years, but not hours or seconds. You can see whole values without decimal points under the maximum temperature column. You can also see that the client is using text descriptions in the Events column.

These facts give you vital clues about what your solution can accomplish. For instance, you will be able to create reports that tell you it was raining on a particular day, but not whether it was raining at noon on that day.

Once you have evaluated the data and identified what is available, you can begin the planning phase for the solution.

Plan the Solution

In each BI solution, you should create a document describing what you are trying to accomplish. Creating this document is the first part of the planning phase.

You also need to decide on a place to store your documentation. This location should be readily accessible to any team member working on the project. In this book, we use a subfolder in a Visual Studio solution folder as our document repository. This is convenient, because we are going to create several Visual Studio projects, and each of these projects will be added to the same Visual Studio solution as our documentation folder. Once complete, all of the projects and the documentation that defines those projects will be included under a single Visual Studio solution folder on the hard drive.

Creating Planning Documents

We created two tables (Tables 2-2 and 2-3) that document information about the client's data and what we know about it so far.

Table 2-2 lists the data source combined with descriptive names in one column and the data types in the other. Because all the data is coming from a text file rather than an existing database table, the data types are all strings.

Table 2-2. *Documenting the Source*

Data Source	Source Data Type
FlatFile.Date	String
FlatFile.Max TemperatureF	String
FlatFile.Min TemperatureF	String
FlatFile.Events	String

In Table 2-3, you see a listing of the destination columns, destination data types, any transformations we can expect to use, and an example of the outcome of those transformations. The purpose of this is to document the design of the destination tables, so we have listed the appropriate data types.

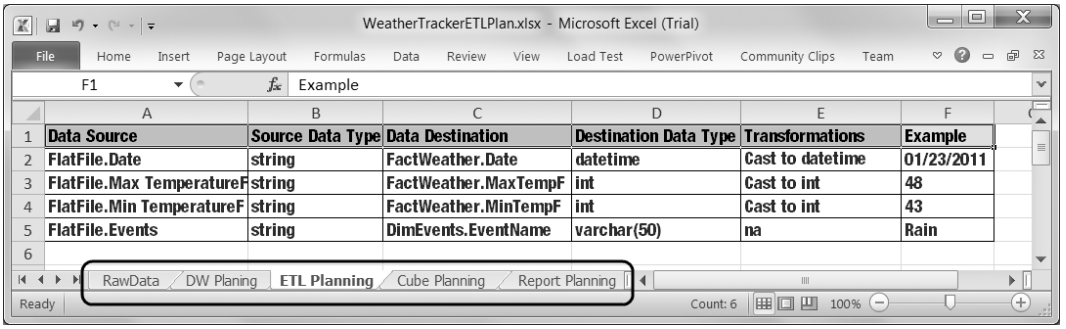
Table 2-3. *Documenting the Destination*

Data Destination	Destination Data Type	Transformations	Example
DimDates.DateName	datetime	add zero as needed and cast to datetime	01/23/2011
FactWeather.MaxTempF	int	cast to int	48
FactWeather.MinTempF	int	cast to int	43
DimEvents.EventName	varchar(50)	n/a	Rain

We often informally record source and destination information using a Microsoft Excel spreadsheet. From this informal evaluation, we then proceed to create more formalized documents toward the end of the solution life cycle. The formal documents will become a part of the BI solution we deliver to a client, while the informal spreadsheet is for development.

One advantage of using Excel is that it may be used to outline many parts of the solution using the different worksheets within one workbook.

As an example, one of the worksheets can include the informal information we have laid out in Tables 2-2 and 2-3, which defines the Extract Transform and Load (ETL) process in a solution. Figure 2-2 shows that we have recorded the need to extract dates from the flat file and convert the string data into a datetime data type, on a worksheet called ETL Planning.



	A	B	C	D	E	F
1	Data Source	Source Data Type	Data Destination	Destination Data Type	Transformations	Example
2	FlatFile.Date	string	FactWeather.Date	datetime	Cast to datetime	01/23/2011
3	FlatFile.Max TemperatureF	string	FactWeather.MaxTempF	int	Cast to int	48
4	FlatFile.Min TemperatureF	string	FactWeather.MinTempF	int	Cast to int	43
5	FlatFile.Events	string	DimEvents.EventName	varchar(50)	na	Rain
6						

Figure 2-2. *Documenting the plan*

During the planning phase, researching how to accomplish the types of transformations you need during the ETL process helps us estimate what needs to be done during the ETL process. It also lets us contact the client earlier if we discover a problem. Although you do not actually create the ETL process yet, you do want to feel confident that you can accomplish the task when the time comes.

Listing 2-1 shows SQL code that takes a date as a string of 11 characters like those found in the text file and converts them into datetime data. One of the transformations listed in the Excel file in Figure 2-2 requires this change; thus, we can test how this is accomplished and whether this data will be clean enough to use for the ETL process we perform later.

Listing 2-1. Sample ETL Code

```
-- Convert the string to datetime
Declare @Date Char(11)
Set @Date='1/23/2011'
Select @Date; -- Outcome = 1/23/2011
Select Convert(datetime, @Date) -- Outcome = 2011-01-23 00:00:00.0
```

Adding Documents to Visual Studio

At this point, we have two documents that outline the BI solution: the original file and our Excel workbook. We should now think about organizing our work by grouping the documents in some manner. As we mentioned earlier, we are placing the documents into a folder that will be added to a Visual Studio solution.

If you are not familiar with Visual Studio already, you should know that it organizes projects and code files under a structure Microsoft calls a *solution*. These Visual Studio solutions consist of a folder with a set of XML files that identify which projects and files are part of the solution.

Creating Visual Studio Solutions and Projects

You can create a Visual Studio solution in a couple of ways. For example, if you create a Visual Studio project, a Visual Studio solution will automatically be created for you. If you are not ready to make a project yet, you can also create a blank solution and add projects to it later. In both cases, you can add documentation and script files to the solution folder at any time.

Each project you make in Visual Studio uses a predefined template. These templates are part of various plug-ins to Visual Studio. Once a project plug-in installs, it becomes part of Visual Studio, similar to how the Adobe's Flash plug-in becomes part of your web browser.

The Visual Studio plug-in that comes with SQL Server is either SQL Server Data Tools (SSDT) or Business Intelligence Development Studio (BIDS) depending on which version of SQL Server you install. As of SQL 2012, BIDS is a subset of SSDT, but in earlier versions it was a stand-alone plug-in. You may find the terms BIDS and SSDT used interchangeably on the Internet, but do not let it worry you too much. Think of SSDT as the newer version of BIDS instead of its replacement, and you will be fine. As you read though this book, you will notice we usually refer to both generically as Visual Studio.

With the BIDS/SSDT plug-in to Visual Studio, you can design SQL Server Integration Services (SSIS), SQL Server Analysis Services (SSAS), and SQL Server Reporting Services (SSRS) projects using templates. These install automatically into Visual Studio 2010 when you install SQL Server 2012. In fact, if you do not have Visual Studio 2010 already, the SQL Server installation will install it as well.

If it still seems confusing, consider the following:

- Visual Studio is a host for development tools.
- If we were to install Microsoft's C# development tools, for example, it would install Visual Studio and the C# development plug-in for Visual Studio.
- If we decided later to add Microsoft's Visual Basic .NET, it only needs to install the plug-in to the already installed Visual Studio.

If we decided later to add Microsoft's SQL Server Data Tools, the installation checks to see whether a compatible version of Visual Studio is already installed: if not, it will install it for you. If it already is installed, it just adds the SSDT plug-in as an additional development tool. Either way, the BIDS/SSDT plug-in becomes part of Visual Studio 2010.

■ **Note** We provide a lot of detail about how to use these project templates throughout the book, so don't be intimidated by the sudden inundation of acronyms. In this chapter, we created all the projects for you as part of the downloadable content. All you need to do is review these projects as we continue through this chapter.

Using Visual Studio

Visual Studio 2010 can be accessed either through SQL Server's menu item (Windows Start Button ➤ All Programs Microsoft SQL Server 2012 ➤ SQL Server Data Tools) or under the Visual Studio menu item (Windows Start Button ➤ All Programs ➤ Microsoft Visual Studio 2010). Both options open Visual Studio 2010 and present a selection of project templates in the New Project dialog window (Figure 2-3).

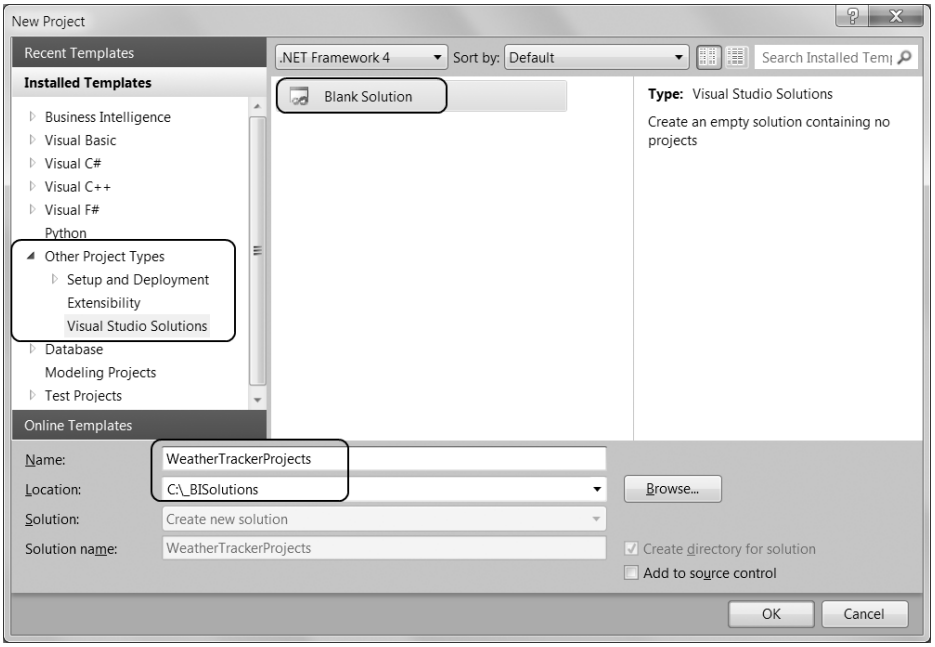


Figure 2-3. *Creating a blank Visual Studio solution*

In Figure 2-3, under Installed Templates, you can see Business Intelligence, Visual Basic, C#, and other categories listed. Beneath these template categories are the templates themselves. To select a template, click a category listed in the treeview and then choose a template in the center of the dialog window.

Each category can have many templates, so Microsoft includes subcategories to help organize the templates. For instance in Figure 2-3, you can see that there is only one template called Blank Solution, under Other Project Types ➤ Visual Studio Solutions.

Be warned, you may not see the same categories and templates on every computer! Those shown in Figure 2-3 appear because the screenshot was taken on a computer that had all of these plug-ins installed. If, however, you only have SQL Server installed, then you will not see the Visual Basic or C# plug-ins on your computer. Instead, you will see only BI projects. That is not a problem, of course, because that is exactly the type of project we want to create.

Creating a Blank Solution

In Visual Studio, new solutions that do not use a project template are referred to as *blank solutions*. Creating a blank solution is quite easy. This is done by selecting File ➤ New ➤ Project from the file menu option.

When the new project dialog window opens, a list of project types is displayed on the left side of the screen. Expanding the Other Project Types by clicking the small arrow (or triangle) allows you to select the Visual Studio Solutions option (Figure 2-3).

Because a Visual Studio solution is a collection of one or more projects, we have named the solution WeatherTrackerProjects in the Name textbox. Place the solution folder somewhere that is easy to find. In Figure 2-3, we typed `C:\_BISolutions` into the Location textbox. (This naming convention corresponds to the downloadable content and the step-by-step guide within the exercises.)

Working with the Blank Solution

After you have chosen a template and configured both the name and location, click OK to close the dialog window and begin working with the new solution. Behind the scenes, Visual Studio creates a number of files and folders, but all you see is a single folder displayed in a treeview-based window called Solution Explorer. This is the main window used to work with Visual Studio solutions.

■ **Note** Visual Studio automatically generates new subfolders for each project within the solution folder. In addition, because we specified a nonexistent folder (`C:\_BISolutions`), Visual Studio creates both the `_BISolutions` folder and the `WeatherTrackerProjects` solution folder for us. In this book, we use the `_BISolutions` folder to organize all of our solutions folders under one principal folder.

When you create a blank solution, Visual Studio shows the solution name in the Solution Explorer window but not much else. We are going to add a new solution folder specifically to hold our solution documents by clicking the Add New Solution Folder button circled in Figure 2-4. Once you click this button, a new folder is created instantly, and the text is highlighted to enable you to rename it easily.

You should rename your folder to something appropriate. This solution folder will hold a collection of documents for our solution, so a name such as SolutionDocuments is appropriate (Figure 2-4).

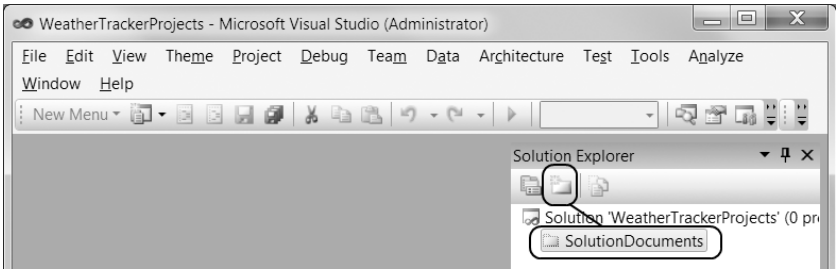


Figure 2-4. *Creating a solution folder*

Once the folder is created and renamed, you can then add documents you have created or collected to it. Simply click the new `SolutionDocuments` folder you created, which highlights the folder. Then right-click the folder and select Add ➤ Existing Item from the context menu, as shown in Figure 2-5.

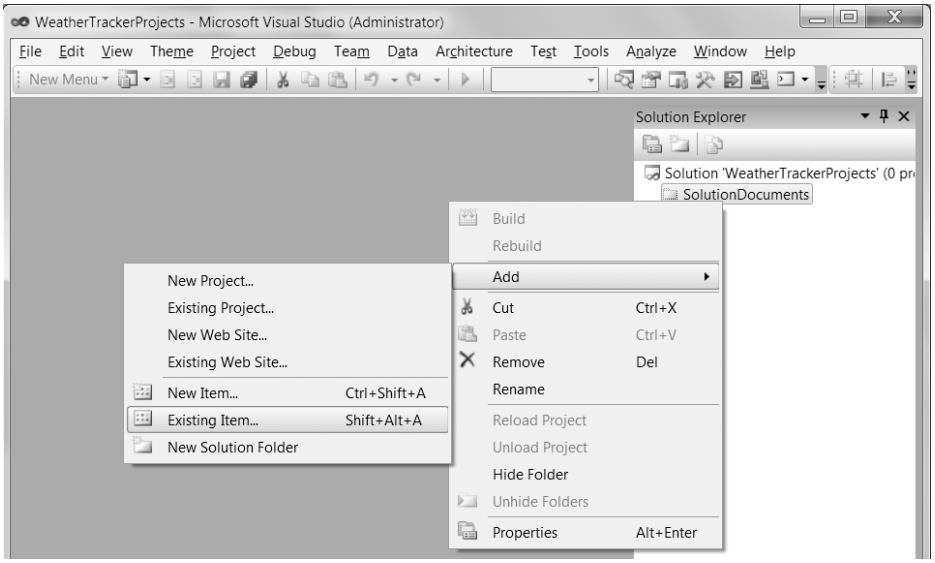


Figure 2-5. *Adding existing files to a solution folder*

Another method is to highlight the new Solution Documents you have created, and from Visual Studio's main menu, select Project➤Add Existing Item. Both allow you to navigate to where the files are located and add them to your solution.

After you have selected your files and added them to your blank solution, Visual Studio will either copy the file to your solution folder or reference the file from its existing location.

■ **Important** About 90% of the time Visual Studio will copy the file instead of making a reference to it. It is always important to verify whether a reference or copy was made. Using references can cause major problems because any changes to the files in your Visual Studio solution will change what you believed to be a copy. You can tell where a file is located by right-clicking the file and selecting Properties from the context menu (similar to Figure 2-5), and the file's path will be displayed in a Property window. In cases where Visual Studio creates a reference, when what you really wanted was a copy, you need to use Windows Explorer to copy the file to the solution folder on your hard drive yourself and then make a reference to the newly copied file.

One of the primary goals of this book is to give you a chance to practice the art of creating BI solutions. To keep things simple, we have created all the WeatherTracker BI solution documents and BI projects for you in this example. This provides you with a quick introduction to the anatomy of a BI solution and introduces you to organizing your projects using Visual Studio, without having to explain how to create these projects and files. Don't worry! We explain how those items are created in the other chapters of this book.

EXERCISE 2-1. PREPARING THE SOLUTION FILES

In this exercise, you add the downloadable book files to your C: drive. You then create a blank Visual Studio solution to hold BI solution documents, and connect to the files you downloaded within the new solution. This step is the foundation of all your future exercises and must be completed for future exercises to work properly.

Later in this chapter, you will add SSIS, SSAS, and SSRS projects to this Visual Studio solution. Figure numbers provide hints for most steps.

Install the Book Files

The files for this exercise, as well as all of the exercises throughout this book, are available in the downloadable book content and need to be installed on your computer before you can continue.

1. If you have not done so, download the book files from the Apress website. These files are in a zipped format.
2. Create a folder on your `C:\` drive called `_BookFiles` and unzip the downloadable files into it.

Each operating system unzips files in a different manner; therefore, we are not including step-by-step instructions on how to unzip these files. Just make sure that the book files are on the root of your `C:\` drive using the name `_BookFiles`. We strongly recommend you use this name, because we continually reference it in the book.

Once unzipped, this folder includes all the files and projects you need to complete each exercise within this book.

Create a Folder for all BI Solutions

We want to have one place where all of the BI solutions you create in this book are stored, so let's create one now.

1. Inside the `_BookFiles` folder, locate the `Chapter02Files` folder and open it.
2. Find a subfolder called `_BISolutions` and use Windows Explorer to copy this folder to the root of `C:\` drive, as shown in Figure 2-6. (To open Windows Explorer, access your computer's Start menu, and click Computer. In the left column select Local Disk (C:). On the right, where all of your `C:\` files are listed, paste the entire `_BISolutions` folder here.)

You now have a second folder on your hard drive called `C:\_BISolutions`.

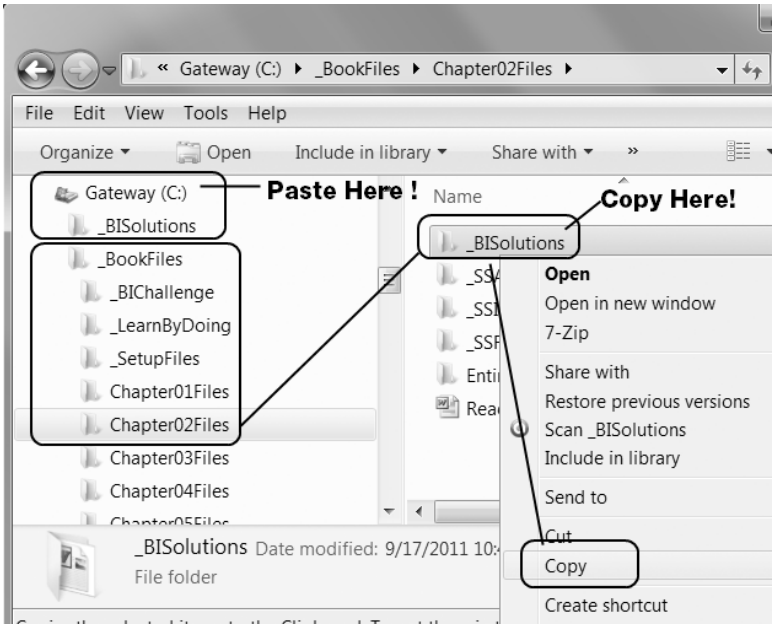


Figure 2-6. *Now both* `_BISolutions` *and* `_BookFiles` *are on the* `C:\` *drive*

Review the WeatherTrackerProjects Files

After you have copied the `_BISolutions` folder to its new location, you may want to look inside it to see exactly what you just copied. Inside this folder is a subfolder called `WeatherTrackerProjects`. Within that folder are four documents we use in this chapter's BI solution:

- `InstWeatherTrackerDW.sql` (SQL code to create a data warehouse)
- `SQLTransformations.sql` (SQL code for the ETL process)
- `WeatherHistory.txt` (a text file with the client's data)
- `WeatherTrackerETLPlan.xls` (an Excel file that outlines the solution plan)

1. Verify that you have both the `_BookFiles` (the original folder that holds all of our chapter files and demos) as well as the `_BISolutions` folder (the folder you will place your work in) directly on your hard drive, as shown in Figure 2-6.

Placing these files directly on your `C:\` drive makes it much easier to navigate to the files you need for these exercises, and leaves less room for confusion later, because we access this folder quite often throughout this book.

Open Visual Studio

You now need to open Visual Studio and create a new solution. The following steps walk you through the process. Visual Studio opens from either the Microsoft Visual Studio 2010 or the SQL Server Data Tools menus. We have chosen to use the Visual Studio option for simplicity, but either menu item works.

1. Open Visual Studio 2010. You can do so by clicking on the Start button and navigating to All Programs ➤ Microsoft Visual Studio 2010 ➤ r. Right-click Microsoft Visual Studio 2010 to see an additional context menu (Figure 2-7). Then, click on the Run as Administrator menu item.

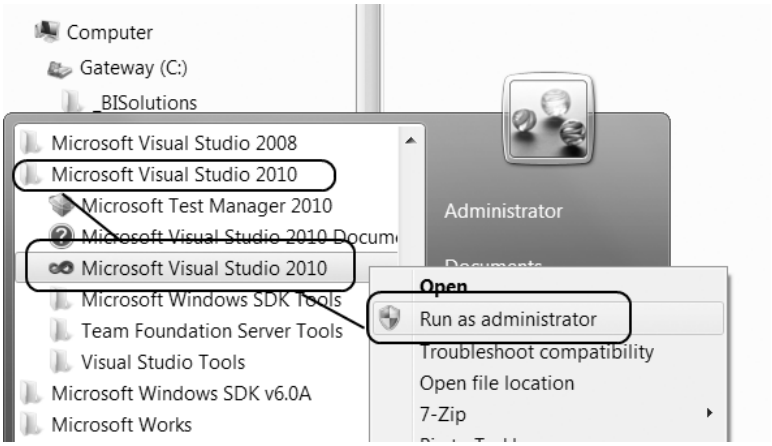


Figure 2-7. *Opening Visual Studio and running as admininstrator*

2. If the User Account Control (UAC) message box appears asking "Do you want the following program to make changes to this computer?" click Yes (or Continue depending upon your operating system) to accept this request.

3. When Visual Studio opens, select File ➤ New ➤ Project from the menu. (Do not use the Create: Project option from the Start Page as you may have done in the past with other types of solutions.)

4. When the New Project dialog window opens, on the left side of your screen click the arrow to expand Other Project Types and select Visual Studio Solutions, as shown in Figure 2-3.

5. In the templates section, select Blank Solution, as shown in Figure 2-3. The Name and Location textboxes are filled in with a default name, but we change these in the next step.

6. In the Name textbox, at the bottom of the screen, type the name **WeatherTrackerProjects**. In the Location textbox, type **C:_BISolutions** (as shown previously in this chapter in Figure 2-3), and finally, click OK.

Once the solution is created, it appears in the Solution Explorer window of Visual Studio, on the right side of your screen.

Review the Files Created by Visual Studio

1. Right-click the solution WeatherTrackerProjects icon, and select the Open Folder in Windows Explorer menu item (Figure 2-8). Windows Explorer will open to the location of your solution folder.

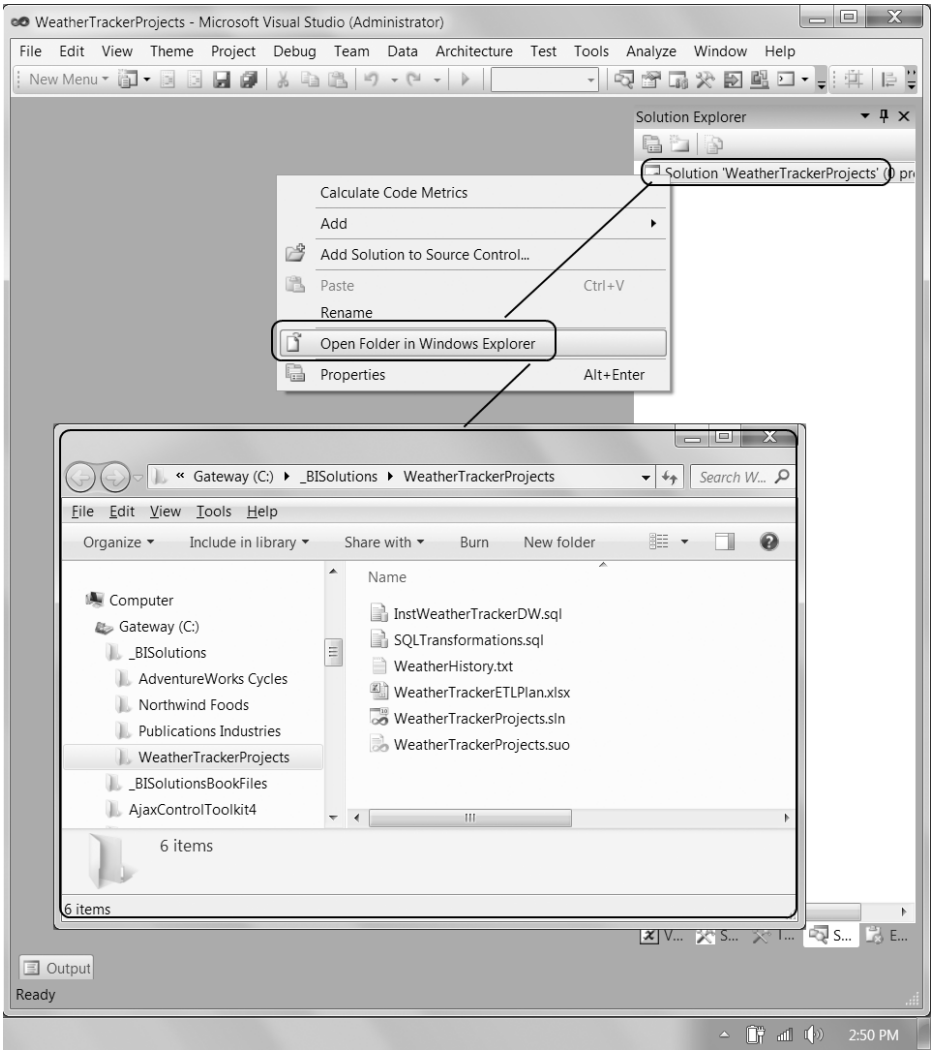


Figure 2-8. *Viewing the files within the solution folder*

2. Review the files in this folder and notice that they are not currently showing in the Solution Explorer window. (Solution Explorer shows only a minimum of files from a solution or project folder, but we can change this by adding the file to the solution as existing items, as we do in just a moment.)

3. Now that you have seen what is in the folder, close Windows Explorer.

Add a New Solution Folder

You can organize your files into logical groups using a solution folder. To do so, you must first create a folder and then add files to it. Let's do that now.

1. Add a new solution folder to your Visual Studio Solution by clicking the Add New Solution Folder button at the top of the Solution Explorer window. (This step is illustrated in Figure 2-4 of this chapter.)
2. Rename the new folder as SolutionDocuments (shown previously in Figure 2-4).
3. Right-click the new SolutionDocuments folder and select Add ➤ Existing Items from the context menu (Figure 2-5). A dialog window will open.
4. In the left column, select your Local Disk (C:). On the right, first double-click `_BISolutions`, and then double-click the WeatherTrackerProjects subfolder to access the files within it.
5. While holding down the control button, click and select the following files: `InstWeatherTrackerDW.sql`, `SQLTransformations.sql`, `WeatherHistory.txt`, and `WeatherTrackerETLPlan.xls` (Figure 2-9).

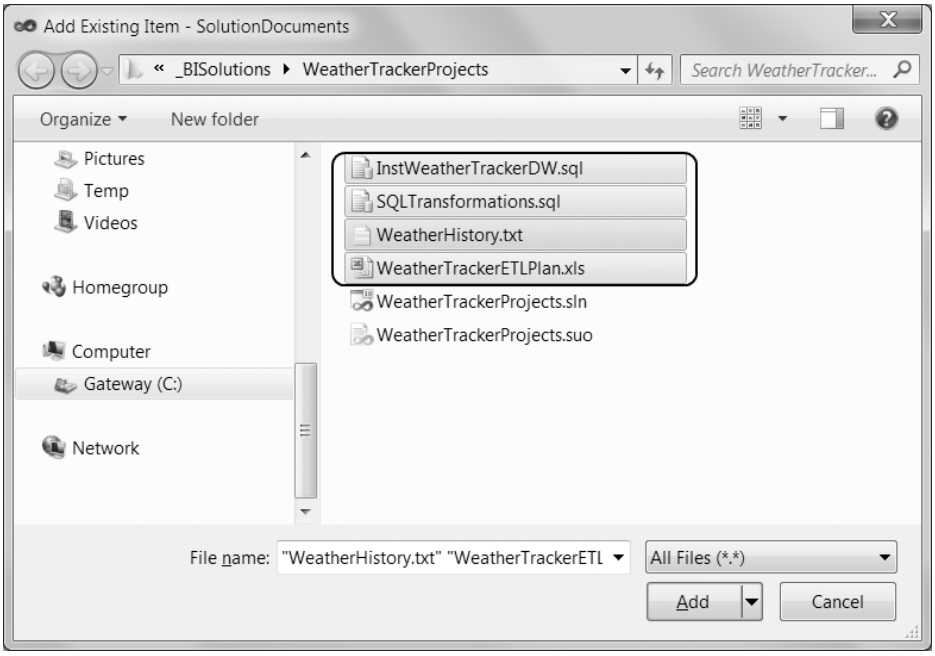


Figure 2-9. *Selecting your BI solution files*

Note: Windows by default hides the file extensions. So, you may see only the first part of the name of each of these files. We recommend turning off this feature when you get the chance. It will often be very helpful to know the extensions of all your files. To find more of this feature, search the Web for "Windows Show Extensions."

6. Click the Add button to add the highlighted items to your `SolutionDocuments` folder. (Visual Studio uncharacteristically creates a reference instead of a copy when you add an existing item to a solution folder.)

7. Visual Studio will open each of the files in Visual Studio as well as Excel so that you can see their content. We are not making any changes to these files. Look at them if you like, but then close them by clicking Excel's closing X and the X on each Visual Studio tab, as shown in Figure 2-10. Note that Microsoft hides the closing X on a tab until it has been selected.

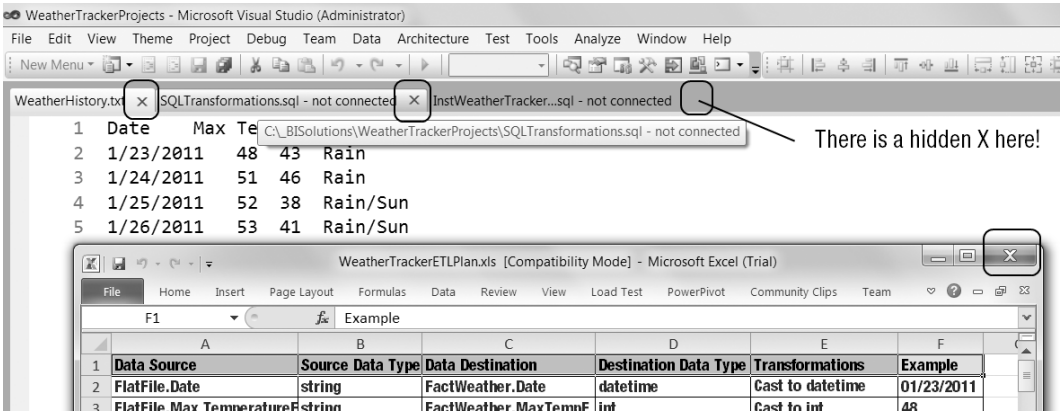


Figure 2-10. *Closing your BI solution files*

8. Use Visual Studio's File menu to save your work by selecting the Save All option.

9. Leave Visual Studio open for now because we continue to work with it in the next exercise.

In this exercise, you created a blank solution and added documents that will be used for creating your SSIS, SSAS, and SSRS projects. We refer to these documents in future exercises.

Creating the Data Warehouse

Once you have assembled the documents that outline your solution plan and after you have added those documents to a Visual Studio solution, it is time to create the BI solution projects starting with the data warehouse. Let's begin this process with an overview of what a data warehouse is and how it is created. Then we provide you with code that creates the data warehouse, and finally, we add that code to a new Visual Studio solution folder called `DWWeatherTracker`.

An Example Data Warehouse

In this book, we describe a data warehouse as a collection of one or more data marts. These data marts consist of one or more fact tables and their supporting dimension tables. In Figure 2-11 you see a design with a single fact table called FactWeather and a one-dimensional table called DimEvents. Notice the correlation between the notation in the Excel spreadsheet of Figure 2-2 and the design of these tables in Figure 2-11.

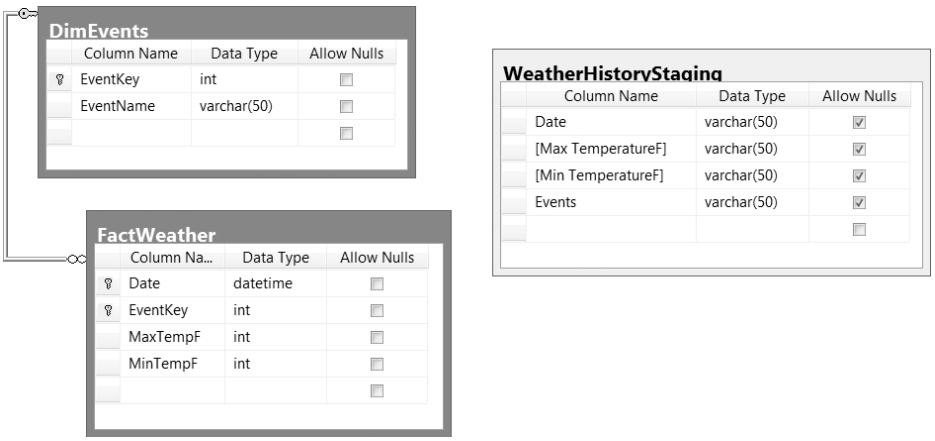


Figure 2-11. *The data warehouse tables*

These tables represent a very minimal design. As shown in Chapter 4, there are typically several dimension tables in a data warehouse, not just one. For now, though, let's keep focusing on the big picture and come back to the details later.

Using SQL Code to Create a Data Warehouse

One of the solution documents, `InstWeatherTrackerDW.sql`, has SQL code that creates the DWWeatherTracker data warehouse for you when it is executed in SQL Server Management Studio. Before we have you execute this code, let's review what it does.

■ **Note** The code file `InstWeatherTrackerDW.sql` can be found as one of the documents you added to your Visual Studio solution in Exercise 2-1. It opens within Visual Studio if you double-click the file. In the next exercise, we open and run the code in SQL Server Management Studio, so you will become used to working with both tools.

Create the Database

The first set of tasks that the SQL code tackles is checking to see whether the database already exists and, if so, drop it. We labeled the first tasks Step 1 in our code (Listing 2-2). After that, in Step 2, the code creates the database and tells SQL Server to use the new database for all the commands that come next.

Listing 2-2. Drop and Create the Database

```
--Step 1) Drop the database as needed
Use Master
Go
If ( exists( Select Name from SysDatabases Where name = 'DWWeatherTracker' ) )
 Begin
   Alter Database [DWWeatherTracker] Set single_user With rollback immediate

   Drop Database [DWWeatherTracker]

 End
```

```
Go
-- Step 2) Create Data Warehouse Database
Create Database DWWeatherTracker
Go
Use DWWeatherTracker
Go
```

Create the Tables

The next three steps outlined in the `InstWeatherTrackerDW.sql` code file creates three tables (Listing 2-3).The first table is to hold raw data imported from the text file `WeatherHistory.txt`. The second table, DimEvents, is our one and only dimension table in this example. The third table, FactWeather, is our fact table.

Listing 2-3. Creating Three Tables

```
-- Step 3) Create a Staging table to hold imported ETL data
CREATE TABLE [WeatherHistoryStaging]
( [Date] varchar(50)
, [Max TemperatureF] varchar(50)
, [Min TemperatureF] varchar(50)
, [Events] varchar(50)
)
-- Step 4) Create Dimension Tables
Create Table [DimEvents]
( [EventKey] int not null Identity
, [EventName] varchar(50) not null
)
Go
-- Step 5) Create Fact Tables
Create Table [FactWeather]
( [Date] datetime not null
, [EventKey] int not null
, [MaxTempF] int not null
, [MinTempF] int not null
)
```

In step 4, the DimEvents dimension table is created (Figure 2-11). In this table, we have both a key column and a name column. This is characteristically the minimum design seen in real-life examples. In most cases, however, there are also additional descriptive columns in the table.

Using the Identity Option

In Listing 2-3, we included an identity attribute on the EventKey column. In SQL Server, a column marked with an identity attribute automatically adds incremental integer values to the column each time a row of data is inserted into the table. In other words, because we have configured the EventKey column to be an identity column, adding a new event name to the DimEvents table will automatically insert an integer of "1" into the EventKey column. When we add another event name, an integer of "2" is inserted for the second row, and so on.

Adding Primary Key Constraints

You should include primary key constraints in all of your dimension and fact tables because they keep your data ordered and free of duplicate values. In most dimension tables, you add a primary key constraint to its single key column. But in fact tables, you add a primary key constraint to multiple key columns, because it is the combination of key values that distinguishes one row from another. When a primary key constraint is associated with multiple columns, these columns form a *composite primary key*.

As an example, there are two key columns in the FactWeather table, the Date and EventKey, both of which refer to dimensional tables. The other two columns in the table are MaxTempF and MinTempF, both of which are measure columns. The multiple dimensional key columns form a composite primary key for a fact table.

The code in Listing 2-4 creates a primary key constraint on the DimEvents and FactWeather tables. Adding the constraint to the table identifies which column or columns are part of the primary key and enforces uniqueness of values across these columns.

Listing 2-4. Adding the Primary Keys

```
-- Step 6) Create Primary Keys on all tables
Alter Table DimEvents Add Constraint
  PK_DimEvents Primary Key ( [EventKey] )
Go
Alter Table FactWeather Add Constraint
  PK_FactWeathers Primary Key ( [Date], [EventKey] )
Go
```

Looking back at Figure 2-11, you can see the primary key icons are on both the Date and EventKey columns, which indicates that both columns are part of a composite primary key. Look for these icons, or something similar, in any database diagram you review.

Adding Foreign Key Constraints

Notice in Figure 2-11 that both the fact table and the dimension table have a column called EventKey. In the fact table, the EventKey column forms a foreign key relationship back to the DimEvents dimensional table. The code in Listing 2-5 adds a foreign key constraint to enforce this relationship and will not allow you to enter key values in the fact table if they do not first exist in the dimension table. For instance, if you try to insert an EventKey value of 42 to the fact table, the constraint would check to see whether an EventKey value of 42 exists in the dimension table. If not, the database engine generates an error message and the insert fails!

Listing 2-5. Adding the Foreign Keys

```
-- Step 7) Create Foreign Keys on all tables
Alter Table FactWeather Add Constraint
  FK_FactWeather_DimEvents Foreign Key( [EventKey] )
  References dbo.DimEvents ( [EventKey] )
Go
```

■ **Note** Many exercises in this book are written in a way that assumes you have some familiarity with SQL programming. We have tried to make our code simple enough for all levels of developers, but some of this subject matter may be difficult if you have never used SQL before. To help you become more familiar with this language, we recommend checking out the excellent, and free, SQL tutorial on the website `www.w3schools.com`.

Running SQL Code from Visual Studio

You can manage and execute your database scripts using Visual Studio even if it is not obvious how to do so. In the next exercise, you have an opportunity to do just that. We provided step-by-step instructions on how to do so.

EXERCISE 2-2. CREATING THE DATA WAREHOUSE

In this exercise, you create the data warehouse and the tables within it. You can do this by using the code found in the `InstWeatherTrackerDW.sql` file. Once that is accomplished, you create a new solution folder in Visual Studio and move the `InstWeatherTrackerDW.sql` file to the new folder. Figure numbers provide hints for most of these steps.

Completion of this exercise is required to be able to complete future exercises throughout this chapter.

Open Visual Studio (Optional)

1. Visual Studio should still be open from the previous exercise. If it is not, please open it. Please remember to run Visual Studio as an administrator by right-clicking the menu item and selecting the Run as Administrator option.
2. With Visual Studio open, access the WeatherTrackingProject solution from the File ➤ Recent Projects and Solutions menu.

Connect to SQL Server and Execute the Code

1. Double-click the file `InstWeatherTrackerDW.sql` file in Solution Explorer. The SQL code you see in Figure 2-12 opens in your main window.

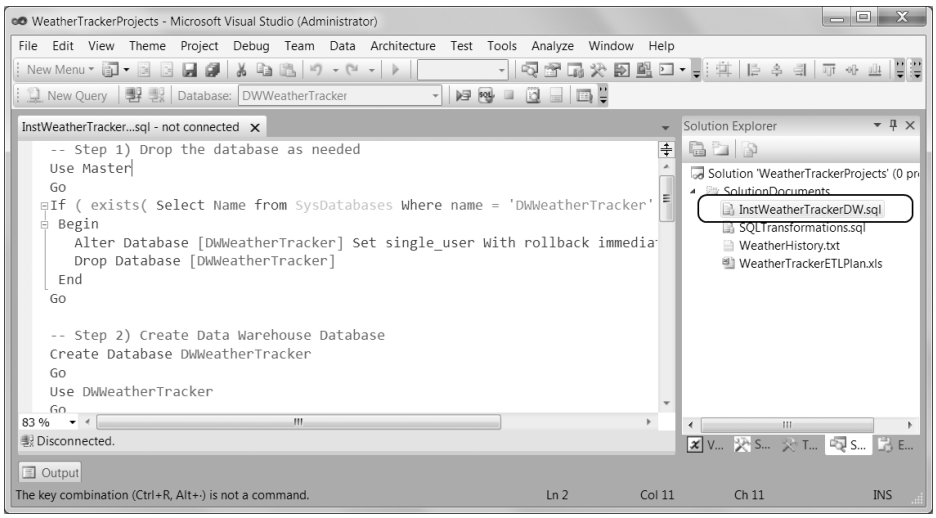


Figure 2-12. *Double-click* `InstWeatherTrackerDW.sql` *to open it.*

2. Without selecting any of the SQL code, right-click a blank area of the query window (the window where the SQL code is) and then click the Connection ➤ Connect menu option (Figure 2-13). The Connect to Database Engine dialog window appears.

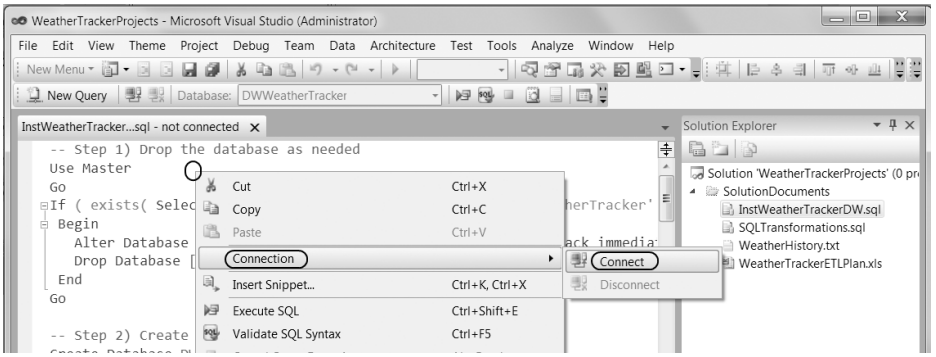


Figure 2-13. *Connecting to SQL Server from Visual Studio*

3. In the Connect to Database Engine dialog window, type in the name of your computer or use the alias of (local) in the Server Name textbox (Figure 2-14); then click the Connect button to make the connection. If you have trouble with this step, see the upcoming "Important" note.

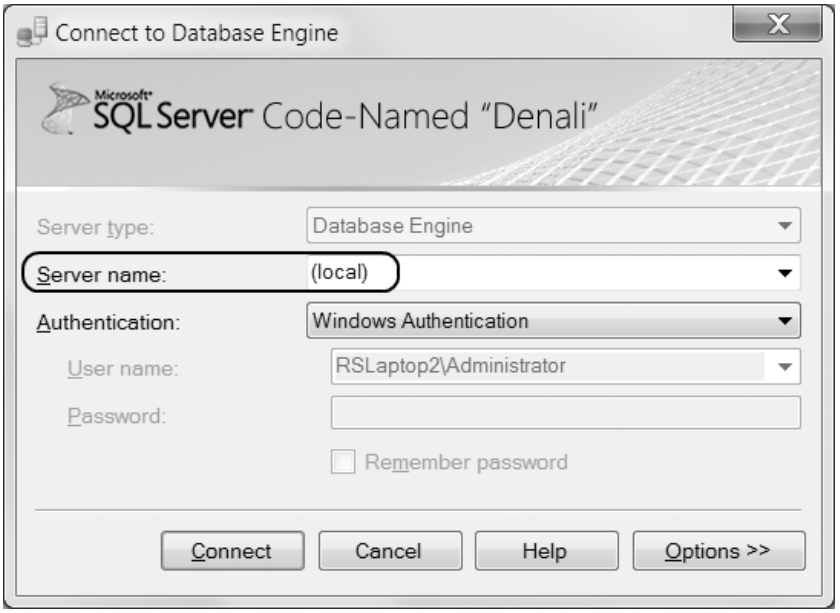


Figure 2-14. *Entering the server name*

Important: If you have installed SQL Server on the same computer multiple times or if you named the instance of your single install, your SQL 2012 installation may be called a name other than (local). For example, Randal has SQL installed as (local)\SQL2012, and Caryn references her server as (local)\Denali. (When you install SQL, you get to make up the name!) If a named instance is used, you must connect to SQL Server using the instance name in the "Server name" textbox. For more information, search the Web for "SQL Server Named Instances."

4. After connecting to the database, execute the SQL code by right-clicking an area in the SQL code window to bring up the context menu, and select Execute SQL (Figure 2-15).

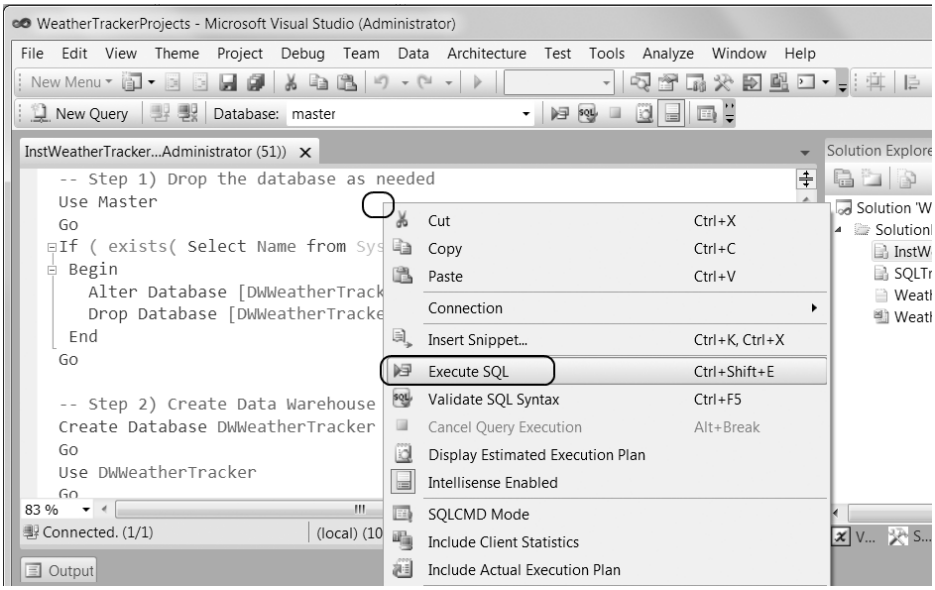


Figure 2-15. *Executing your code*

5. The code in the SQL file should complete successfully in only a few seconds. You will know that it has worked when the message displayed in Figure 2-16 appears. This is a good sign, but you should also verify that the database was created by connecting to it. We will do that next.

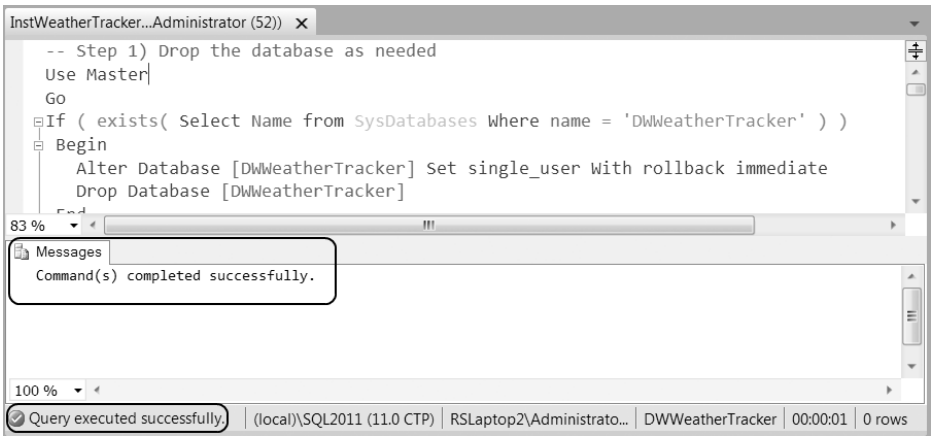


Figure 2-16. *The SQL code executed successfully*

Verify That The Database Was Made

1. Open the Server Explorer window of Visual Studio. You can do so by using the View➤Server Explorer menu item (Figure 2-17). Be careful, because it is easy to click the Solution Explorer item by mistake. Server Explorer should display on the left side of Visual Studio.

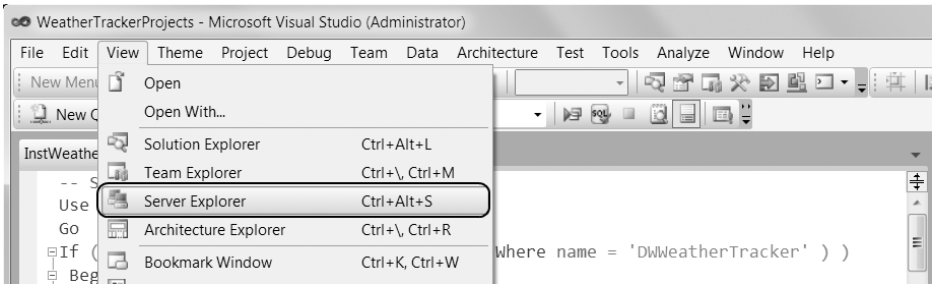


Figure 2-17. *Displaying Server Explorer*

2. In Server Explorer, right-click the Data Connections icon, and select Add Connection from the context menu (Figure 2-18). The Add Connection dialog window appears (Figure 2-19).

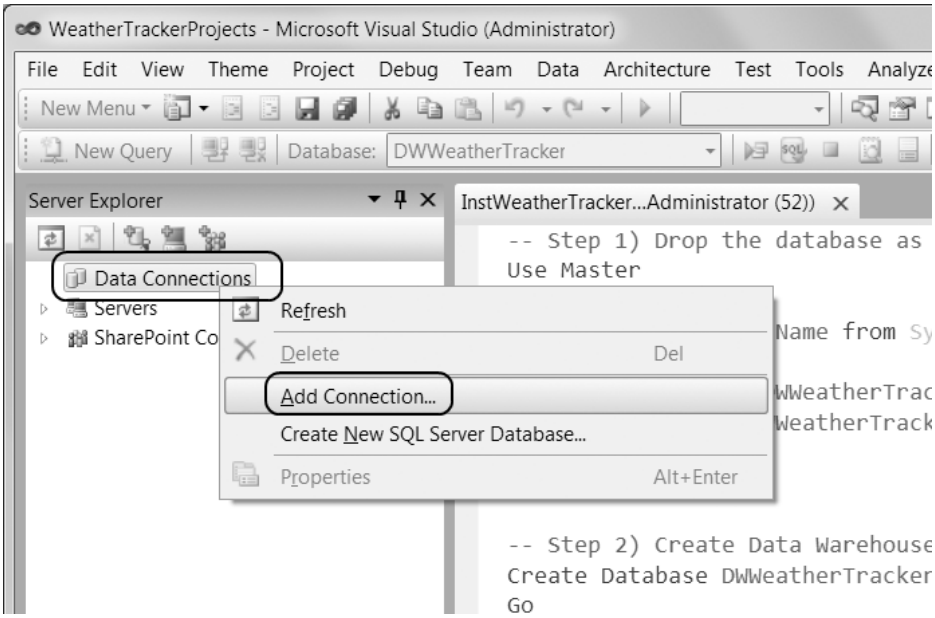


Figure 2-18. *Connecting to the SQL database engine from Server Explorer*

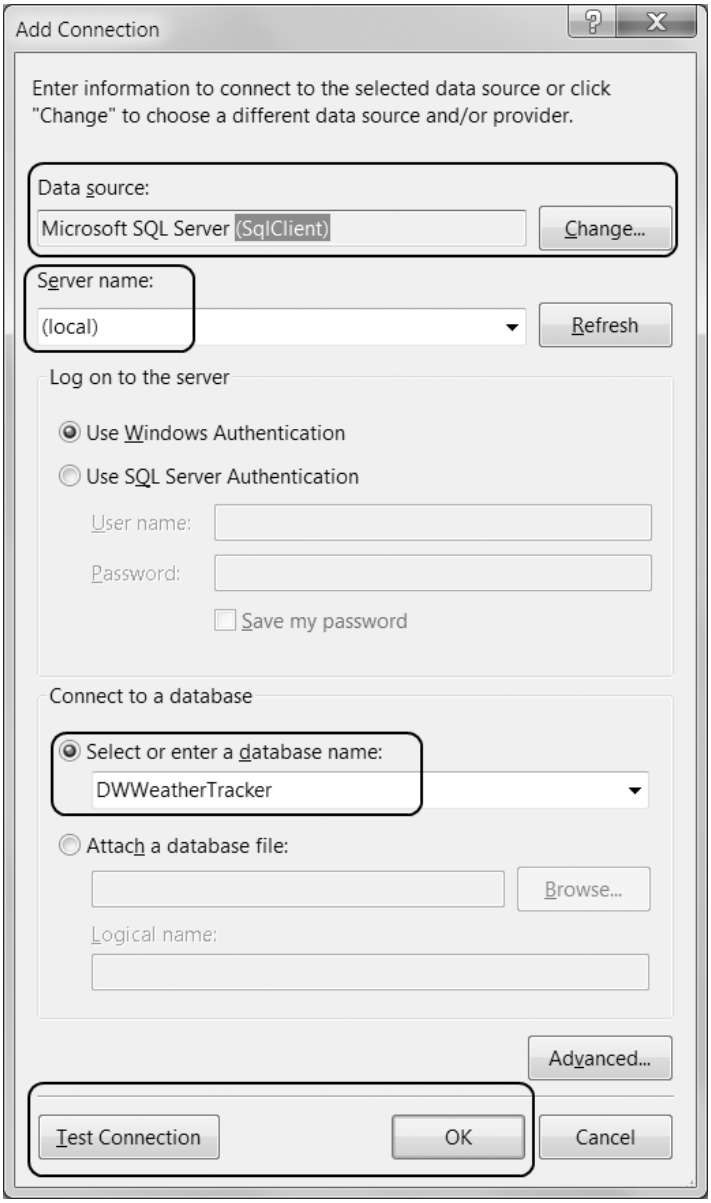


Figure 2-19. *Configuring the connection*

3. In the Choose Data Source dialog window (Figure 2-20), set the data source to Microsoft SQL Server (SQLClient). If you need to change this setting, click the Change button.

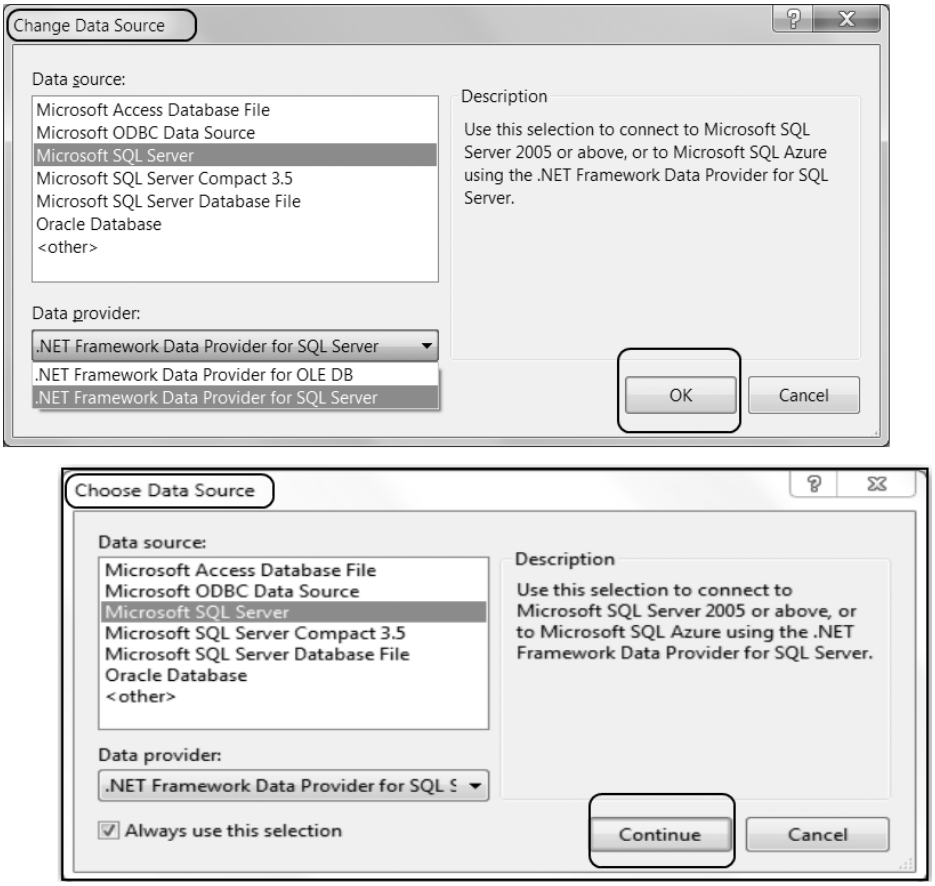


Figure 2-20. *Setting the connection to use SQL Server as the data source*

Important: Depending on a combination of things, Visual Studio will sometimes display the Choose Data Source window, shown at the bottom of Figure 2-20, instead of the Add Connection window. Both windows look almost identical, and we cannot be sure which will open on your computer. You can use either one to select your data provider.

If the Choose Data Source window appears before the Add Connection window on your computer, just select the Microsoft SQL Server data source and the .NET Framework Data Provider for the SQL Server data provider; then click the Continue button, and the Add Connection window appears. Add the Microsoft SQL Server (SQLClient) setting to the Data Source dropdown box, as shown in Figure 2-19.

If the Add Connection window appears but the Microsoft SQL Server (SQL Client) setting is not in the Data Source dropdown box, then click the Change button, and it will open the Change Data Source window, also shown in Figure 2-20.

4. In the Add Connection dialog window (Figure 2-19), set the server name to either the name of your computer or the alias of (local) in the Server name dropdown box. If you are using a SQL named instance, then you have to include that name as well (for more information, search the Web for “SQL Server named instances”).

5. In the "Select or enter database name" dropdown box, select the DWWeatherTracker database (Figure 2-19).
6. Now, test your connection using the Test button, and when it succeeds, close it by clicking OK. Then click OK to close the Add Connection dialog window.
7. Click the small arrow next to the DWWeatherTracker database icon to expand Server Explorer's list of database objects (Figure 2-21).

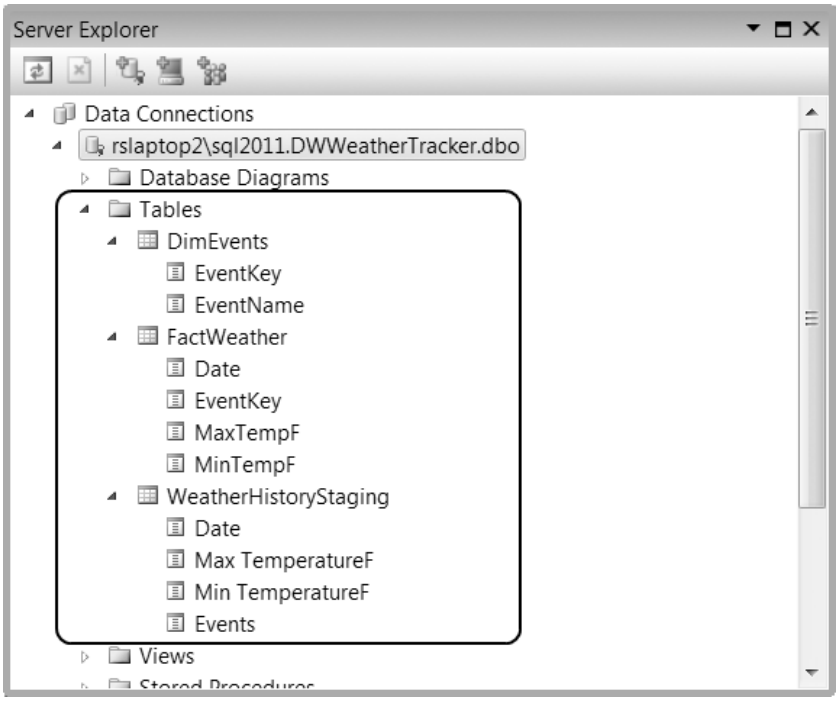


Figure 2-21. *Viewing the tables and columns in DWWeatherTracker*

8. Click the small arrow next to the Tables folder to expand Server Explorer's list of tables.
9. Click the small arrow next to the DimEvents table to expand the column listing.
10. Double-click the file `WeatherTrackerETLPlan.xls` in Solution Explorer. This opens an Excel window.
11. Verify that the tables are properly made by comparing them to the Excel spreadsheet, as shown in Figure 2-22.

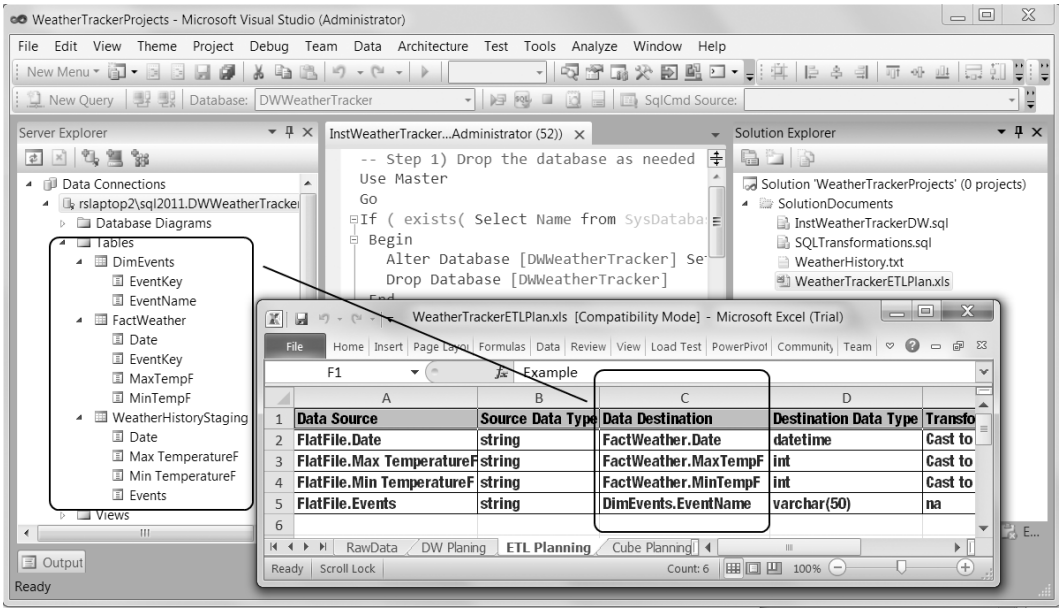


Figure 2-22. *Verifying the tables and columns in DWWeatherTracker*

12. All should be correct, so you can close the Excel spreadsheet (`WeatherTrackerETLPlan.xls`), the SQL file (`InstWeatherTrackerDW.sql`), and Server Explorer.

Place the Code File in a New Solution Folder

Now that the data warehouse is built and you have verified that it is correct, let's move the SQL code file to a new Visual Studio solution folder.

1. In Solution Explorer, select the solution WeatherTrackerProjects icon by clicking it and then click the Add New Solution Folder button to create a new solution folder. (Be sure you do not have the Solution Documents folder highlighted, or the new folder will be created as a subfolder of it instead of a subfolder of the entire solution.) Select the title and then click it again to rename it WeatherTrackerDataWarehouse.

2. Click the `InstWeatherTrackerDW.sql` file in the `Solution Documents` folder and drag it into your new `WeatherTrackerDataWarehouse` folder. Note that this does not move the file to a new location on the hard drive; it just adds a new visual reference to the file in Solution Explorer.

3. Now you have two references to this file, one in each solution folder. We now remove the original. Right-click the `WeatherTrackerDW.sql` file that is within `SolutionDocuments` and select Remove from the context menu. This does not delete the file but only removes that reference. Verify that what you see in Solution Explorer looks similar to Figure 2-23.

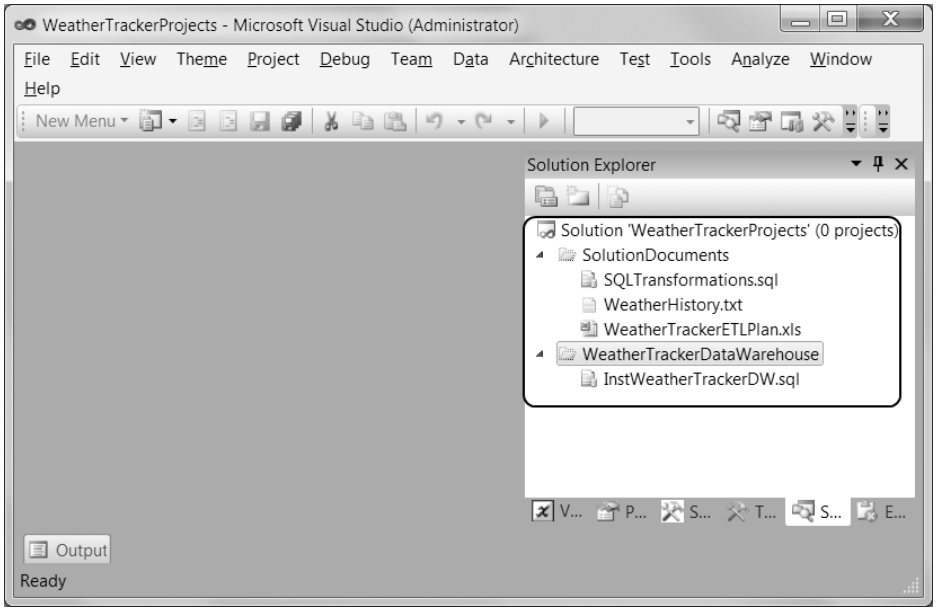


Figure 2-23. *Solution Explorer at the end of Exercise 2-2*

4. Leave Visual Studio open for now because we continue working with it in the next exercise.

In this exercise, you executed a SQL script that created a data warehouse. You then placed this script into a new solution folder using Visual Studio. Soon, you will be adding additional projects to the solution. Your ultimate goal is to place all the code and projects you need for the WeatherTracker BI solution into this one Visual Studio solution. We continue this process in the next exercise by adding a SQL Server Integration Services project.

Create the ETL Process

With the data warehouse created, it is time to start the extract, transform, and load (ETL) process. During this phase of a BI solution, just as the title of the process states, you must first extract the source data, then transform it as necessary, and finally load it into the data warehouse tables. In the WeatherTracker project, the text file called `WeatherHistory.txt` is the source of the data. The destinations are the tables you created in Exercise 2-2.

The transformations needed for the WeatherTracker ETL process are listed in the Excel spreadsheet we created earlier (Figure 2-2). To create the ETL process using SSIS, we examine this spreadsheet making special note of column names, data types, and transformations listed. Let's review an SSIS project based on our recorded plan.

ETL with an SSIS Project

SSIS represents Microsoft's premier ETL tool. It is one of Microsoft's business intelligence servers, and it is one of the project types available in Visual Studio. To create an SSIS project, start Visual Studio and select File ➤ Add ➤ New Project from its main menu, as shown in Figure 2-24. Doing so forces the Add New Project dialog window to appear (Figure 2-25).

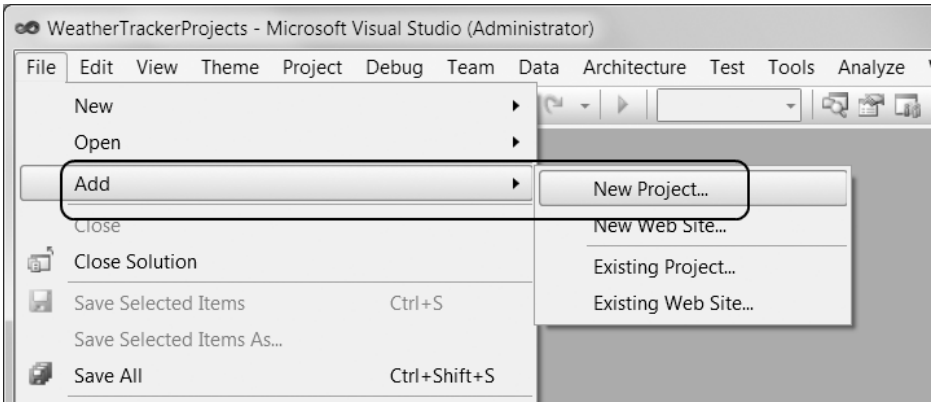


Figure 2-24. *Adding another project to the current Visual Studio solution*

In the Add New Project dialog window, you have selected the type of project you want it to make, so expand the Business Intelligence category and select the Integration Services subcategory, as shown in Figure 2-25. Then select the Integration Services Project template, available in the center of the dialog window.

Before you click the OK button to close the dialog window, you should set the project's name to something appropriate and verify the project's location. In Figure 2-25, we have set the Name textbox to WeatherTrackerETL and left the Location textbox set at (`C:\_BISolutions\WeatherTrackerProjects`). After configuring these settings and clicking the OK button, a new subfolder is created under the `WeatherTrackersProjects` solution folder containing the SSIS project.

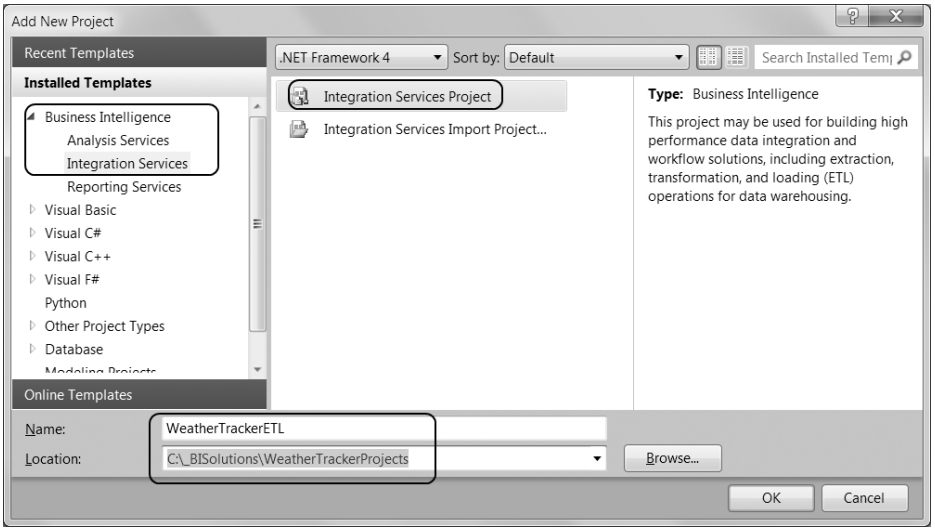


Figure 2-25. *Selecting the Integration Services Project template and naming the new project*

■ **Note** Although it is possible to place the project in a location other than a subfolder of the solution, doing so would make it more difficult to locate all of the projects for a given solution. Therefore, we recommend you always keep your projects for a particular BI solution together under one Visual Studio solution folder.

Creating an SSIS Package

An SSIS project consists of one or more package files. When you first create an SSIS project, its template includes an empty SSIS package called `Package.dtsx`. The file displays in the Solution Explorer window (Figure 2-26). Unless you are creating a throwaway demo package, you should rename the file to something indicating its purpose. You can do so by right-clicking the file and selecting the Rename menu item from the context menu.

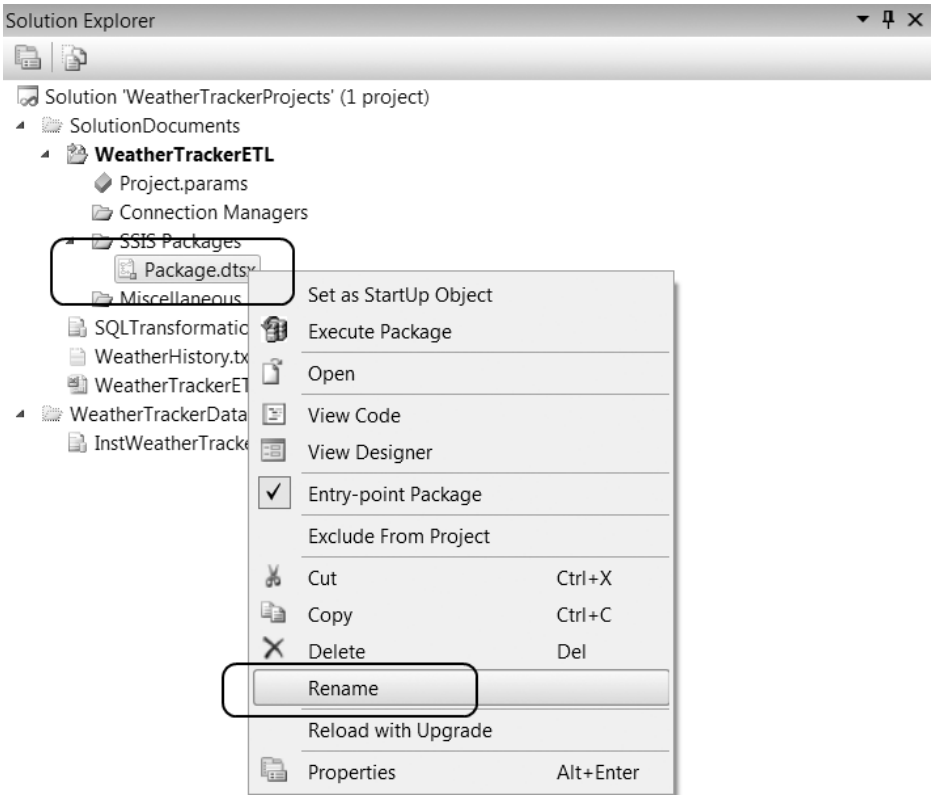


Figure 2-26. *Renaming an SSIS package*

THE GETTING STARTED (SSIS) WINDOW

In SQL 2012, SSIS has a new start-up window that provides helpful links to videos and articles (Figure 2-27). At some point, you may want to view these videos, but most of the time you can simply close this window.

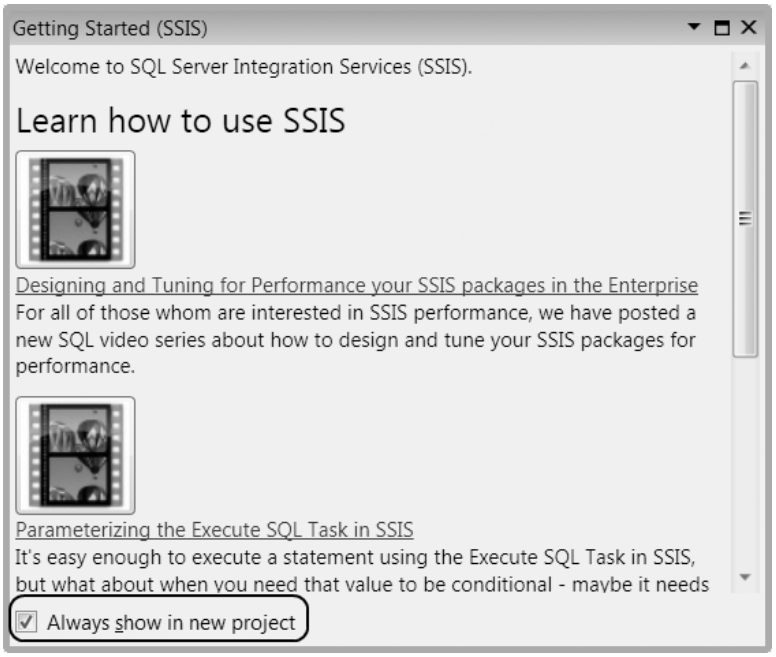


Figure 2-27. *The new Getting Started (SSIS) window*

You can close this window each time you make a new SSIS project, or you can click the "Always show in new project" checkbox to change this behavior. If you want to see this window at another time, there is a Getting Started menu item under the SSIS main menu that displays it again.

An SSIS package file is essentially a text file formatted as in the XML language. While this file can be manually programmed, you likely will let Visual Studio do the coding for you. To use this feature, you drag and drop items from the SSIS Toolbox onto a package's design surface. This act invisibly writes your XML code for you. (We will elaborate on this in a moment.)

As of SQL 2012, SSIS now includes a dedicated Visual Studio Toolbox in addition to the standard Visual Studio Toolbox (Figure 2-28).

Each item in the Toolbox represents a set of SSIS commands. For example, Figure 2-28 shows a Data Flow task icon and an Execute SQL Task icon within the Toolbox. These tasks represent a collection of individual SSIS programming commands used during ETL processing.

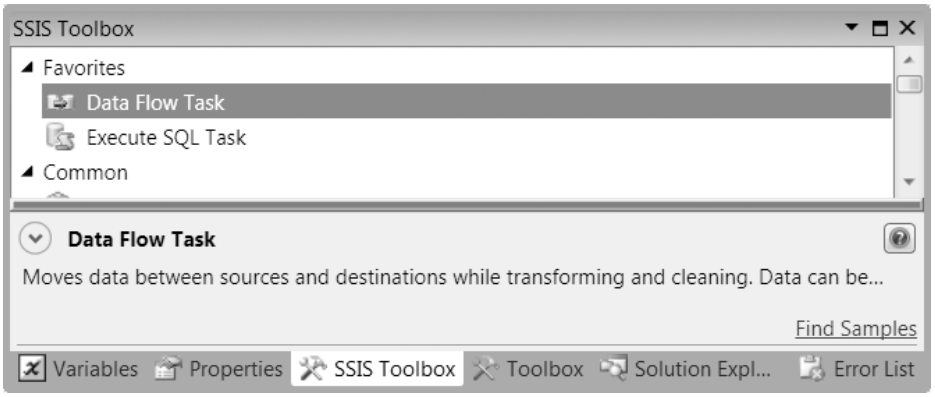


Figure 2-28. *The new SSIS Toolbox for SQL 2012*

Outlining the Control Flow Tasks

To configure a new SSIS package, we recommend outlining what you intend to accomplish by adding tasks from the Toolbox onto the package's designer interface. We show an example of what this looks like in Figure 2-29.

The designer interface is separated by tabs. SSIS tasks are created and configured on the Control Flow tab (Figure 2-29).

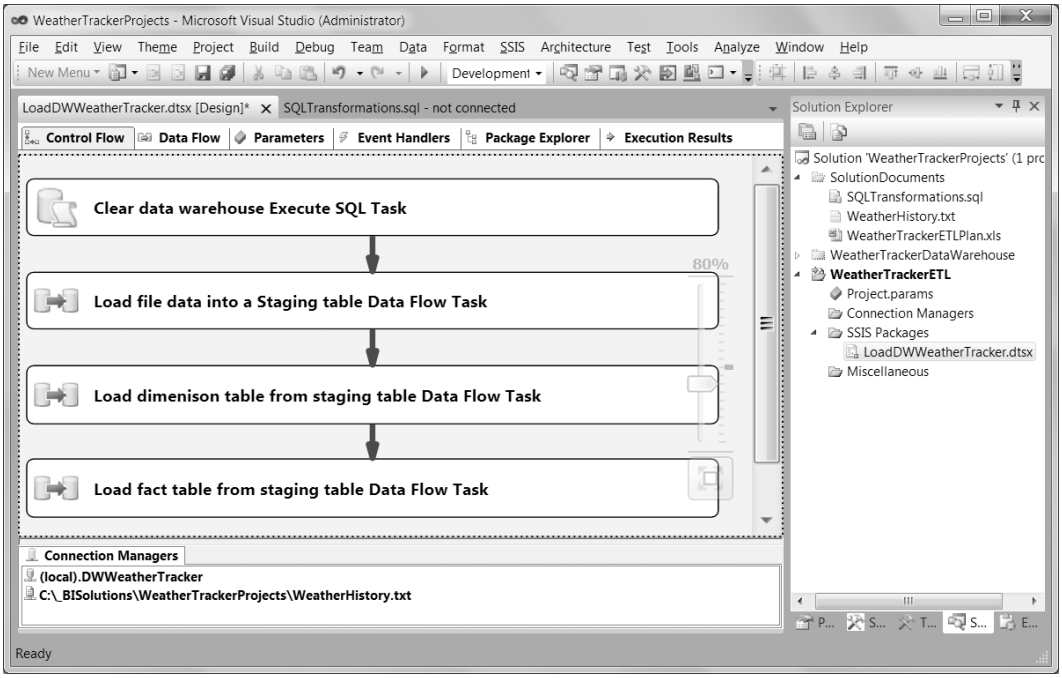


Figure 2-29. *Outlining Control Flow tasks and adding connections*

Each task must be added from the Toolbox onto the Control Flow surface and then configured. One of the first things to configure are the names of the tasks. Notice that we have configured our tasks to have a uniquely descriptive name. This is an important step because each package can have a large number of tasks within it. Without proper naming of both the package file and the tasks within the package, it will be confusing to you as well as to anyone who will be maintaining the package over time.

The tasks shown in Figure 2-29 include an Execute SQL task and three Data Flow tasks. There are also three Precedence Constraints shown, indicated by the arrows in Figure 2-29. Precedence Constraints arrows represent the flow of the tasks, that is, which task will run first, next, and last. You can create a precedence constraint by clicking one task and dragging the resulting—magically appearing—arrow to another task.

SSIS Connections

Each SSIS package needs one or more connection objects to perform the ETL processing. When a package is first made, it does not include any connections, but they can be added to the package from the Connection Manager tab (Figure 2-30). After you outline your SSIS package, you will have a good idea of what connections you will need and can begin to create the connection objects for your tasks. Connection objects can be created by clicking in the Connection Managers area at the bottom of the screen (Figure 2-30) and choosing a connection type from the context menu that appears.

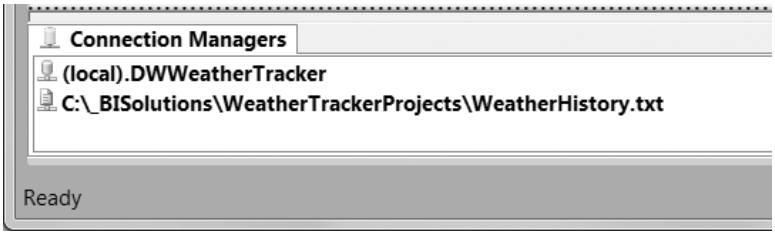


Figure 2-30. *Adding Connection objects*

■ **Note** We go into more detail about how to make connections in Chapter 7.

Configuring a Flat File Connection

SSIS can connect to text files, databases, and even web services. In our example, we connect to both the `WeatherHistory.txt` file, that contains the client's data, and the DWWeatherTracker database, which we created in the previous exercise. Note that each connection is also named accordingly.

To configure a flat file connection, use the Flat File Connection Manager Editor dialog window. All of these connection dialog windows have one or more pages. The pages are listed on the left side of the dialog window and the configurations for each page are displayed on the right (Figure 2-31). This is a common pattern throughout SSIS.

For example, in Figure 2-31, you can see that we configured the File name property, on the General page, of a Flat File Connection Manager Editor window. This is how SSIS knows which file to import data from.

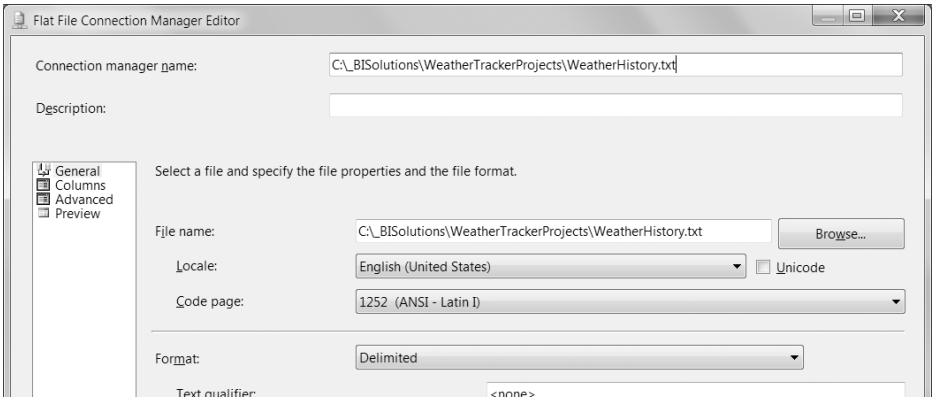


Figure 2– 31. *Configuring a flat file connection*

Configuring a SQL Server Connection

When configuring a SQL Server connection, the editing window allows you to identify the server name as well as the database name, as shown in Figure 2-32. This dialog window is almost identical to the one you used while connecting to the data warehouse in Exercise 2-2. Microsoft reuses this same dialog window in all of the BI projects, so expect to see it a number of times as you proceed through the book.

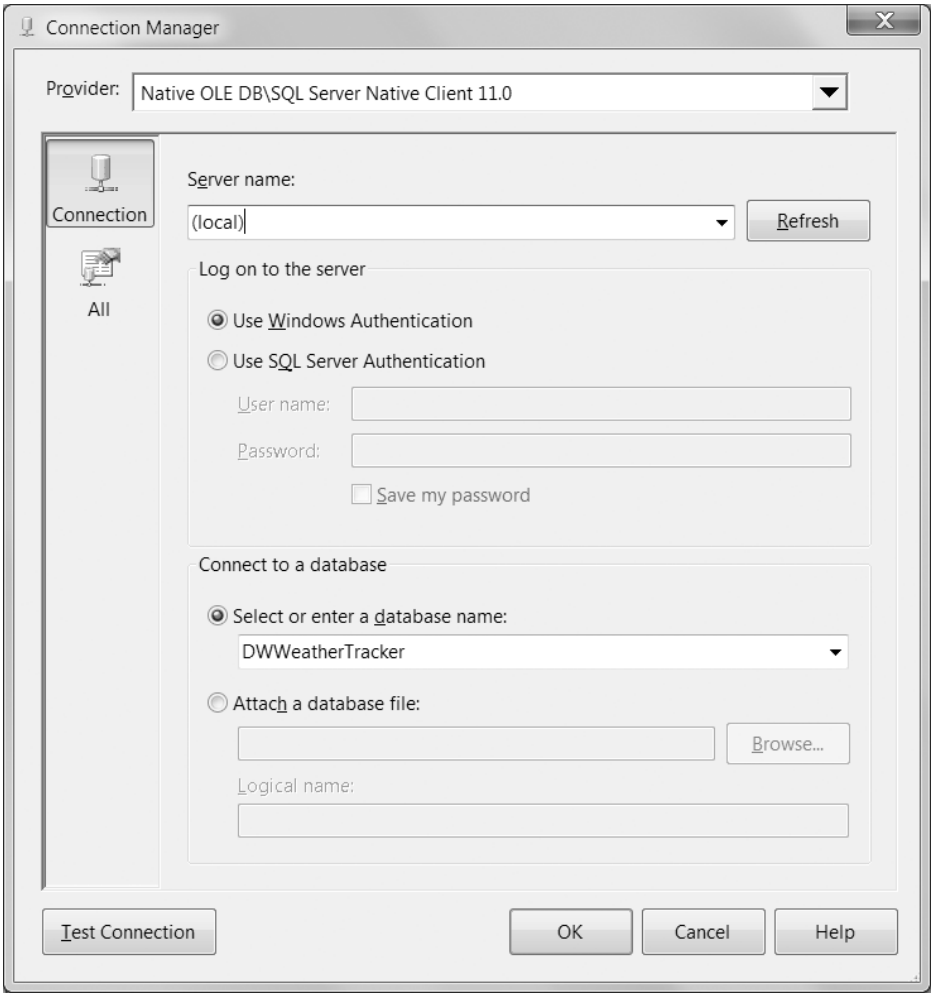


Figure 2-32. *Configuring a SQL Server connection*

After you have created and configured the connection, you can configure the SSIS tasks. We typically do so in the order defined by the precedence constraints. The first task in our package is an Execute SQL task, so we start there.

Configuring an Execute SQL Task

As the name implies, an Execute SQL task allows you to run SQL statements from SSIS packages. They are often used to clear out the data warehouse tables so they can be refilled with new data. This "flush and fill" technique works only with smaller data warehouses tables, but because it is the simplest technique, we will use it in our WeatherTracker ETL project.

Tables can be cleared by using a set of Delete From < table name > SQL commands like the ones shown in Figure 2-33. When SSIS runs an Execute SQL task, these SQL statements are executed on the connected SQL server.

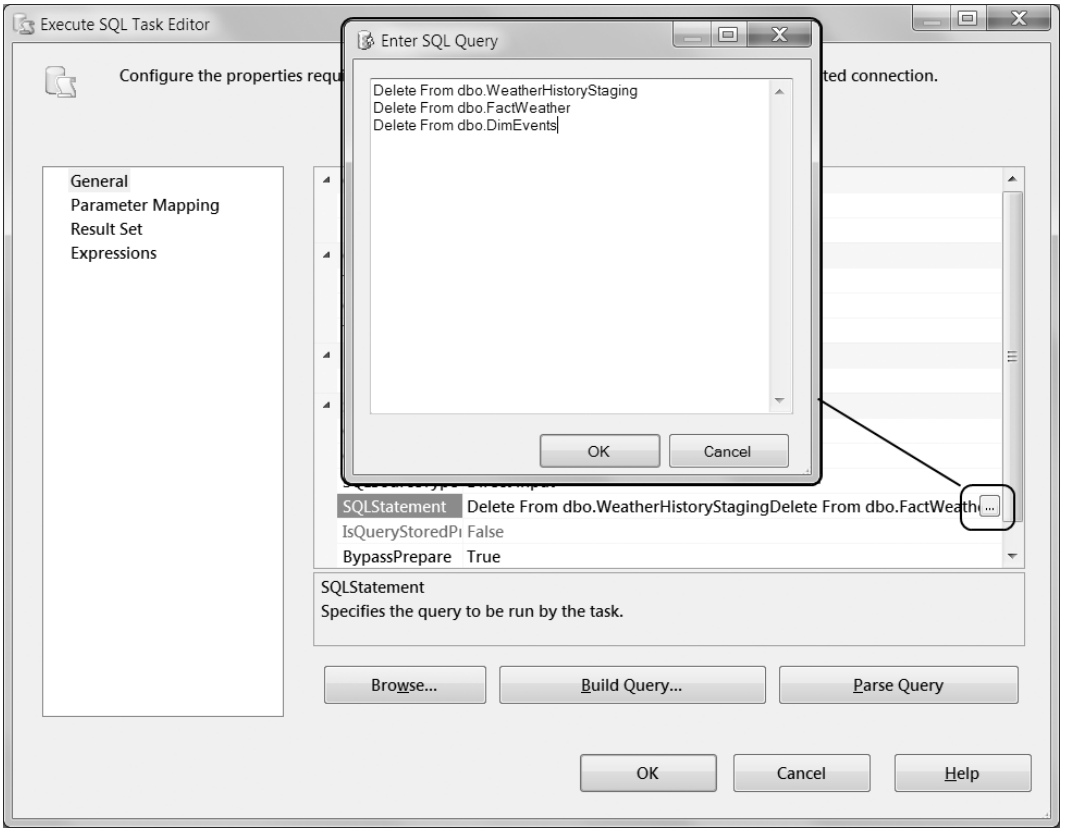


Figure 2-33. *Editing an Execute SQL task*

Configuring Data Flow Tasks

In addition to the Execute SQL tasks, Data Flow tasks are something you will use on a regular basis. Their purpose is to transfer data from one location to another.

Once you have placed a Data Flow task onto the Control Flow surface, you need to configure it. To configure a Data Flow task, you either double-click the Data Flow or simply highlight it and click the Data Flow tab, as shown in Figure 2-34. Both options take you to the same location and allow you to edit the Data Flow task.

Data Flows are somewhat unique in that they have their own editing tab and their own set of Toolbox items. If you watch the Toolbox as you switch between the Control Flow tab and the Data Flow tab, you can see Toolbox items change.

The Data Flow's Toolbox items are specifically designed to move data from one location to another and to apply transformations as the data moves from point to point. There is a large set of Toolbox items to choose from, and they can be grouped into categories. The first category is Data Flow Sources. These items allow you to pull data from text files or database tables.

The next category is Data Flow Transformations. These optional tasks provide ways to manipulate the data as it moves from the source to the destination.

The final category is Data Flow Destinations. These are used to connect to files or database tables that you want to fill with data. In summary, each Data Flow task consists of at least one source and one destination and optionally one or more transformations.

Because each data flow always has a source and destination, you can start outlining one source task and one destination task. In our example, we need a flat file source and a SQL Server destination (Figure 2-34).

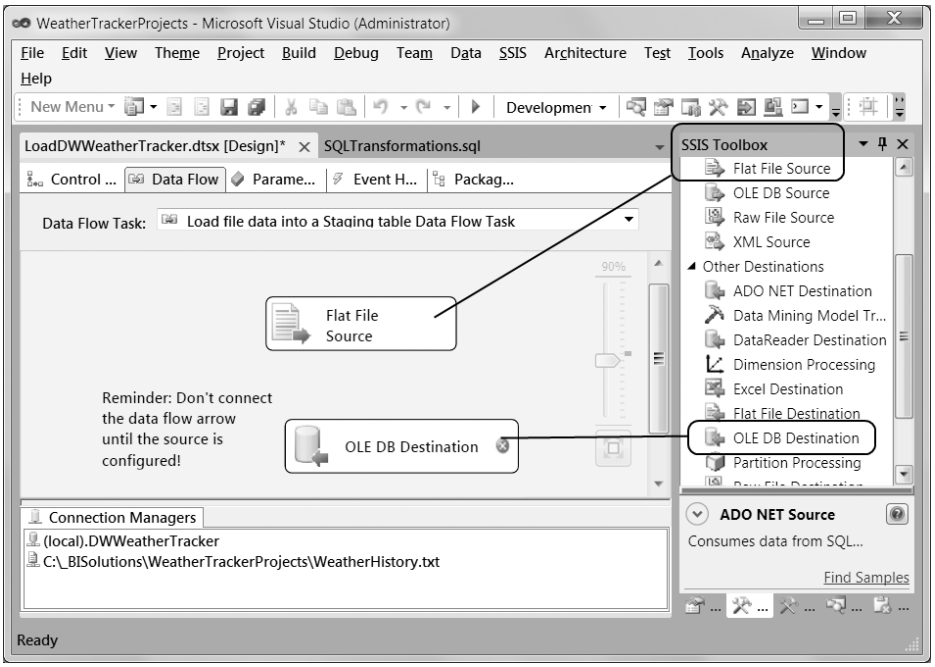


Figure 2-34. *Outlining the first Data Flow task*

When configuring a Data Flow task, it is important to configure the data source first before you configure the data destination. One common mistake is to outline the process of your data flow by putting a source and destination task onto the data flow surface and then connecting the tasks immediately. You do not want to connect the destination until after you have configured the data source. Doing it out of order will automatically, but improperly, configure the data destination.

■ **Note** If you mistakenly edit the destination task before you configure and connect the data source to it, the data destination becomes corrupt. The simplest way to resolve this is to delete the data destination and replace it with a new one from the Toolbox. Afterward, configure and connect the data source before you attempt to edit the destination task.

Configuring Additional Data Flows

Since a single SSIS package can consist of many Data Flow tasks, Microsoft made it easy for you to switch between them. At the top of the Data Flow tab there is a dropdown box labeled Data Flow Task. You can select between the individual Data Flow tasks that are part of your SSIS package using this dropdown box (Figure 2-35).

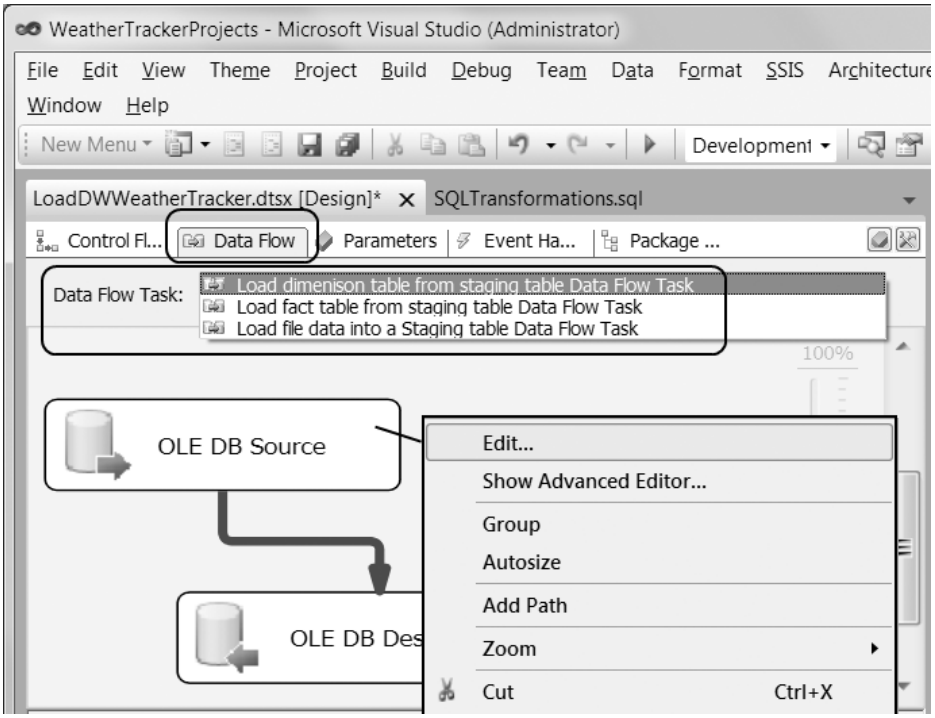


Figure 2-35. *Navigating between Data Flow tasks*

Once you focus on a selected Data Flow task, you can edit its SSIS items by either double-clicking them or using the Edit option from the context menu that appears when you right-click an item (Figure 2-35).

Many items that you configure have both a standard and advanced editor. Each has its own dialog window. The standard dialog windows have all the settings that you would commonly use, and in our example, they contain all the settings we need.

Configuring a Data Source

To use a data source, you must first configure it. For example, in an OLE DB source, there are three basic configurations you need to adjust: the OLE DB connection manager, the data access mode, and the SQL command text.

In Figure 2-36, you can see that we configured the OLE DB source to use the (local).DWWeatherTracker connection manager, one of the two connection objects that we created earlier (Figure 2-32).

We also configure to use a SQL command as the data access mode, which allows you to use a SQL statement instead of just the name of a table in a database. Using a SQL statement is preferred since you can filter out columns or rows you do not want. You can also apply transformations, such as data conversions, when the SQL code executes. This is a simple and effective way to transform the data as it is retrieved (Figure 2-36).

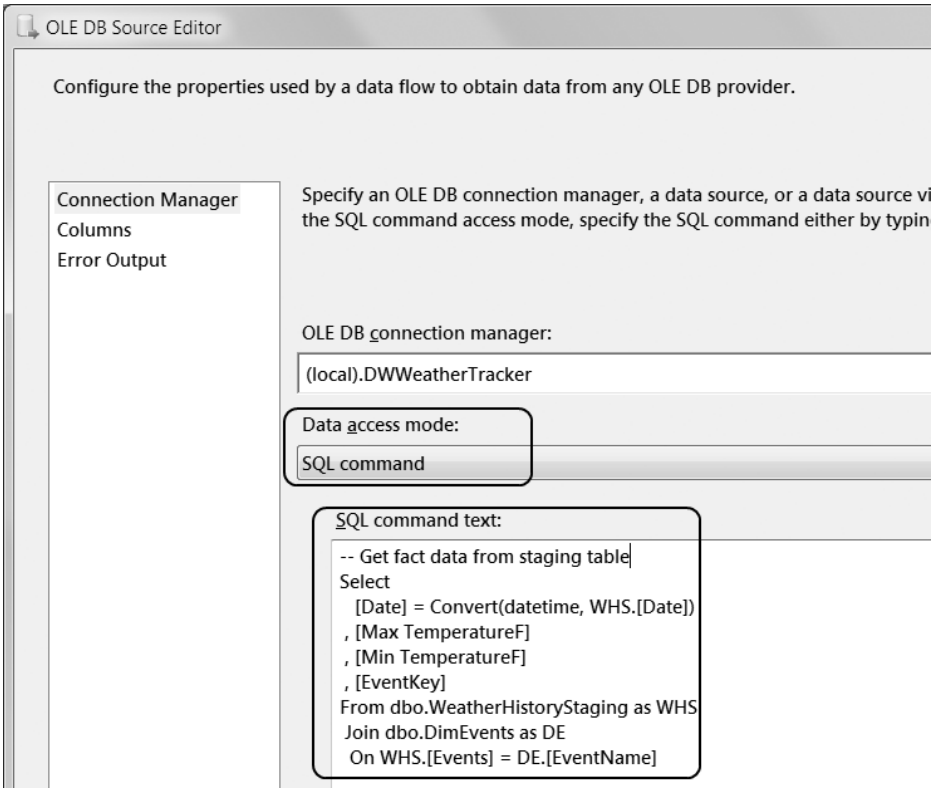


Figure 2-36. *Using a SQL statement to refine the data source*

Executing an SSIS Task

Once you have created and configured an SSIS package, test your work by executing the package. To do this, right-click the package in Solution Explorer and select Execute from the context menu, as shown in Figure 2-37.

Executing the SSIS package may take a while as your installed SSIS service reads the underlying XML code instructions, attempts to make the connections to the text file and the database, and then performs the extraction, transformation, and loading tasks.

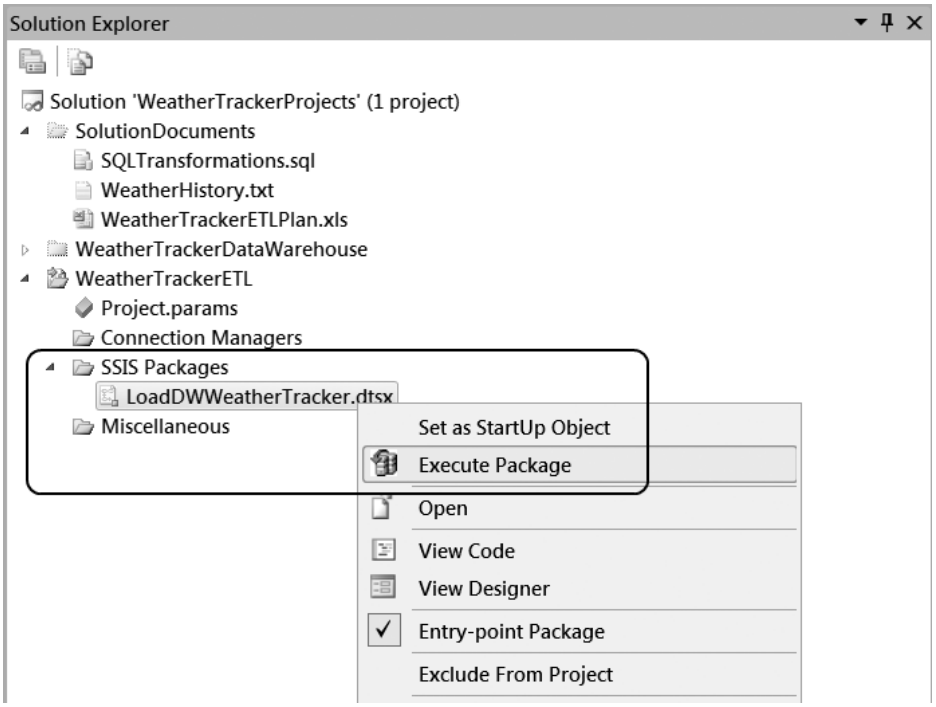


Figure 2-37. *Executing the package*

Completing the Package Execution

While a task is executing, it shows an indicator icon on the right side of the task (Figure 2-38). As each individual task processes, the icon changes from a yellow wheel icon to a green check mark icon once it completes successfully.

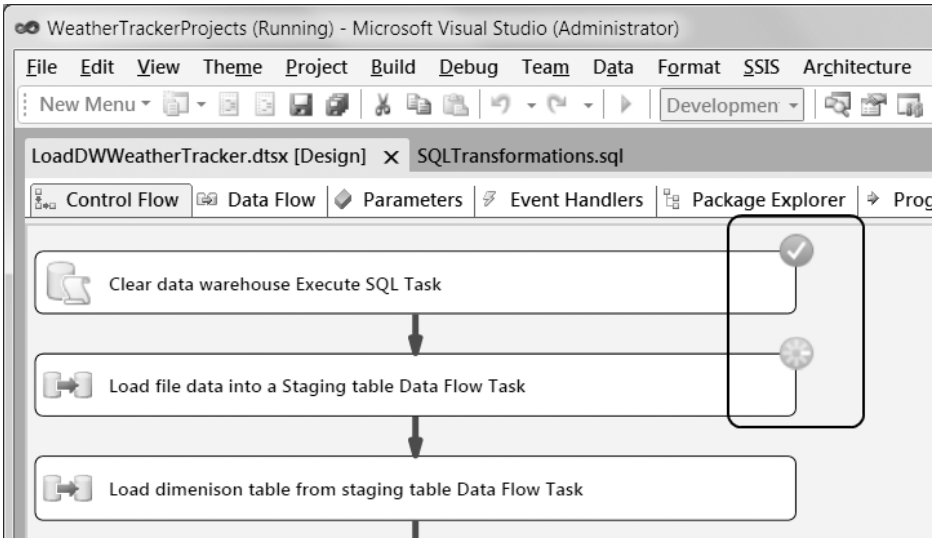


Figure 2-38. *The package while it is executing your code*

If SSIS encounters an error, the task that is causing the problem displays a red X icon, and the execution of the package comes to a halt.

When all of the tasks complete, successfully or not, manually stop its debugging process using the Debug menu at the top of the Visual Studio window or by selecting the stop debugging hyperlink at the bottom of the Visual Studio window (Figure 2-39).

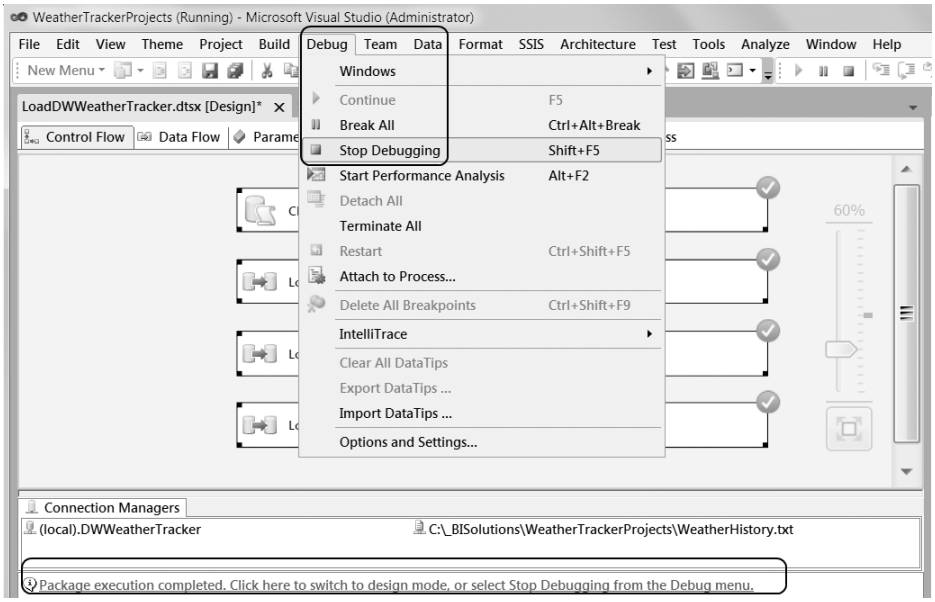


Figure 2-39. *Stopping the execution*

You now have an overview of how to create, configure, and execute an SSIS package. In Chapter 7 we look at this in depth, but even at this level you should have a pretty good feel for the process. It is now time to get some practice by doing another exercise in which you add our existing package to a new SSIS project you create. You then verify its configurations and finally execute it to fill up your data warehouse.

EXERCISE 2-3. ADDING AN SSIS PROJECT TO YOUR SOLUTION

In this exercise, you make a new SSIS project and add it to your current Visual Studio Solution. You then add an existing SSIS package to the project that is included in the downloadable content. After the package has been added to your solution, verify that the connections are configured correctly and finally execute the package. Completion of this exercise is required to complete the other exercises throughout this chapter.

1. Visual Studio should still be open from the previous exercise, but if not, please open it and access the WeatherTrackingProject solution from the File ➤ Recent Projects and Solutions menu. Remember to always use Visual Studio as an administrator by right-clicking the menu item, selecting Run as Administrator, and then answering Yes to close the UAC pop-up window.

Add a New Project to a Current Solution

First, we add a new project to the WeatherTrackerProjects solution. The following steps guide you through this process.

1. With the WeatherTrackerProjects Visual Studio solution open, click the solution 'WeatherTrackerProjects' icon in Solution Explorer. (The Visual Studio menus are context sensitive, and we want to be working with the solution itself and not any of the solution folders.)
2. Use the File menu to add a new project to your current Visual Studio solution by clicking File ➤ *Add* ➤ New Project. (Make sure you do not choose File ➤ New ➤ Project. The two options are easily confused.) See Figure 2-24.
3. When the Add New Project dialog window opens, select the Business Intelligence ➤ Integration Services option on left side of this window, and on the right side, select Integration Services Project from the options. See Figure 2-25.
4. At the bottom of this dialog window, in the Name textbox, type **WeatherTrackerETL**. Verify that the Location textbox reads C:_BISolutions\WeatherTrackerProjects. See Figure 2-25.
5. Click OK to close this dialog box.

Add an Existing SSIS Package to the Project

Once the WeatherTrackerETL project has been created, it appears in your Solution Explorer window. We are now going to add an existing package to the WeatherTrackerETL project.

1. Select the SSIS Packages folder and right-click the `Package.dtsx` file to access the context menu. This is similar to Figure 2-26, but we choose a different option from the context menu.
2. We do not need the `Package.dtsx` file, so click the Delete context menu option. A pop-up message box appears asking whether you are sure you want to permanently delete this. Click OK to continue.
3. Now we add a premade `.dtsx` file to the project from the downloadable book files. Right-click the SSIS Packages folder and select the Add Existing Package option from the context menu. See Figure 2-40. The Add a Copy of Existing Package dialog window appears (Figure 2-41).

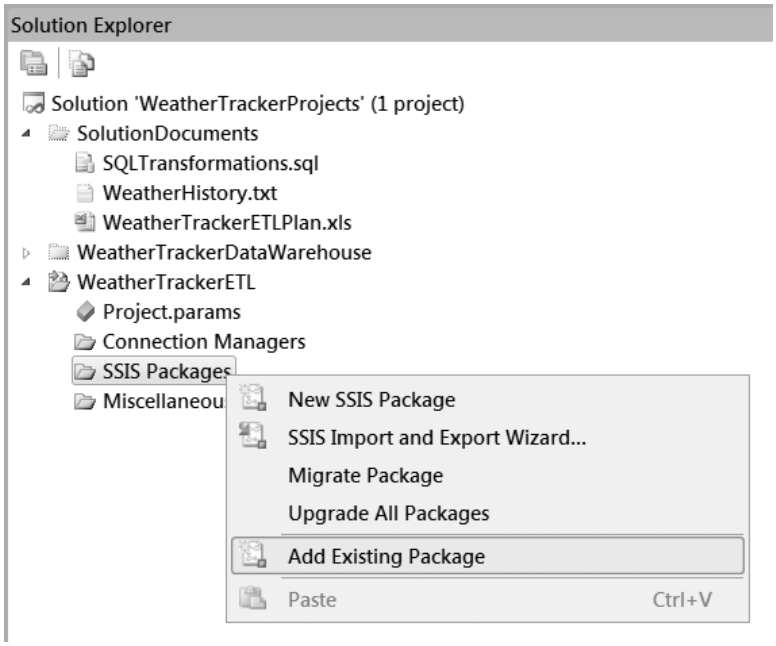


Figure 2-40. *Adding an existing SSIS package to the project*

4. Within the Add Copy of Existing Package dialog window, select File System from the Package Location drop-down box (Figure 2-41).

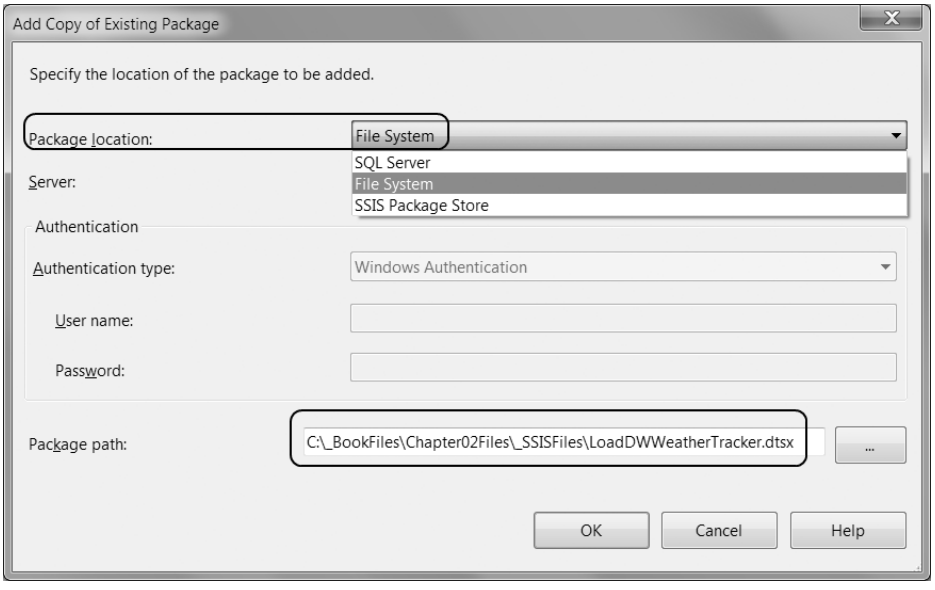


Figure 2-41. *Identifying the path to the existing SSIS package*

5. For the Package Path textbox, click the ellipsis button, browse to the file `C:\_BookFiles\Chapter02Files\_SSISFiles\LoadDWWeatherTracker.dtsx`, and click Open. Then click the OK button to close the dialog window and have Visual Studio copy the `.dtsx` file to your SSIS project. The file should now appear in Solution Explorer.

6. Once the file is added, verify in the Visual Studio Properties window that the package file was copied from the proper file. (If you do not see the Properties window, you can open it by right-clicking `LoadDWWeatherTracker.dtsx` and selecting Properties.) The path should show the `_BISolutions\...` path (not `_BookFiles`).

Verify the Connections

Whenever you are working with someone else's SSIS package, there is a chance that the SQL server instance name on their computer is not the same as the one on your computer. At this point, we need to make sure that the connections are correct for your computer by opening and editing the package.

1. To open and edit the package, double-click the `LoadDWWeatherTracker.dtsx` package file in Solution Explorer or right-click the file and choose Open from the context menu. The package opens, and the Control Flow tab displays the four tasks shown in Figure 2-29.

2. Locate the two connections under the Connection Manager tab at the bottom of Visual Studio, as shown in Figure 2-30.

3. Right-click the connection called (local).DWWeatherTracker (Figure 2-42), and uncheck the Work Offline option from the context menu, as needed.

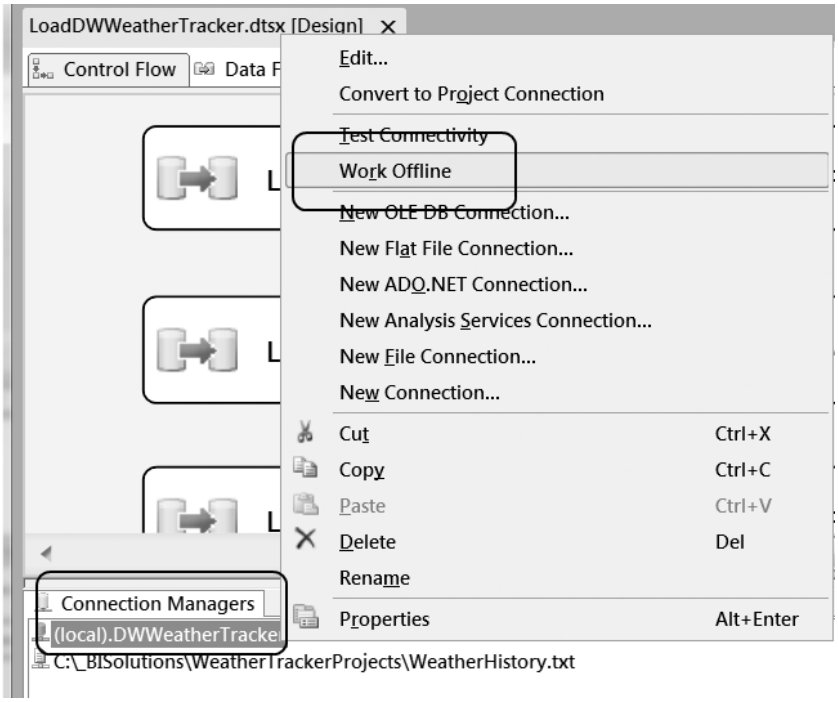


Figure 2-42. *Unchecking the Work Offline option*

4. Double-click the first connection called (local).DWWeatherTracker (Figure 2-42).
5. When the Connection Manager dialog window appears, click the test connection button at the bottom of the window to verify that you can connect to the data warehouse database. A window will open with the text Test Connection Succeeded (Figure 2-32). If this does not occur, change the name of the server in the Server Name dropdown box to match the name of your SQL server installation and try the test button again.

Important: You do not need or want to change the SSIS connect name even if you change the SQL Server name.

6. Double-click the second connection called WeatherHistoryTextFile. When the Flat File Connection Manager editing window appears, verify that the file name is pointing to the `C:\_BISolutions\WeatherTrackerProjects\WeatherHistory.txt` text file (Figure 2-31).
7. Click the Preview page on the left side of the dialog window. After verifying that the data looks like Figure 2-43, click OK to close it.

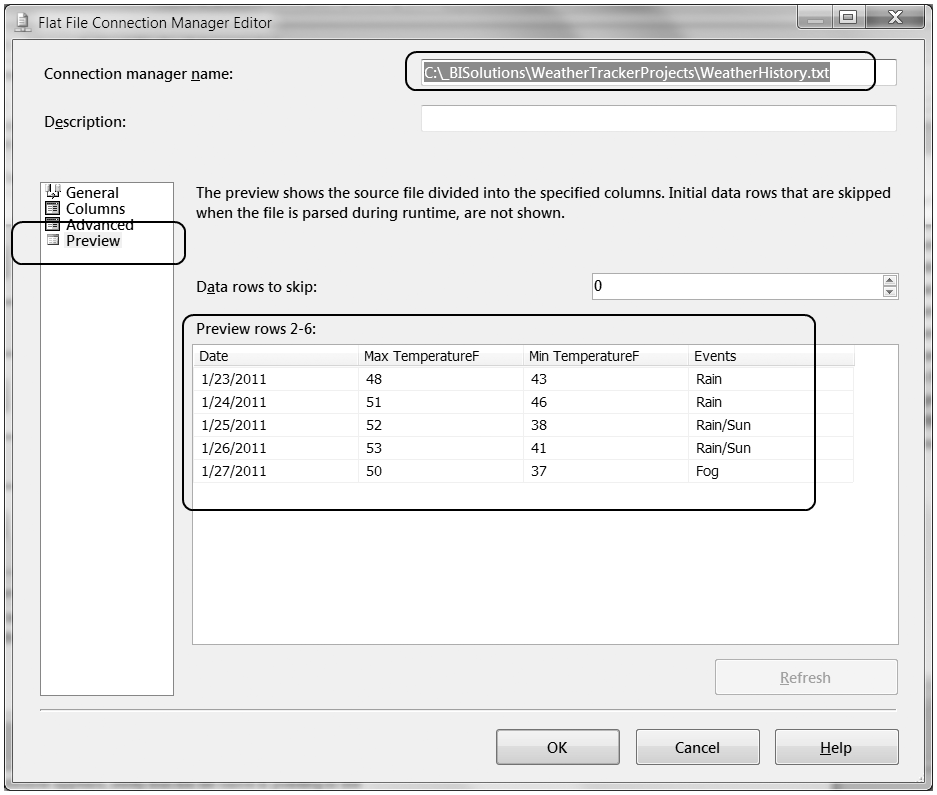


Figure 2-43. *Verifying the file connection*

Important: If the file connection does not work, please check that you correctly placed the `_BISolutions` folder on the `C:\` drive. (You can review Exercise 2-1 for more information.) Click the Data Flow tab at the top of your

screen and review the three data flows that are listed in the Data Flow Task drop-down box within this tab (Figure 2-35). In future chapters, we show you how to create these types of tasks yourself. At this point, you are just reviewing these to familiarize yourself with the process.

Execute the SSIS Package

After you have reviewed the Data Flow tasks and connections, it is time to run the package and fill the data warehouse.

1. While in the Control Flow tab, right-click the `LoadTablesFromFiles.dtsx` package in the Solution Explorer and select Execute Package from the context menu (Figure 3-37).
2. The package will run, and all the tasks on the Control Flow designer window should finish with a green checkmark icon (Figures 2-38 and 2-39).
3. Once the package has completed running, use the Debug ➤ Stop Debugging menu option to stop the package execution (Figure 2-39).

In this exercise, you created an SSIS project within your Visual Studio solution. You then added a preexisting package from the downloadable content included with this book. Then you verified that the package connections were valid, reviewed the tasks that were configured, and executed the package.

In the next exercise, you will add a premade SSAS cube project to this same Visual Studio solution, so please leave Visual Studio open for now.

Creating a Cube

Great! So now, you have a data warehouse filled with data. It is time to create a cube that uses it! The most common way to make a cube is by using the SQL Server Analysis Server (SSAS) template in Visual Studio.

Like SSIS, you can add an Analysis Server project to an existing Visual Studio solution. Do this using the File ➤ Add ➤ New Project menu item within Visual Studio. When the Add New Project dialog window appears, select the Analysis Server Project template from among the Business Intelligence projects, as shown in Figure 2-44.

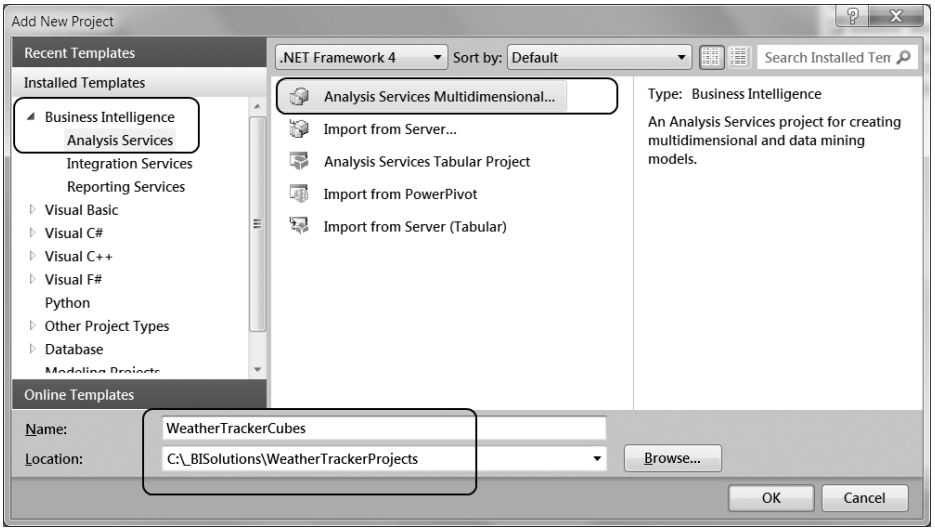


Figure 2-44. *Adding a new SSAS project to the WeatherTrackerProjects solution*

In this dialog window, you have a choice of several options. The choice we focus on in this chapter is the Analysis Services Multidimensional Project (Figure 2-44).

As always, you should give the project a descriptive name that indicates what the project is used for. In Figure 2-44, we have named the project WeatherTrackerCubes.

After clicking the OK button, a new SSAS project appears in Solution Explorer, as shown in Figure 2-45. Note that you now have two BI projects and two Solution folders in this one Visual Studio solution.

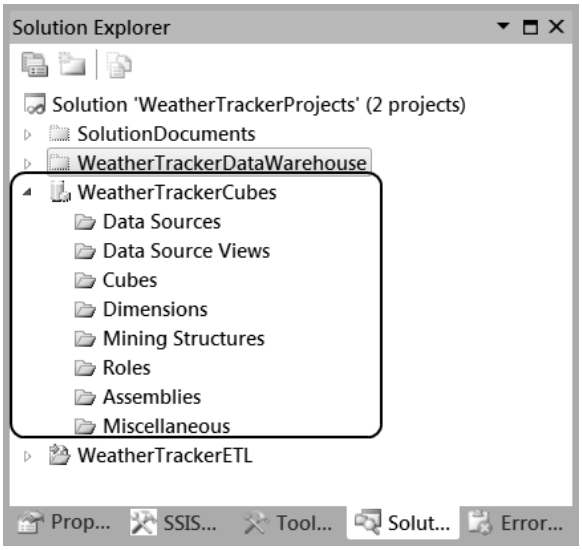


Figure 2-45. *The new SSAS project in Solution Explorer*

Making a Connection to the Data Warehouse

You must have a source of data to make a cube. Most often, this source is a data warehouse such as the one that we built at the beginning of this chapter. In an SSAS project, you are able to connect to the data warehouse by making a data source object. A data source can be created by right-clicking the folder named Data Sources in the Solution Explorer window and selecting New Data Source from the context menu, as shown in Figure 2-46.

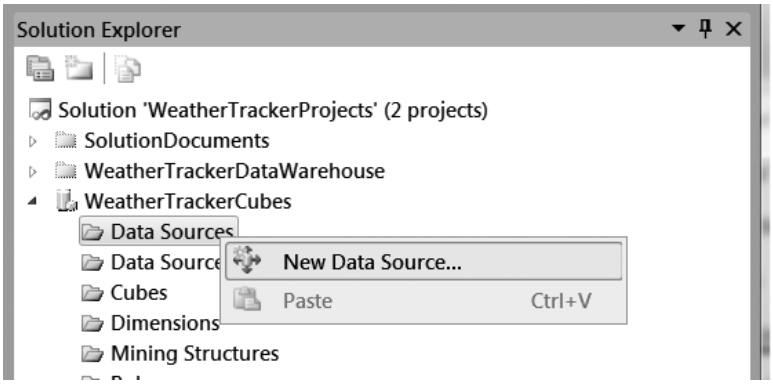


Figure 2-46. *Adding a data source*

While working in SSAS, a wizard will help you create a data connection. As we will see in Chapters 9 through 12, most of the objects in SSAS are created using one wizard or another.

The Data Source Wizard starts with the welcome screen, but it does not provide much information. Clicking Next, however, moves you to the next page, which then allows you to either create a new data connection or use an existing one. Because SSAS is a project inside a Visual Studio solution, if you have created a previous connection to a database, the existing connections listed are ones that Visual Studio remembered from previous projects. As you can see in Figure 2-47, the DWWeatherTracker database is an example of a previously created connection. Therefore, you will be able to select it as an existing data connection on the Data Source Wizard's second page.

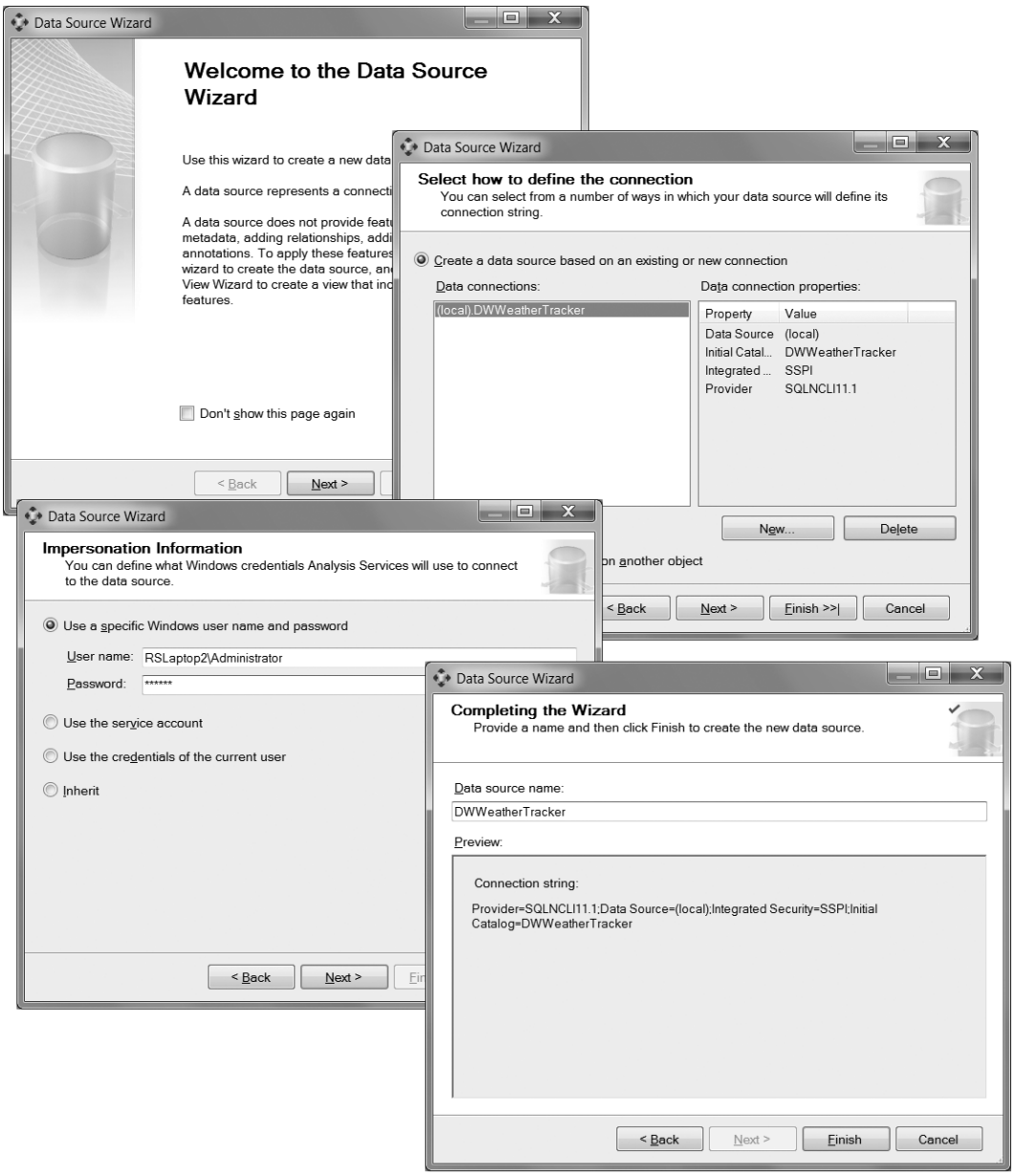


Figure 2-47. *Creating an SSAS data source*

As you proceed to the next page, you are asked for impersonation information. This is an important screen, because it defines the account that will be used to connect to the Analysis Server from Visual Studio and also from the Analysis Server to the data warehouse. This is another topic that is discussed further in Chapter 9.

The best choice is to enter your computer name followed by a slash and a personal account name. Make sure the account you use has access to both the Analysis Server and the SQL Server where your data warehouse is hosted. In Figure 2-48, Randal has an account on his laptop computer called Administrator,

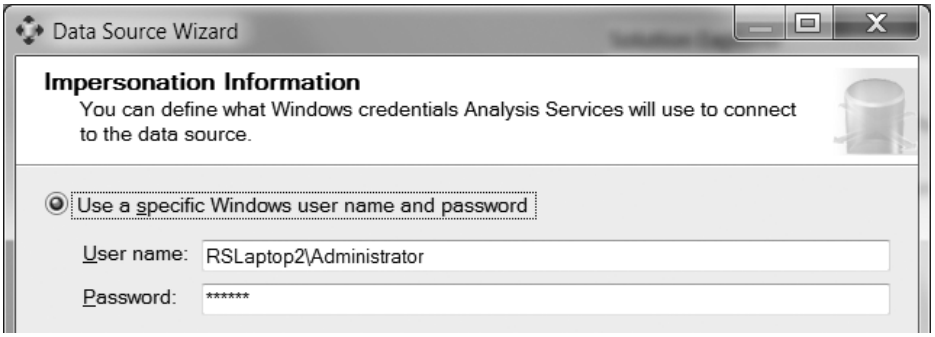


Figure 2-48. *Configuring the impersonation information*

so he typed in **RSLaptop2\Administrator**. Note the direction of the slash. On web addresses you use a forward slash, but here it must be a back slash.

■ **Note** Using a full administrator account is not recommended in a production environment, but for demonstration purposes, it is a good choice because it resolves a lot of issues that can be difficult to troubleshoot. In this book, we use an administrator account for all of our exercises. If you are particularly familiar with Windows, SQL Server, and Analysis Server security, you can use an account that has restrictive privileges. Otherwise, we highly recommend using an administrator account for the book exercises. Search the Web for "True Windows Administrator" for more information.

The next screen of the wizard allows you to name the connection and finish the wizard. The wizard puts spaces between any words it believes to be a concatenation of multiple words. It does this simply by looking for uppercase and lowercase letters. Although this may make it more readable, our experience has shown that spaces in names cause problems for some computer programs. Therefore, we can either take out the spaces that the wizard puts in or just change the name altogether.

Creating a Data Source View

After the connection is made, you need to create a data source view. A data source view provides the foundation of any cubes or dimensions you create. Since Analysis Server 2005, you no longer build cubes as dimensions directly against the data source. Instead, Microsoft provides an abstraction layer represented by the data source view (Figure 2-49).

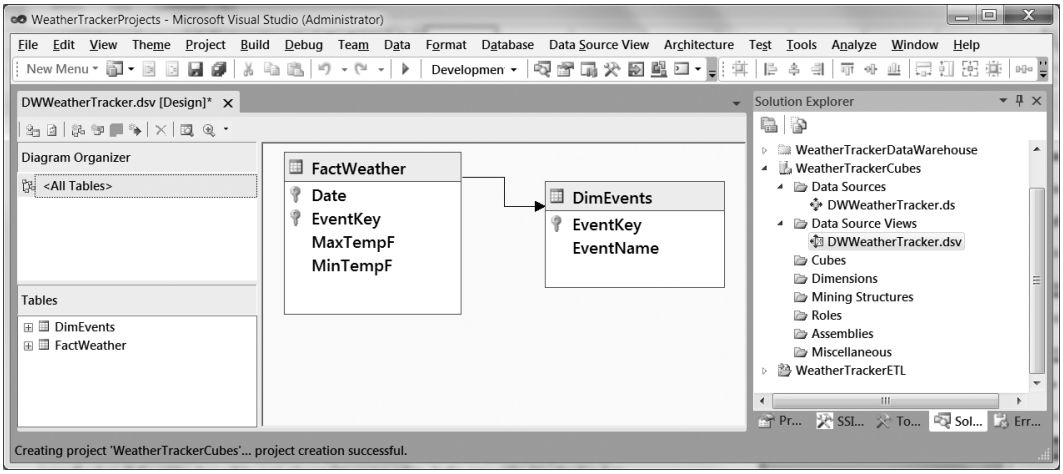


Figure 2-49. *An SSAS data source view*

In the data source view, you can add any tables that you want to use for the creation of both cubes and dimensions. In this example, we need only our single fact table and the single dimension table, but normally you would have many dimension tables associated with one or more fact tables.

One important aspect of the data source view is the relationship line between the fact tables and the dimension tables. This is similar to a foreign key relationship. If your data warehouse has a foreign key relationship between the tables, the wizard automatically adds relationship lines for you. Otherwise, you have to add them yourself.

It may be perplexing that the relationship lines do not visually attach to the columns between the tables, but don't worry. The columns are connected even though the data source does not display it that way. If you were to double-click the relationship line, you would see a dialog box that opens indicating which columns are connected.

As with most things in SSAS, you use a wizard to create the data source view. To start the wizard, right-click the Data Source Views folder in Solution Explorer and select New Data Source View from the context menu.

When the wizard launches, it shows a welcome screen (Figure 2-50). Click the Next button at the bottom of this screen to proceed to the screen you are able to configure. The configuration on the next page involves selecting the data source you will be using. If there is only one data source, the choice is pretty obvious. Therefore, simply click the next button to move to the next page.

On this page, select the tables that are to be included in the data source view. As we mentioned previously, you need to select only the tables required for your cubes. In the case of the WeatherTracker project, this would be the DimEvents and FactWeather tables.

The fourth page of the wizard allows you to name your data source view and finish the wizard. Once again, the wizard provides extra spaces in the name, and you may want to remove them.

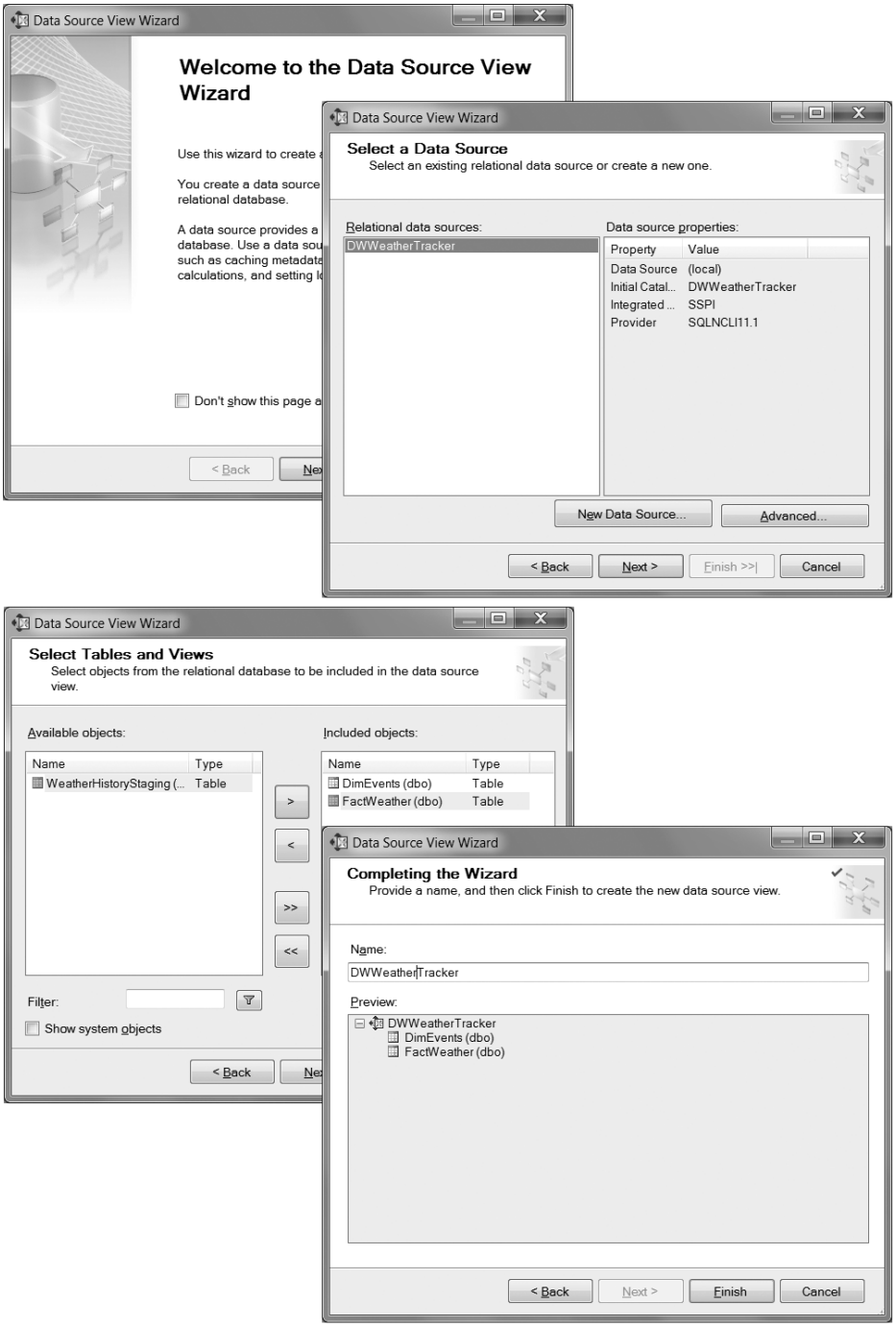


Figure 2-50. *Creating an SSAS data source view*

Creating Dimensions

It is probably not surprising that you use a wizard to create dimensions as well. Start the wizard by right-clicking the Dimensions folder in Solution Explorer. Right-clicking this folder brings up a context menu, and from there you can select the New Dimension option (Figure 2-51).

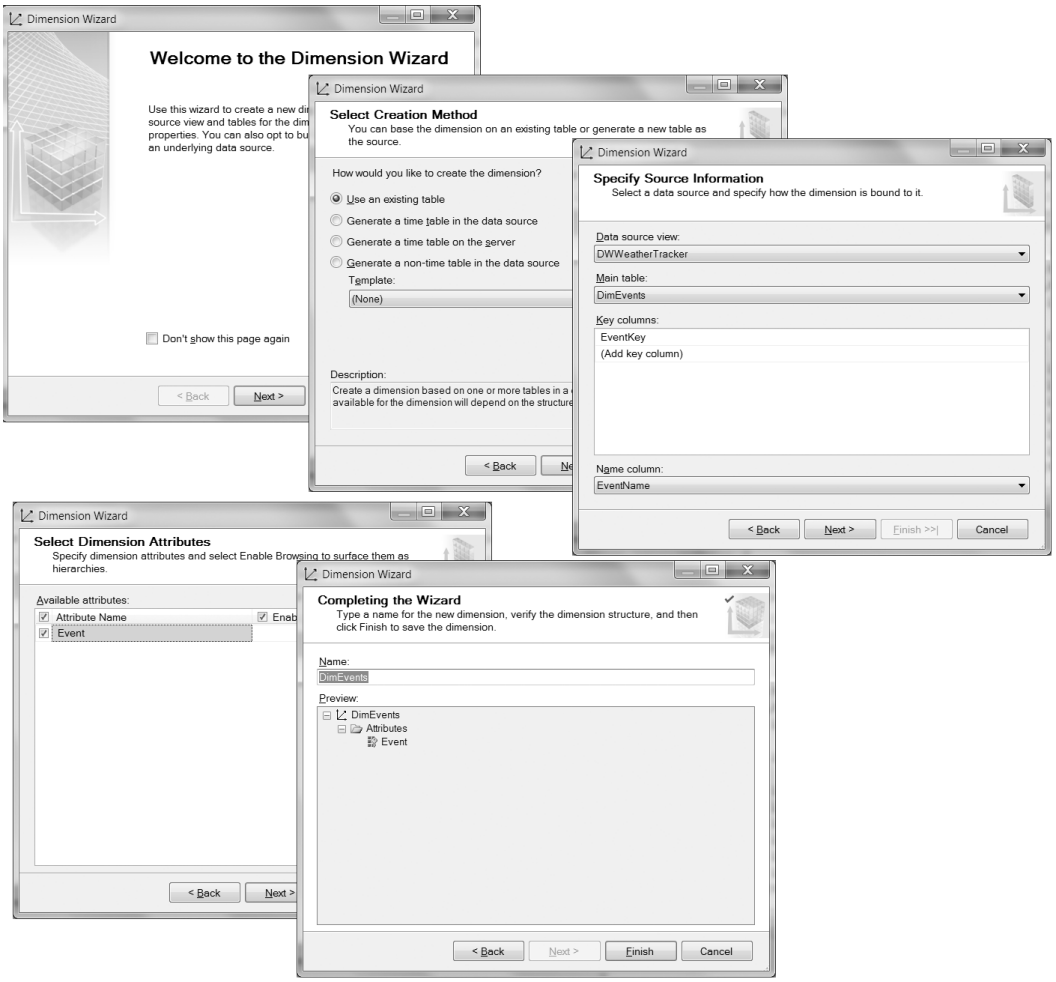


Figure 2-51. *Using the Dimension Wizard*

When the wizard launches, you are given a choice to use existing tables or create new tables within the data warehouse. The most common option is to use the tables you created in an existing data warehouse.

The third page of the wizard allows you to select which data warehouse dimension table you would like to base the new SSAS dimension upon. In our example, we use the DimEvents table. While choosing a table, you also configure the Key and Name columns by using the drop-down boxes on the wizard page. We can use the EventID, for example, as the key column and the EventName for the name column. SSAS uses the key column internally, but it displays the name column in the reporting applications, such as Microsoft's Excel or Reporting Services.

After clicking the Next button, you have a chance to rename your Event ID column to a friendlier name. This is how the column will appear in reporting applications. Because the reporting application will see the value of the name column, keeping the name EventId is not logical, so we change it to Event.

Clicking the Next button once again brings you to the final page of the wizard, which allows you to name the dimension and finish the wizard.

We need to create a dimension for the dates as well. To do this, we once again start the wizard and proceed through its pages, but this time we use the fact table as a dimension table. We do this because all the dates we need are in the fact table. As you can see from this example, a fact table can occasionally be used as a dimension table. We will tell you more about this in Chapters 4 and 9.

Creating Cubes

Creating a data source, a data source view, and the dimensions are all preparatory to creating a cube. The act of creating a cube is done using another wizard (Figure 2-52).

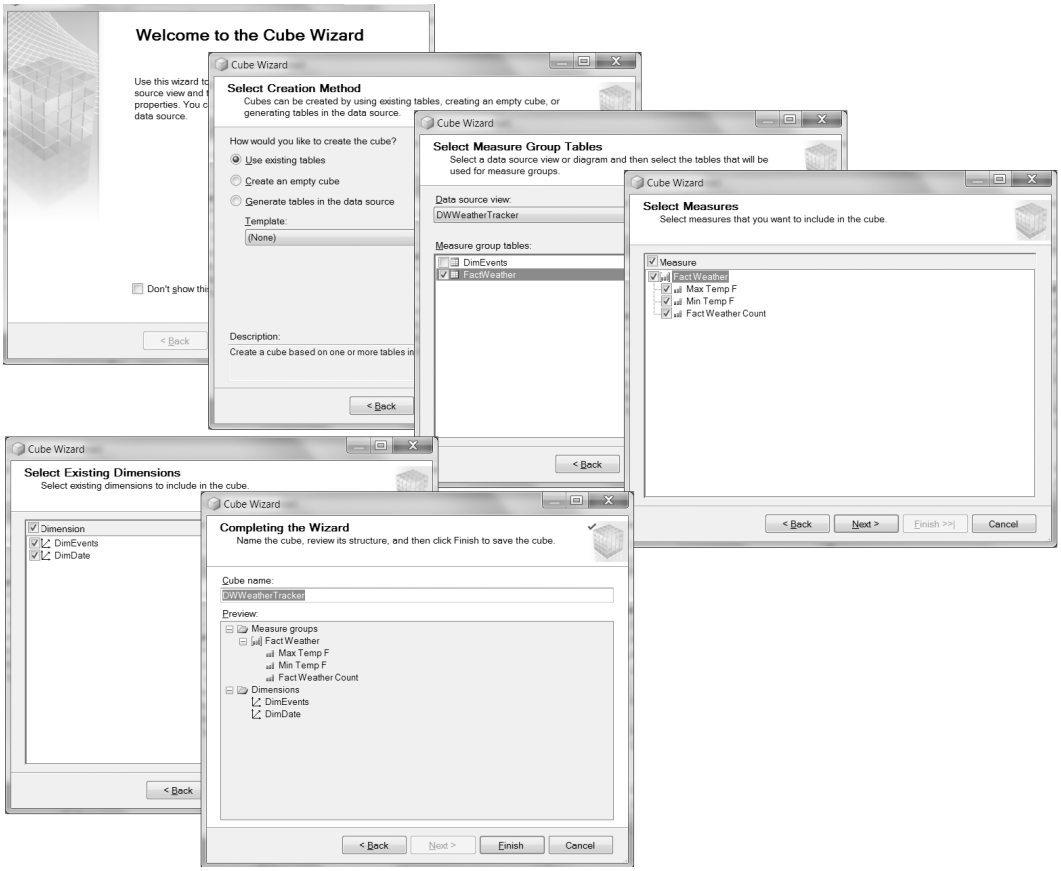


Figure 2-52. *Using the SSAS Cube Wizard*

We cover all of the different pages of this wizard in Chapter 10, but here is a quick overview:

- The first page of the wizard is a welcome page.
- The second page allows you to use existing tables or create new tables similar to the Dimension Wizard.
- On the third page, the wizard allows you to select which tables represent your fact tables.
- The fourth page allows you to select your measures.
- On the fifth page of the wizard, you are allowed to select your existing dimensions.
- The final page of the wizard allows you to name the cube and finish the wizard.

Deploying and Processing

With the completion of this wizard, all of the code needed to create the cube, dimensions, data source views, and data connection will be in a set of Visual Studio XML files. Analysis Server projects use an XML-based programming language that is similar to Integration Services. Each SSAS object created using the wizard creates XML files that represent those objects.

For the XML code to be translated into something that the Analysis Server can understand, you must build the Visual Studio solution. This building process creates a master XML file that can then be uploaded to the Analysis Server.

To upload the file to the Analysis Server, you need to deploy it. This act of deploying is effortlessly accomplished using Visual Studio's menu options. When the XML code deploys to the Analysis Server, the data source, data source view, dimensions, and cubes are all created on that SSAS server.

The cubes and dimensions will not have any data yet. To fill the cubes and dimensions with data, you must process both of them. The act of processing copies data from the data warehouse to the SSAS objects. This is accomplished from the Visual Studio's menu options.

We realize that was a very fast overview and that you likely have many questions, but we discuss all of this again in Chapter 10 in more detail. For now, let's have you add the SSAS project we created for this chapter to your current BI solution in this next exercise.

EXERCISE 2-4. ADDING AN SSAS PROJECT TO YOUR SOLUTION

In this exercise, you add the authors' completed Analysis Server project to your own Visual Studio solution. You verify that the connection in the data source is valid for your machine and then review the data source view, dimensions, and cube. You then build the XML code in project files into a master XML code file, deploy the master XML code file to your SSAS server, and finally process the data from the data warehouse into the SSAS objects.

Completion of this exercise is required to complete the other exercises throughout this chapter.

1. Visual Studio should still be open from the previous exercise, but if not, open it running as administrator (right-click the Visual Studio menu item, select Run as Administrator, and answer Yes to close the UAC pop-up window), and access the WeatherTrackingProject solution from the File ➤ Recent Projects and Solutions menu.

Add an Existing Project to a Current Solution

The first thing we do is let Visual Studio copy all of the SSAS project files from their current location in the downloadable book files folder, `C:\_BookFiles`, to your solution folder. We do that by adding an existing project to the current solution.

1. Using Visual Studio's menu, select the File ➤ Add ➤ Existing Project option (Figure 2-53), and the Add Existing Project dialog window appears.

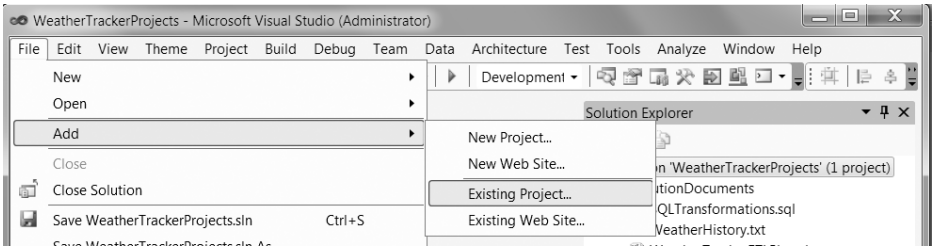


Figure 2-53. *Adding an existing project to your Visual Studio solution*

2. Locate the WeatherTrackerCubes project folder that you unzipped from the authors' downloadable files (Figure 2-54). They are at this location on your hard drive: `C:\_BookFiles\Chapter02Files\_SSASFiles\WeatherTrackerCubes.`

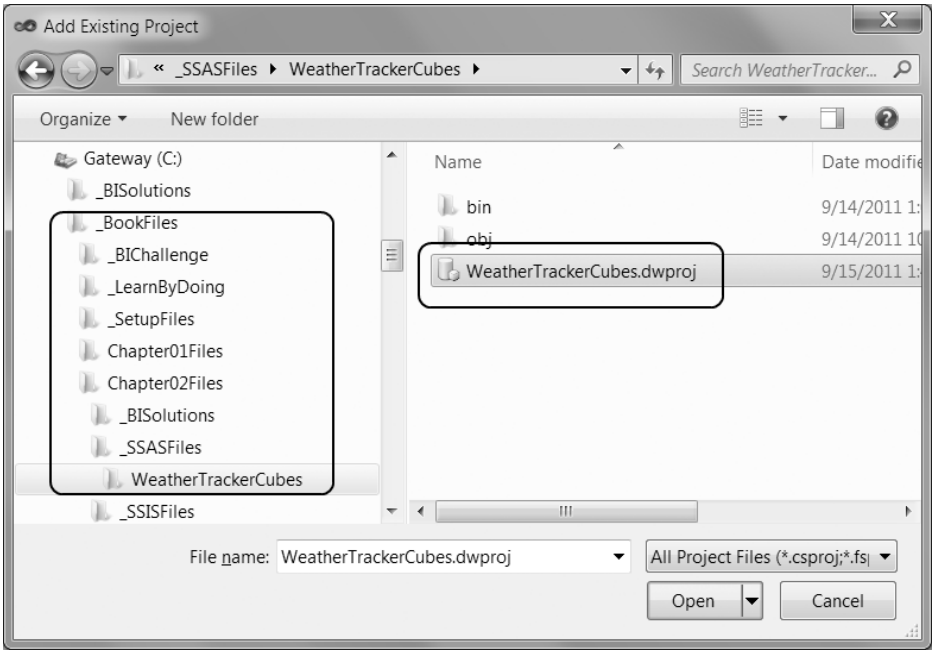


Figure 2-54. *Locating the SSAS project file*

3. Once you have located this folder, look inside to find the Analysis Server Project File, `WeatherTrackerCubes.dwproj`. Note that the file extension may not show in the title, depending on your operating system's settings, but the icon and type description will look like Figure 2-54.
4. Click this file; then click the Open button to close the dialog window. (A warning may appear to remind you to only open projects from a trustworthy source. If it does, click OK to continue.)
5. The downloadable SSAS project files will now be added to your Visual Studio solution.

Configure the Data Connection

Whenever you use a Visual Studio Project that was prepared on someone else's computer, you will likely need to change, or at least check, that the connections are appropriate to your own computer. Let's do that now.

1. After Visual Studio finishes adding the project to your WeatherTrackerProjects solution, it appears in the Solution Explorer window. If it is not expanded, click the arrow to the left of the SSAS project name to expand the Solution Explorer tree, as shown in Figure 2-49.
2. Locate the `WeatherTrackerDW.ds` file under the Data Sources folder. Right-click this file and choose the View Designer option from the context menu. The Data Source Designer dialog window appears, as shown in Figure 2-55.
3. Once the dialog window appears, click the Edit button to display the Connection Manager dialog window (Figure 2-55).

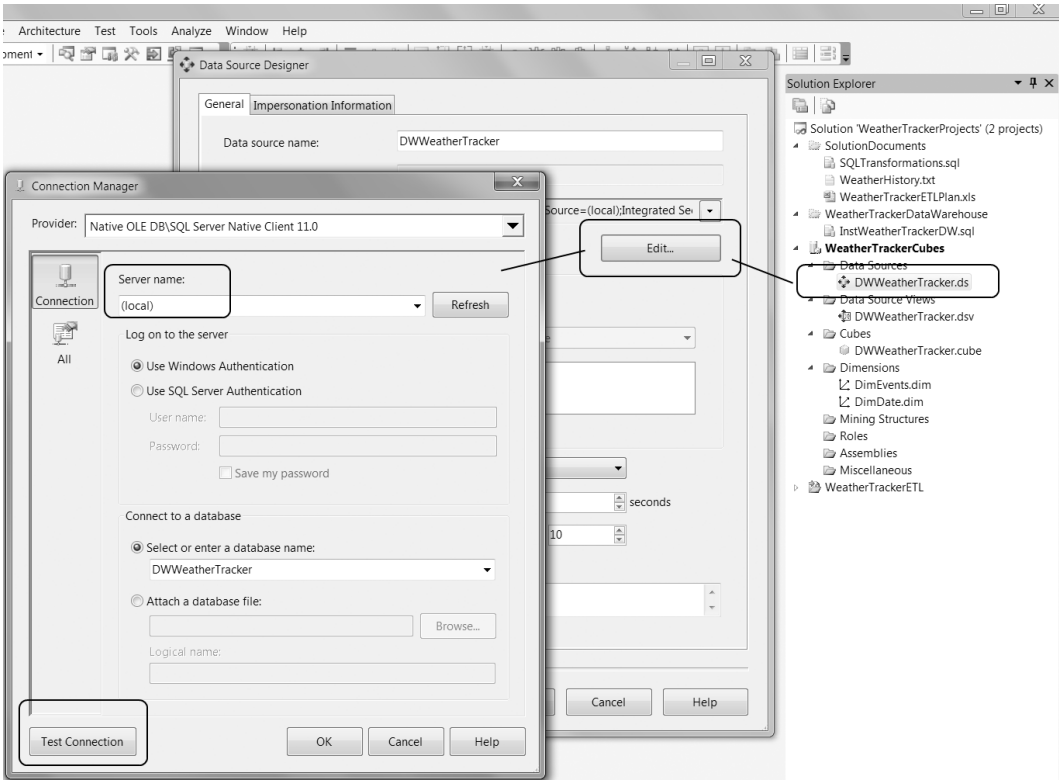


Figure 2-55. *Verifying the connection*

4. Once this dialog window appears, verify that the server name is appropriate for your computer by reviewing the Server Name textbox. If the server name is incorrect, change it to the correct one now.

Important: Remember, if you have installed SQL server on the same computer multiple times, your SQL 2012 installation may be called a name other than (local). For more information, search the Web for "SQL Server Named Instances."

5. Confirm that the DWWeatherTracker database is selected in the "Select or enter database name" dropdown box (Figure 2-55).
6. Click the Test Connection button to verify that your connection works (Figure 2-55).
7. Click OK to close the Connection Manager window and return to the Dialogue Source Designer window (Figure 2-55).
8. Please leave the Dialogue Source Designer dialog window open for now.

Change the Impersonation Settings

Each SSAS project uses an impersonation account to deploy and process its cubes and dimensions. Now configure the connection to use the impersonation information suitable for your computer.

Important: The account you use must have access to both Analysis Server and SQL Server. On a nonproduction machine, we recommend using an administrator account. In addition, a blank password will not work here. If you try to use one, you will find later that the cube deployment will fail.

1. From the Data Source Designer dialog window, select the Impersonation Information tab, as shown in Figure 2-56.
2. Change the username and password (do not leave the password blank) to an account that will work on your computer (Figure 2-56). For example, on Randal's laptop, he uses RSlaptop2\Administrator for the user account, because this account has full admin rights on both SQL Server and Analysis Server. You need to change it to your <computer name>\Administrator.

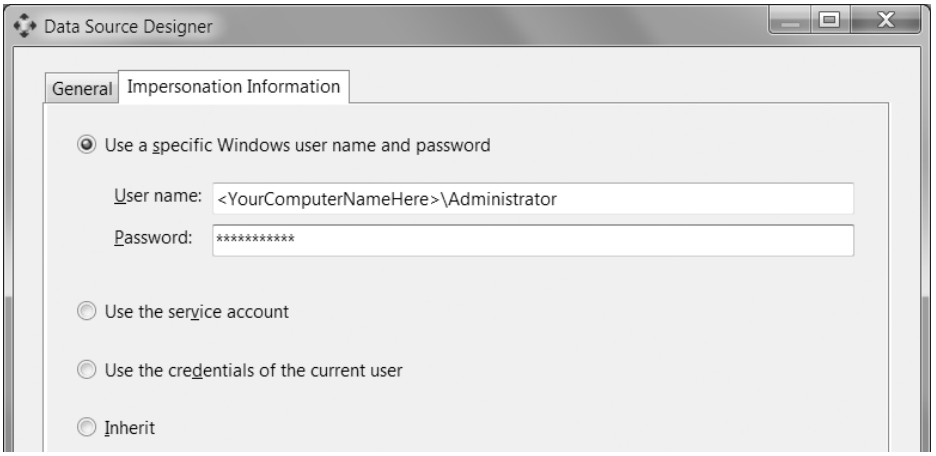


Figure 2-56. *Setting the impersonation name and password*

3. Click OK to close the Data Source Designer dialog window.

Review the Project Files

In the next series of steps, simply review the downloaded project files to help familiarize yourself with how these files are used. In Chapter 9, we discuss what each file does and how to configure the files.

1. Double-click the `WeatherTrackerDW.dsv` file to open a data source view designer tab in the center of your screen. This file can be located under the Data Source Views folder in Solution Explorer.
2. Review the tables displayed in this designer window and note the relationship arrow that connects the two tables. Double-click the arrow (your cursor will turn into a double-ended arrow when you hover over it) to cause the Edit Relationship dialog window to appear.
3. Verify that the EventKey columns are connected via this relationship, and click the OK button to close the dialog window.

4. Expand the cubes folder in Solution Explorer and locate the DWWeatherTracker.cube file within the cube file. Double-click this file to bring up a cube designer window.
5. This window is divided into the Measures and Dimensions pane on the left side of the screen. Click the small expansion + symbol on each of the objects on the left of your screen to expand each treeview.
6. With the treeview expanded, note that the measures and dimensional attributes are displayed. This indicates which measures and dimensional attributes are parts of the cube.
7. In Solution Explorer, expand the Dimensions folder and note that the DimTime dimension and the DimEvents dimension have their own program files.
8. Leave open or close the windows you just reviewed if you want, but do not close Visual Studio yet.

Deploy the Cube and Dimensions

From here, we proceed to building, deploying, and processing the cube and dimensions. Although there should not be any problems at this time, if there are errors, an Error List window will appear. Try your best to read any error message that may occur, resolve the error and build it again until the build succeeds. The most common errors involve an incorrect name or password used for impersonation (Figure 2-56).

1. Locate Visual Studio's Build menu and select the Build WeatherTrackerCubes option (Figure 2-57). Visual Studio will build all the files in this project into a deployment file and present a "Build succeeded" message at the bottom of Visual Studio's window.

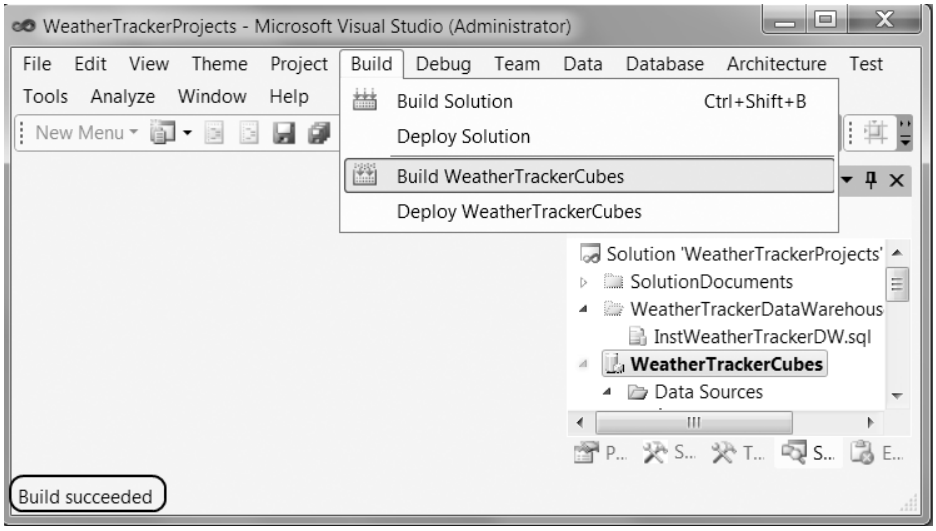


Figure 2-57. *Building the SSAS project*

2. Locate the WeatherTrackerCubes project icon in Solution Explorer. Right-click this icon and select Properties from the context menu. The WeatherTrackerCubes Property Pages dialog window will appear (Figure 2-58).

3. Select the Deployment page under Configuration Properties on the left side of this dialog window, and verify that the Target Server name is correct for your computer. In most cases, the word *localhost*, or (*local*), will work for the server name. If this is not the case on your computer, please change it accordingly (Figure 2-58).

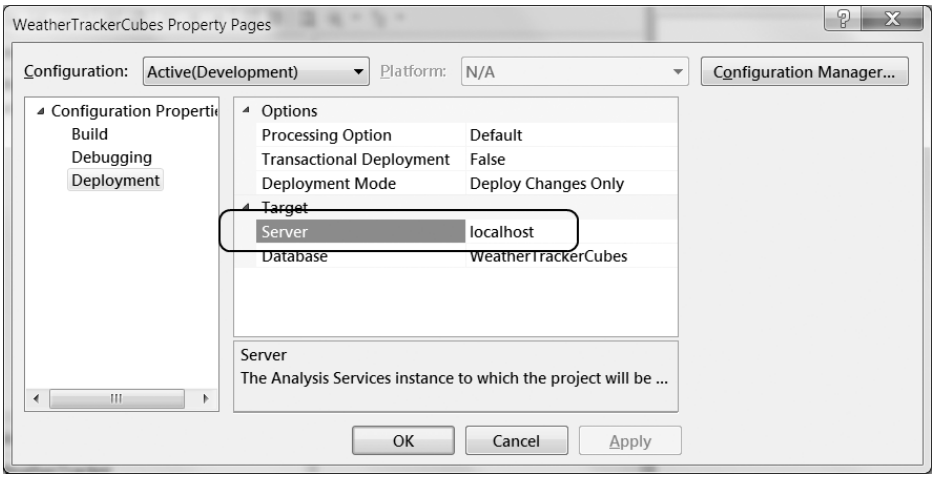


Figure 2-58. *Configuring the server property for deployment*

4. Click OK to close this dialog window.

5. Using Visual Studio's build menu, select the Deploy WeatherTrackerCubes option. This is similar to what you see in Figure 2-57, except you should select the Deploy option rather than the Build option.

6. A dialog window will appear to ask if you would like to build and deploy. Click Yes to continue (Figure 2-59).

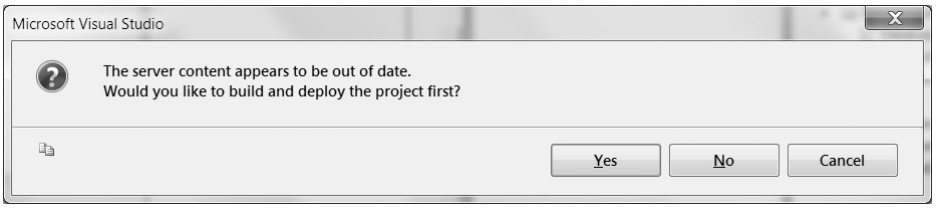


Figure 2-59. *Confirm that you want to build and deploy*

Important: Visual Studio may act differently on your computer and not display the message in Figure 2-59. Not to worry; it still performed the action in the background. In addition, the message may or may not appear whenever you are processing or deploying the cube or dimensions.

7. Visual Studio will connect your analysis server and upload the master XML file that was created during the build process, and a Deployment Progress window will appear. When it has completed deployment, Visual Studio will indicate that it was successful, as shown in Figure 2-60.

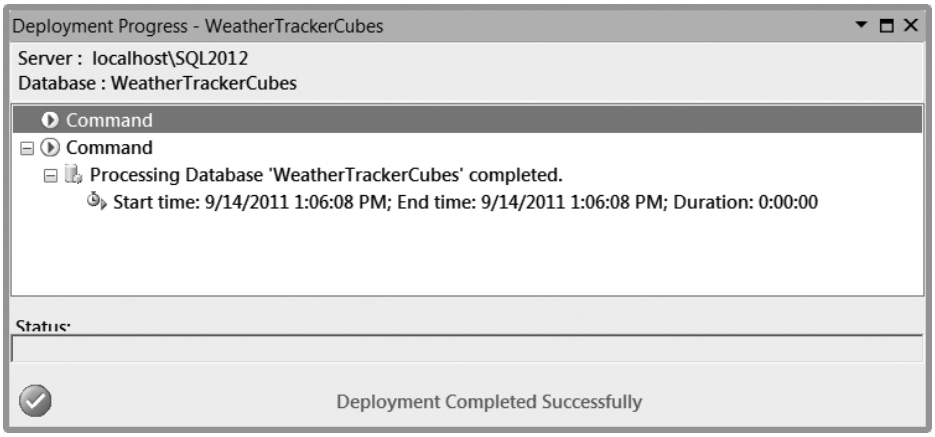


Figure 2-60. *The Deployment Progress window*

8. When the deployment has completed successfully, close the Deployment Progress window.

Process the Database

At this point, SSAS has created the cube and dimensions on the SSAS server, but the data has yet to be copied from the data warehouse into either of these objects. To do that, you must process them. Let's do that next.

1. Right-click the WeatherTrackerCubes project icon in Solution Explorer, and select Process from the context menu. The Process Database dialog window appears (Figure 2-61). (This menu is context sensitive, so make sure you are on the project icon before you access the menu, or it will not say *Process Database*.)

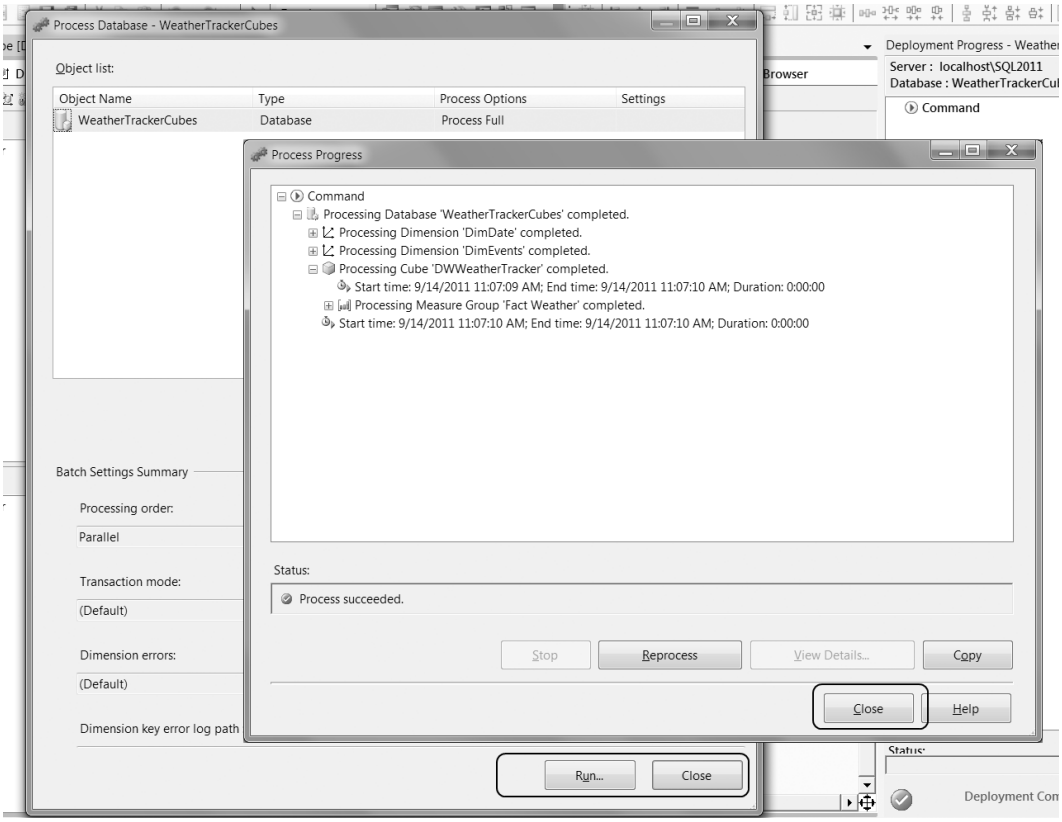


Figure 2-61. *The Process Database and Process Progress windows*

2. Click Run. Visual Studio will tell SSAS to start processing data from the data warehouse into your SSAS objects. Visual Studio will display a Process Progress dialog window.

3. When the processing completes, a status message of Process Succeeded will appear. Click the Close button to exit the Process Progress dialog window, and click Close again to exit the Process Cube dialog window.

In this exercise, you added an Analysis Server project to your existing Visual Studio solution and then built, deployed, and processed the SSAS database. Since the SSAS database now has data, you can create reports based on it. The next step of a BI solution is to create a report and verify that the data is clean, consistent and useful.

Creating Reports

Both the cube and the data warehouse now have data, so let's create some reports. Microsoft has a number of reporting tools that can be used, but we take a look at its server-based reporting tool, SQL Server Reporting Services (SSRS).

SSRS is part of Microsoft's BI application stack and has its own template in Business Intelligence Development Studio. You can create an SSRS project by opening Visual Studio and selecting a project template, just as you did with SSIS and SSAS.

Two templates are associated with the Reporting Server: the first one is the Report Server Project Wizard, and the second is the Report Server Project (Figure 2-62). Both of these templates create report server projects that look identical to each other, but one launches the report creation wizard. We will let you guess which one.

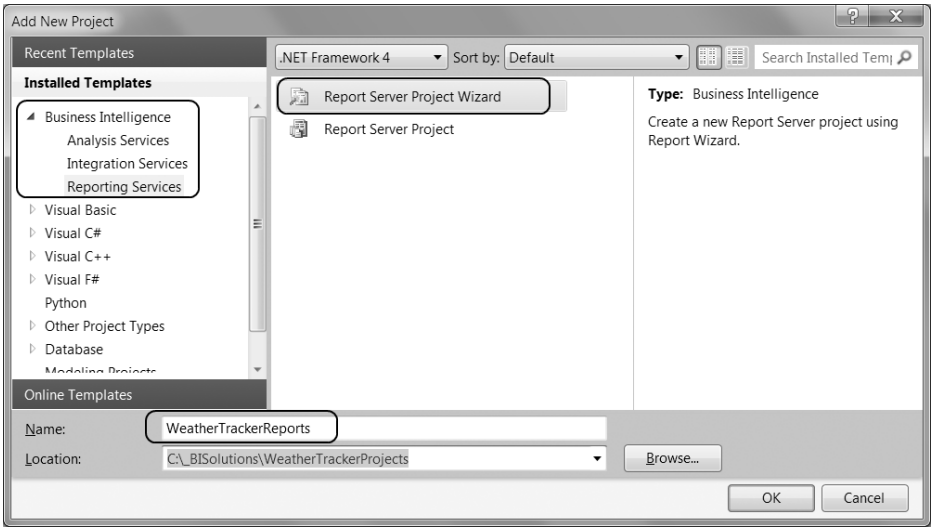


Figure 2-62. *Adding an SSRS project to your solution*

Using the SSRS Wizard

The simplest choice of how to start a new Report Server project is to use the wizard. Once you choose this project template, the Report Wizard launches. The first few pages of the wizard ask you to define the connection to either your data warehouse or your cube. These pages look like Figure 2-63.

Figure 2–63. *The first two pages of the Report Wizard*

After you have created a connection, the next page of the wizard allows you to create a report query. Microsoft added a graphical way to build programming statements to make it easier for developers who are not SQL or MDX programmers. If you choose a relational database for the connection, you see a SQL query builder. However, we chose an SSAS connection, so the wizard will display an MDX query builder. You need to click the Query Builder button before the Query Builder editor displays (Figure 2-64).

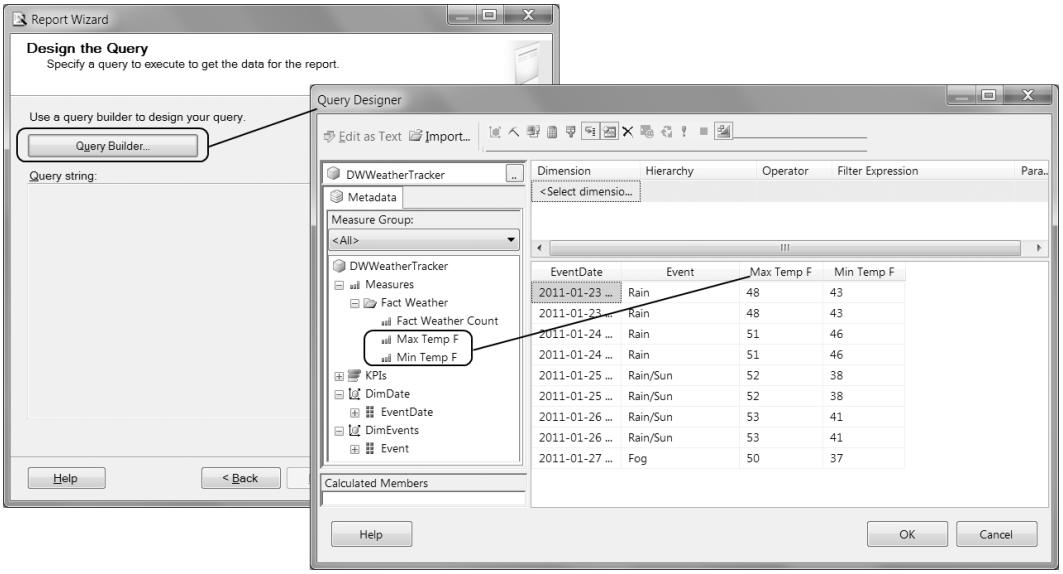


Figure 2–64. *The third page of the Report Wizard*

You can drag and drop elements in the MDX query designer from the treeview on the left side of this dialog window onto the report display section in the center of the dialog window. MDX code is written behind the scenes every time you add or subtract an item from the report display section. When you click the OK button and return to the previous dialog window, the MDX code is displayed.

After you create a report query, the wizard moves you through three pages that define the style of the report. In Figure 2-65, we configure the wizard to create a tabular-stepped report with data grouped by events.

You are able to choose between tabular or matrix formats. The tabular format looks very similar to an Excel spreadsheet, and the matrix format looks similar to an Excel pivot table. We cover more on this subject in Chapters 16 and 17.

Depending on your choice, you are asked to further clarify how you want the data presented in your report. Because we chose a tabular format, the next screen of the wizard (as shown in Figure 2-65) allows us to display data in groups, and the next screen allows a choice between a data-stepped or blocked format.

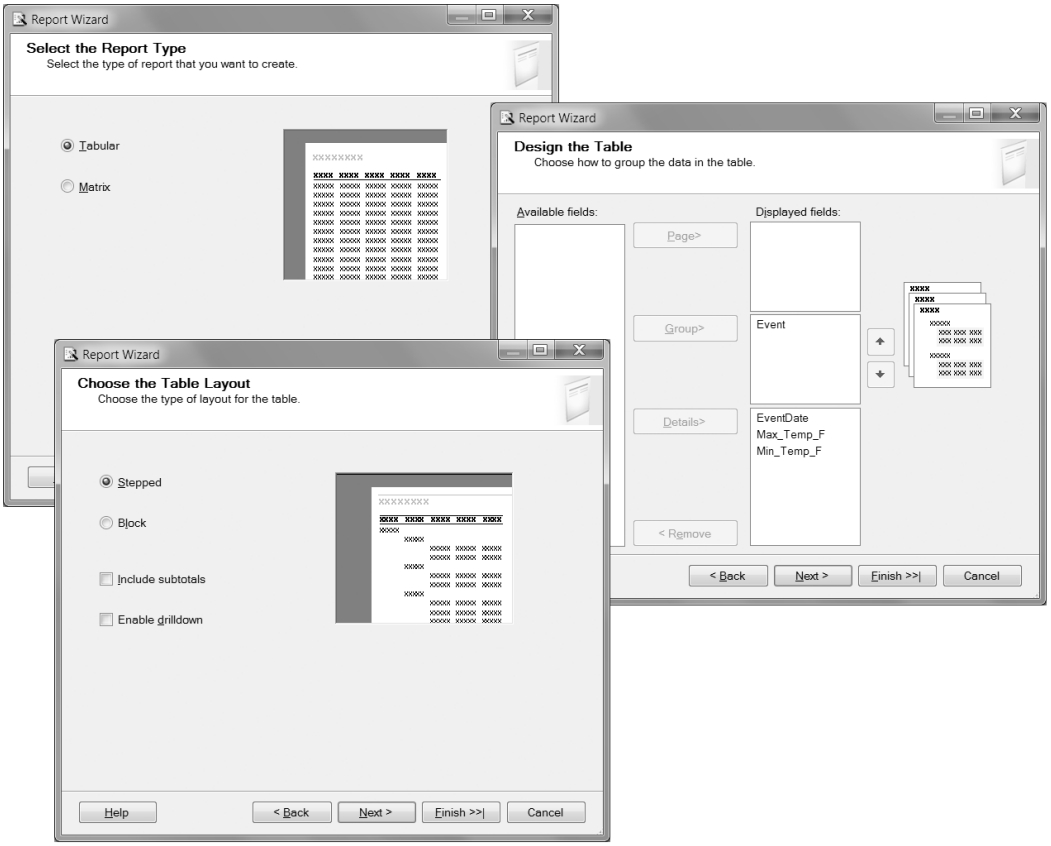


Figure 2-65. *The fourth, fifth, and sixth pages of the Report Wizard*

The last two pages of the wizard allow you to choose some basic colors for the report and summarize the choices you have made throughout the wizard (Figure 2-66).

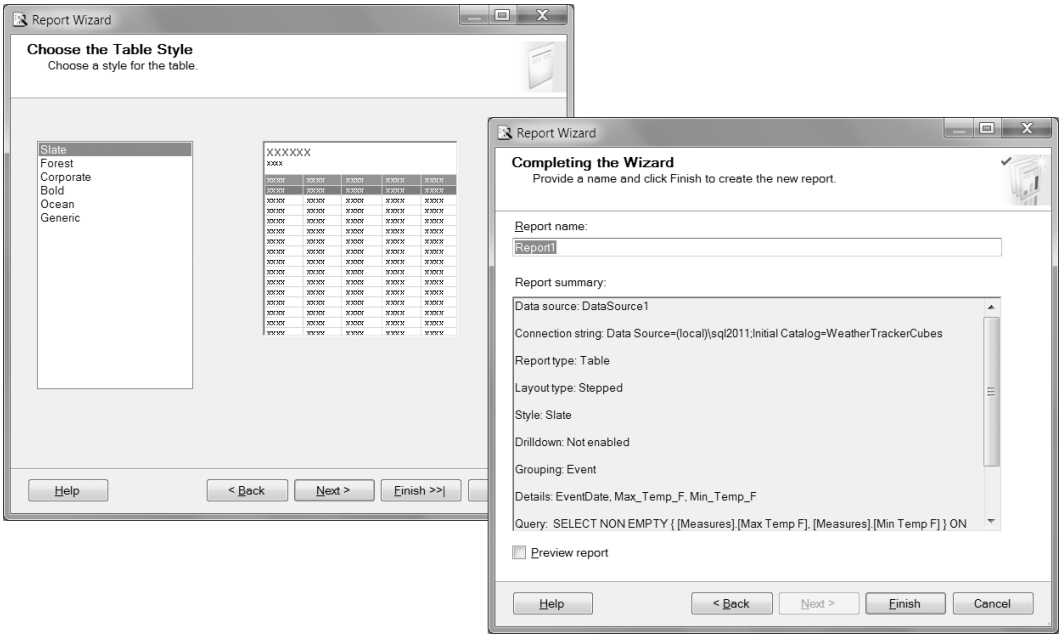


Figure 2-66. *The last two pages of the Report Wizard*

Once the wizard is completed, you end up with a single report page that includes all of the fundamental report data. From there, your task is to continue refining the look and feel of the report. If you would like more reports, this same wizard can easily be launched from Visual Studio by just adding a new report to the project.

The report editor displays in the center of Visual Studio (Figure 2-67). This editor has two options to choose from: Design and Preview.

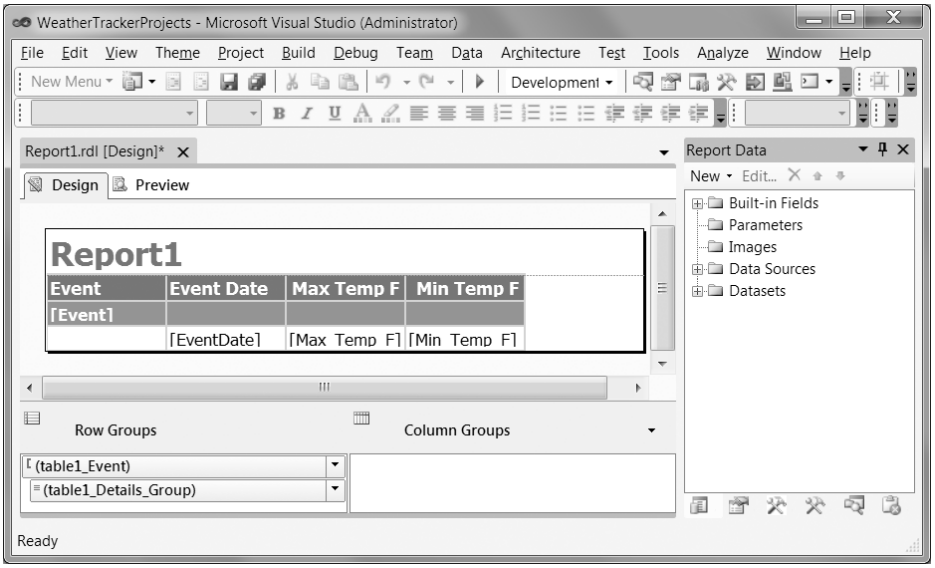


Figure 2-67. *The SSRS Report Wizard is complete.*

When displayed in Design mode, the report can be modified. When displayed in Preview mode, it displays the final colors and appearance that the client can expect to see. You can switch between these two display options to complete the report configuration until you are satisfied that the information you want to share is clearly and professionally presented, as shown in Figure 2-68.

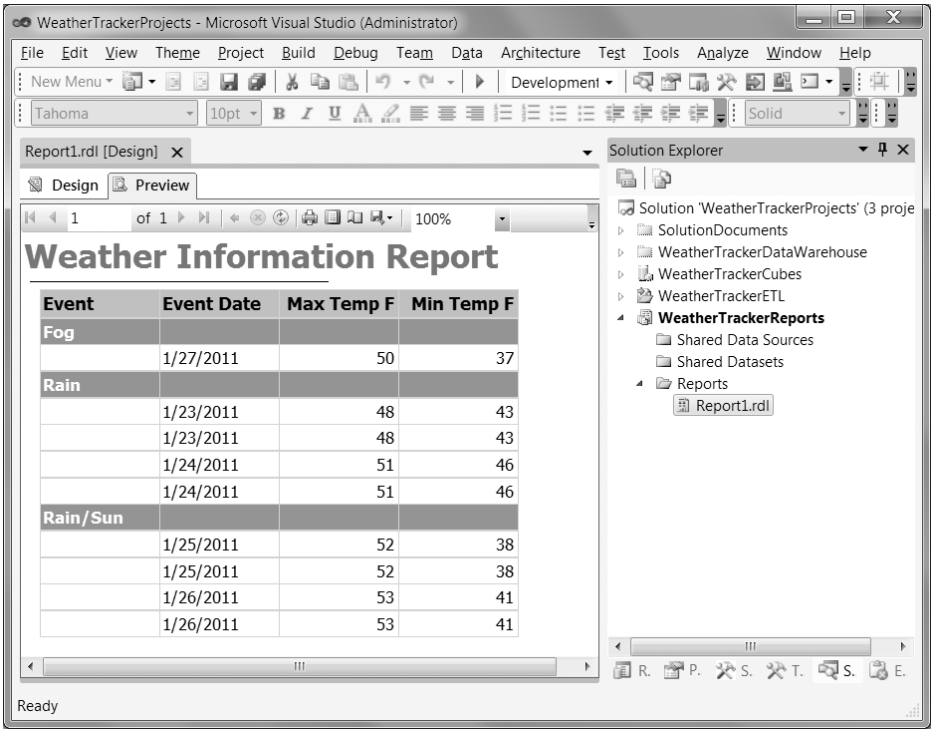


Figure 2-68. *Previewing a report*

Manually Creating SSRS Reports

When creating an SSRS object, you are not as reliant upon a wizard as you are with an SSAS object. You typically create your SSRS reports manually once you become familiar with this method.

To manually create a report, you begin by adding a data source object to the report. This is very similar to how it is done in SSIS and SSAS. Once the data source object has been added, you must add one or more SSRS datasets to the report to define which data you will use. An SSRS dataset consists of an associated data source connection and a programming query, typically in SQL or MDX code. This code is either entered manually or by the same query-building tool that is utilized by the SSRS wizard.

There are a few other ways to create SSRS reports beyond these two options. In addition to developing reports, there is a management aspect that you need to know about. All of this is discussed in Chapter 16. For now, let's do an exercise to create an SSRS report.

EXERCISE 2-5. CREATING A REPORT

In this exercise, you create an SSRS report using the Report Wizard.

1. Visual Studio should still be open from the previous exercise, but if not, please open it and access the WeatherTrackingProject solution from the File ➤ Recent Projects and Solutions menu. (Remember to always run Visual Studio as an administrator by right-clicking the Visual Studio menu item, selecting Run as Administrator, and then answering Yes to close the UAC popup window.

Create a Report with the Report Wizard

1. Using the Visual Studio menu, select the File ➤ *Add* ➤ New Project option. The Add New Project dialog window appears.
2. Select the Business Intelligence option on the left side of this window and the Report Server Project Wizard on the right side (Figure 2-62).
3. Locate the Name textbox and change the name to *WeatherTrackerReports*. Verify that this project will be created in the `C:\_BISolutions\WeatherTrackerProjects` folder (Figure 2-62).
4. Click OK to close the dialog window. You should see the new SSRS project added to your solution in Solution Explorer, and the Report Wizard welcome screen will appear.
5. You can read what is on this page or just click Next to move to the next page (Figure 2-63).
6. The next page of the wizard is the Select the Data Source page. Change the value in the Name textbox to *WeatherTrackerCubes*.
7. Locate the Type dropdown box and use it to select Microsoft's SQL Server Analysis Services connection type (Figure 2-63).
8. Locate the Edit button and click it to bring up the Connection Properties dialog window.
9. When the Connection Properties dialog window appears, type in the name of your SSAS server in the "Server name" textbox. This is usually your computer name, localhost or the word (local) with parentheses; then click OK (Figure 2-63).
10. Click Next to go to the next page of the Report Wizard dialog window (Figure 2-64). This page, called Design the Query, allows you to create SQL, MDX or DMX queries using a query-building tool. Click the Query Builder button. MDX is the language used to query Analysis Server cubes, so clicking this button will bring up an editing tool that creates MDX code, and the Query Designer dialog window will appear (Figure 2-64).
11. On the left side of this dialog window is a representation of the DWWeatherTracker cube. Click the + symbol to expand the Measures, and then click the Fact Weather treeview icons.
12. Locate the Max Temp F icon. Drag this icon to the center of the dialog window. This measure is displayed as a representation of the report data.
13. Locate the Min Temp F icon. Drag this icon to the center of the dialog window. This measure is also displayed as a representation of report data.
14. Locate the Event icon directly under the DimEvents icon. Drag this icon to the center of the dialog window. This dimensional attribute is now displayed along with the measures.
15. Locate the EventDate icon directly under the DimDate icon. Drag this icon to the center of the dialog window. This dimensional attribute is now displayed as well.

16. Click OK to close the Query Designer dialog window.
17. You will see the MDX code that was created when the Query Designer window closes. Click Next to continue to the next page of the Report Wizard.
18. The next page of the wizard, shown in Figure 2-65, allows you to select between two report formats. In this case, you use the Tabular format. So, verify that tabular is selected, and click Next.
19. The next page allows you to group report data together. Click the Event field in the Available Fields window pane on the left side of this dialog window. Then click the Group button to add the event field to the Group window pane (Figure 2-65).
20. Click the date field and then click the Details button to add this field to the Details window pane. Continue adding MinTempF and the MaxTempF fields by selecting each and clicking the Details button so that all three fields are seen in the Details window pane (Figure 2-65). Click Next to continue.
21. On this page, verify that the Stepped radio button is selected, and check the Enable drilldown checkbox. This allows the report user to expand and contract the report items much like the treeview in Solution Explorer. Click Next to continue (Figure 2-65).
22. On this page, you can choose between different table styles. These styles only affect the visual look of the report and not its functionality. Because the look is not that important at this time, choose one that you like and click Next to continue (Figure 2-66).

Important: If this is the first time a report has been created in a Visual Studio solution, the wizard will move to the Choose Deployment Location page. You can accept any default values and click Next to move to the last screen of the wizard. (If you have not set up your Report Server, Chapter 16 walks you through the process. Until then, you will not be able actually to deploy the report, but that is not necessary in this chapter.)

23. On this last page of the wizard, supply a name for your report and review the choices you have made. Change the name to *WeatherEventsReport*, and click finish to create the SSRS report file (Figure 2-66).
24. When the Report Wizard completes, you should see the `WeatherEventReport.rdl` file in Solution Explorer. You will also see the report in a report designer window (Figure 2-67).
25. Click the Preview tab to preview the report that you have made with the SSRS Report Wizard. Your report should look similar to Figure 2-68.

At this point, the report has been created, but it may not look perfect. You can adjust the report by navigating between the design and preview windows, changing the color scheme, column widths, and value formats until you are satisfied with the look and feel of your report. This is covered more extensively in Chapters 16 and 17, so for now we leave the report looking as it is.
In this exercise, you created a report against the SSAS cube you created in Exercise 2-4. In future chapters, we show you how to create more complex reports using Microsoft's Reporting Server and Excel.

Testing the Solution

Testing the solution is much like editing a book. The tester must understand the basic look and feel of the content, verify that the information presented is accurate in a format prescribed by the business requirements and confirm that the end product has not deviated too far from its original specification. Having someone edit your work is always preferable to doing your own editing, because it is quite easy for you to overlook mistakes in your own work.

As a developer, you can help this process with documentation and consistency. These tools play a major part in making any solution easy to review and validate. The Excel spreadsheet in Figure 2-2, for example, can be used to document the data source columns and data destination columns for testing purposes. Using this, the tester can verify that the column names, data types, and transformations outlined in the Excel spreadsheet are the same at the end of the project as they were at the beginning.

Any deviations from this plan can be questioned, and responses to why the deviation occurred can be answered. These answers are recorded in hopes that the changes created during the first iteration can be anticipated in the next step of the BI solution.

We delve deeper into the details of how to test a BI solution in Chapter 18.

Approve, Release, and Prepare

At the end of the BI solution is a formal approval process. During this phase it is important to review whatever changes are found during the testing process and to document an evaluation of the findings. Typically this document is in the form of a Microsoft Word document that describes, in paragraph form, the various aspects of the individual projects that were included in the BI solution.

It does not have to be very long nor does it have to read like a literary novel. It should simply state the facts so that you can plan the next version of the solution using knowledge gleaned from the previous version. Once this is presentable, you can release the solution to the client. You may need to draft a user manual to be released at the same time. At this point, it is a good idea to gather feedback from the users.

Although there will always be another "final" version, eventually you will come to a place where a new final version will not happen nearly as frequently. Therefore, you should always plan for the version of the BI solution you are releasing to be transitory so that it can be improved upon through experience gained and through user's feedback. We discuss this process further in Chapter 18.

Moving On

In this chapter, we discussed how to create a BI solution from start to finish. We started by examining the requirements and identifying the data, and then we moved on to planning the BI solution using simple documentation. Next, we built a data warehouse in SQL Server, filled it with data using Integration Services, created a cube with Analysis Services, and made a report with Reporting Services.

Now we restart at the beginning with an in-depth look at planning your BI solution in the next chapter.

What's Next?

There are many ways to create a BI solution. Learning about other methods will supply you with additional tools to customize the steps we have laid out here to match those in your organization. We recommend the following book as a good place to start: *Delivering Business Intelligence with Microsoft SQL Server 2012* by Brian Larson (McGraw-Hill Osborne).