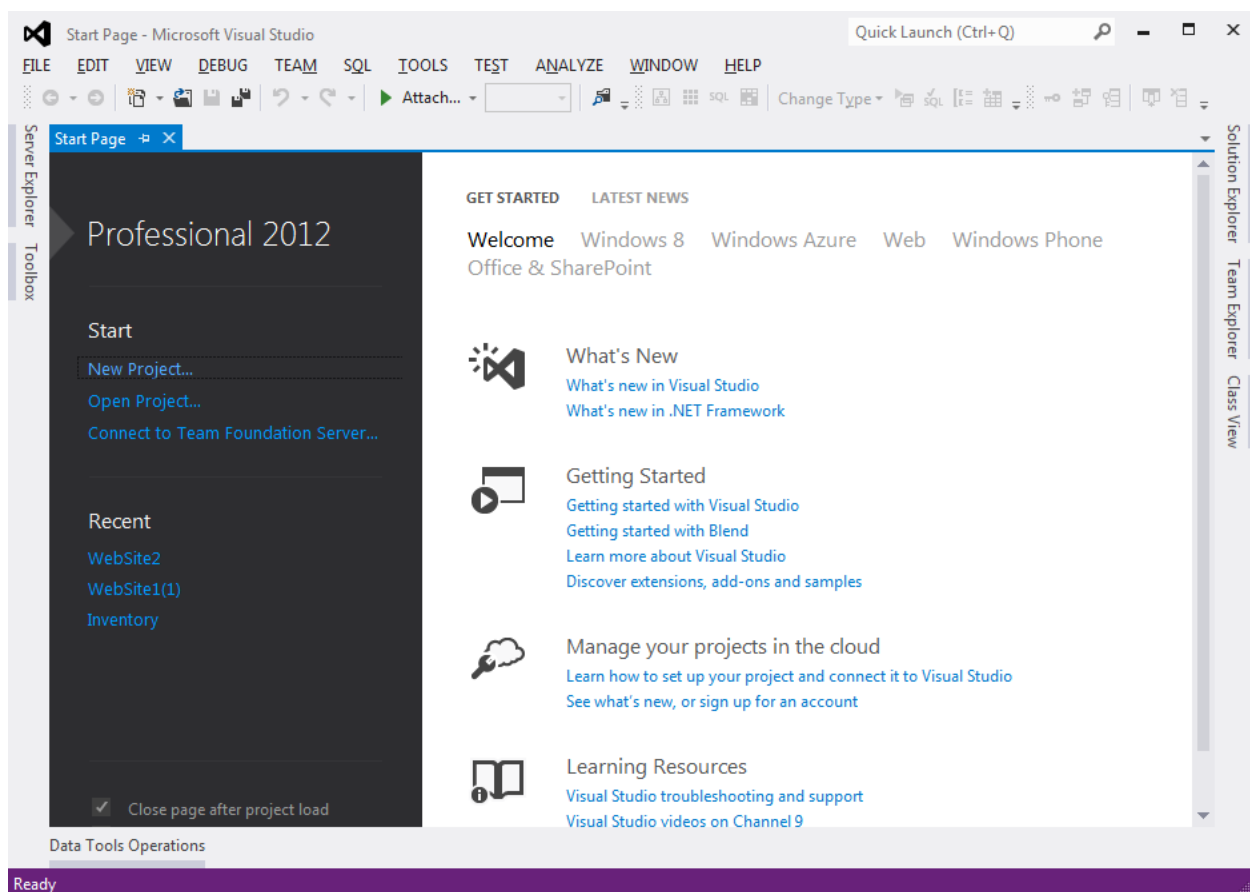


Creating Database Tables in Microsoft SQL Server

Microsoft SQL Server is a relational database server that stores and retrieves data for multi-user network-based applications. SQL Server databases are not designed to stand alone. Rather they generally are accessed through other applications often running on networks, including the Internet. Therefore you often will see SQL Server databases accessed as applications running in web browsers written in languages such as Java, C# (pronounced “C-sharp”), ASP.NET as well as many others.

There are several Microsoft software programs that let database developers create and work with SQL Server databases. The one of these applications that will be used in this course is “Microsoft Visual Studio”, an **integrated development environment (IDE)** for designing applications in Microsoft languages. These languages include Visual C#, Visual Basic and Visual C++.

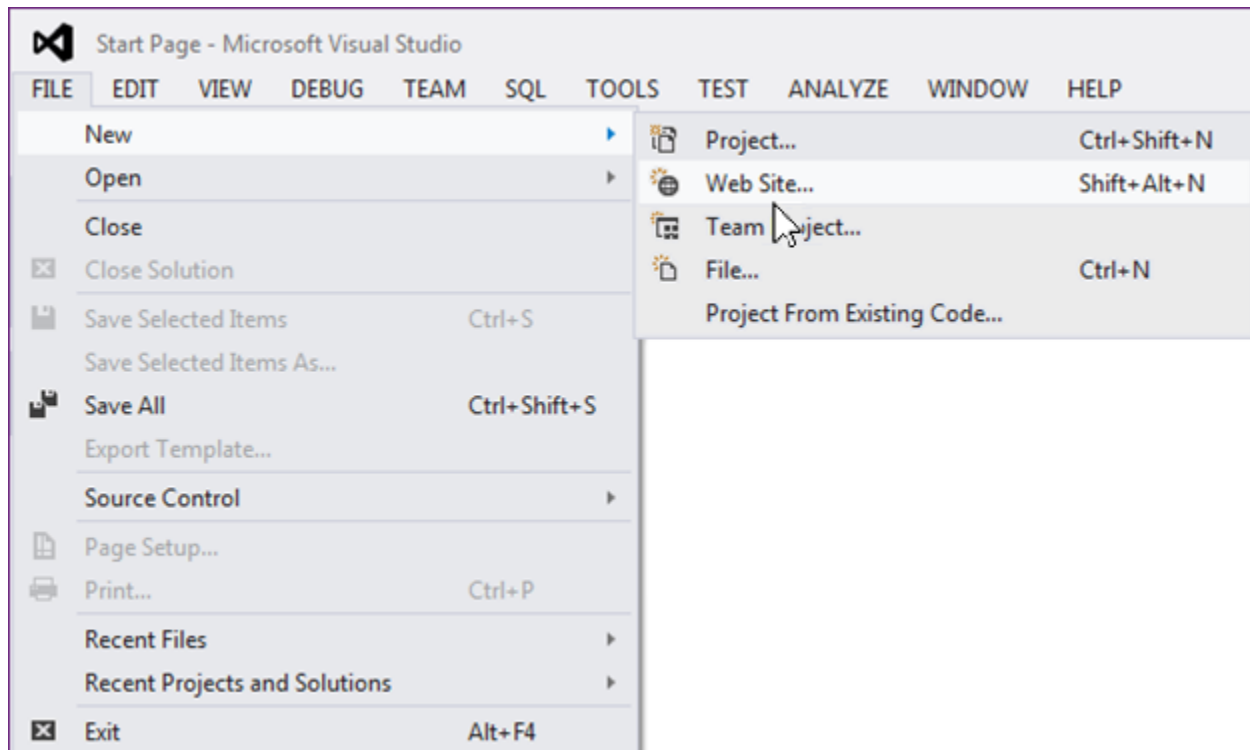
Visual Studio is launched (started) like any other application in Microsoft Windows. A developer begins at the “Start” button selecting “Programs” from the Start menu, or double-clicks an icon located on the Windows **Desktop** (or within a folder on the Desktop) that represents the Visual Studio application. When Visual Studio first starts, the opening IDE window looks like the image below.



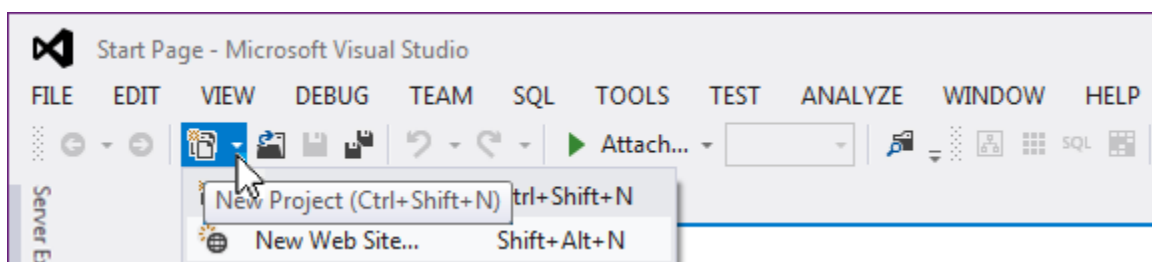
In Visual Studio every new SQL Server database is part of a new project. One type of project is an application that runs as an application in a “window” on the Windows Desktop. Another is a web application designed to run as a website in a browser. For this course we will develop our databases in ASP.NET “Web Site” applications.

We make this choice for two primary reasons. First the videos that accompany this course use a program similar to Visual Studio called Microsoft Visual Web Developer. Essentially this is a version of Visual Studio that only can be used to develop Web Sites. With only minor differences most operations are pretty much the same in Visual Studio as they are in Visual Web Developer. The second reason for developing our databases in Web Sites is that is what we will do in one of the later chapters. Specifically we will create a “Web-Based” ASP.NET SQL Server application to view and manipulate our database within a web browser.

Start by creating a new “ASP.NET Website”. To do this, click the **File** menu, select **New** from the drop-down menu and the **Web Site...** command from the “New” sub-menu:

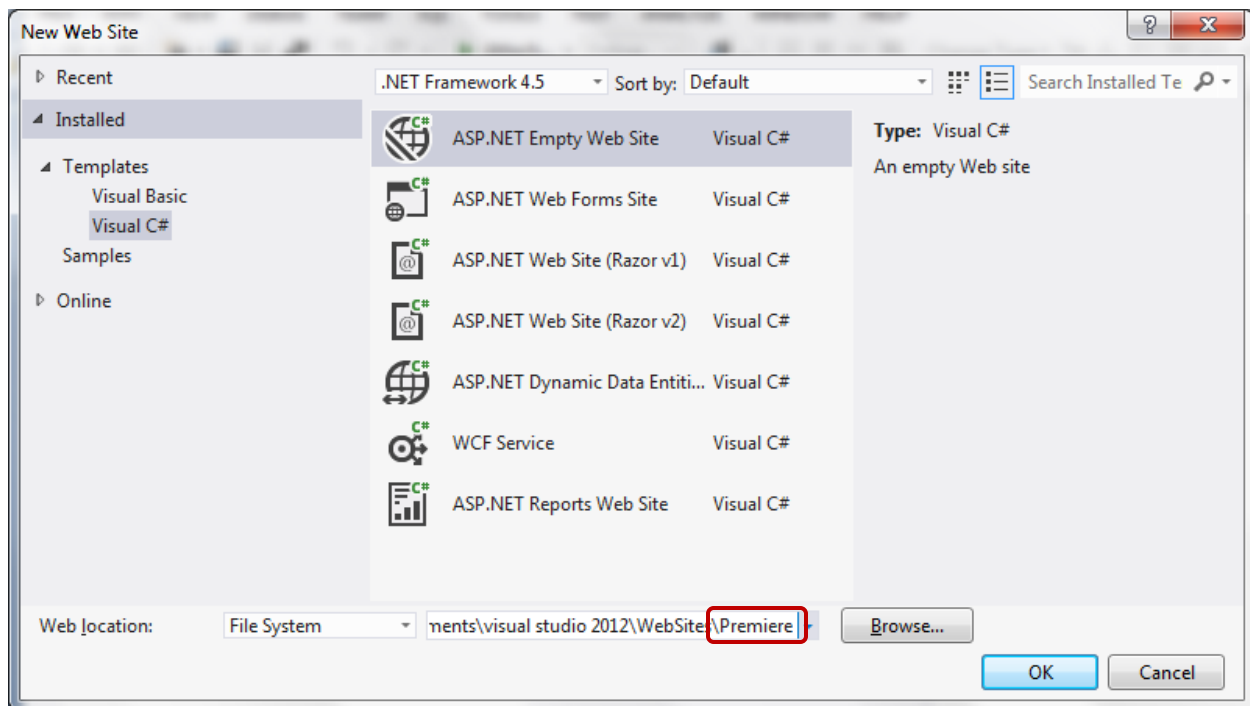


Alternately the developer may click the **New Project** button on the “Standard” toolbar (if the button is available) and select the command “New Web Site...” from the short-cut menu as in the image below.

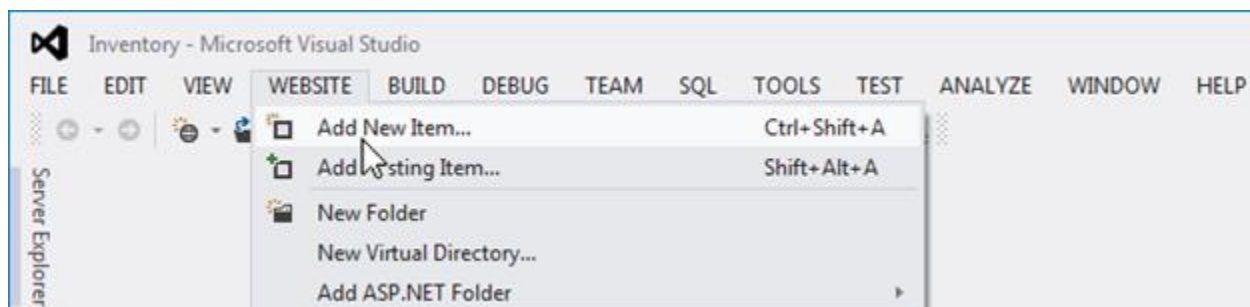


This opens the “New Web Site” dialog window in which the following should be performed:

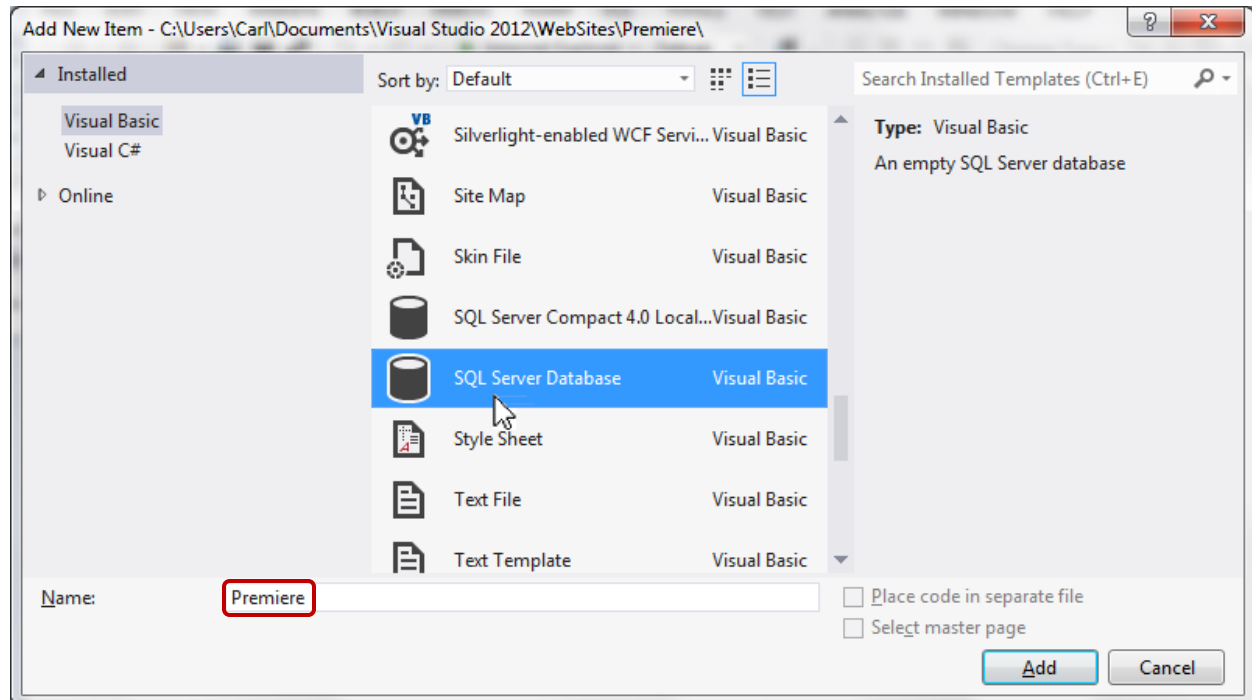
1. Confirm that **Visual C#** is selected from “Installed Templates” (this is the language that we will use in a later chapter when we develop our own “Web-Based” ASP.NET SQL Server application)
2. Click and select the **ASP.NET Empty Web Site** option (it is better to select this option rather than “ASP.NET Web Site” which creates a new website with a whole series of web objects of which you know nothing about at this point in time)
3. Select the **Web location:** by clicking the <Browse...> button and find the “Save To” folder (location); in the example below the location is “C:\Users\User1\My Documents\Visual Studio 2012\WebSites”
4. Enter the name for the website; in the example below the website name is “Premiere” so that the complete entry is “C:\Users\User1\My Documents\Visual Studio 2012\WebSites\Premiere” which should be entered and then click the <OK> button (actually “Premiere” is not a filename but rather the name of the **folder** into which all the files in the Web Site are placed).



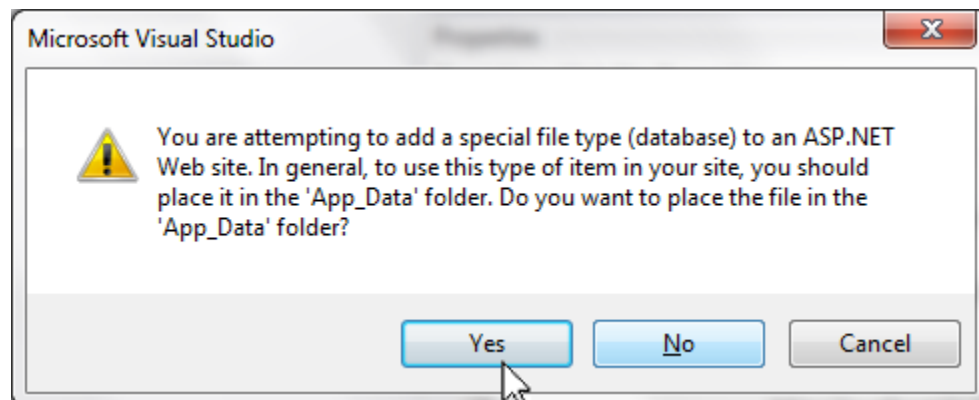
After a few moments the new “empty” Web Site is created. Now the new SQL Server database can be created. To do this click the <Add New Item> button on the “Standard” toolbar.



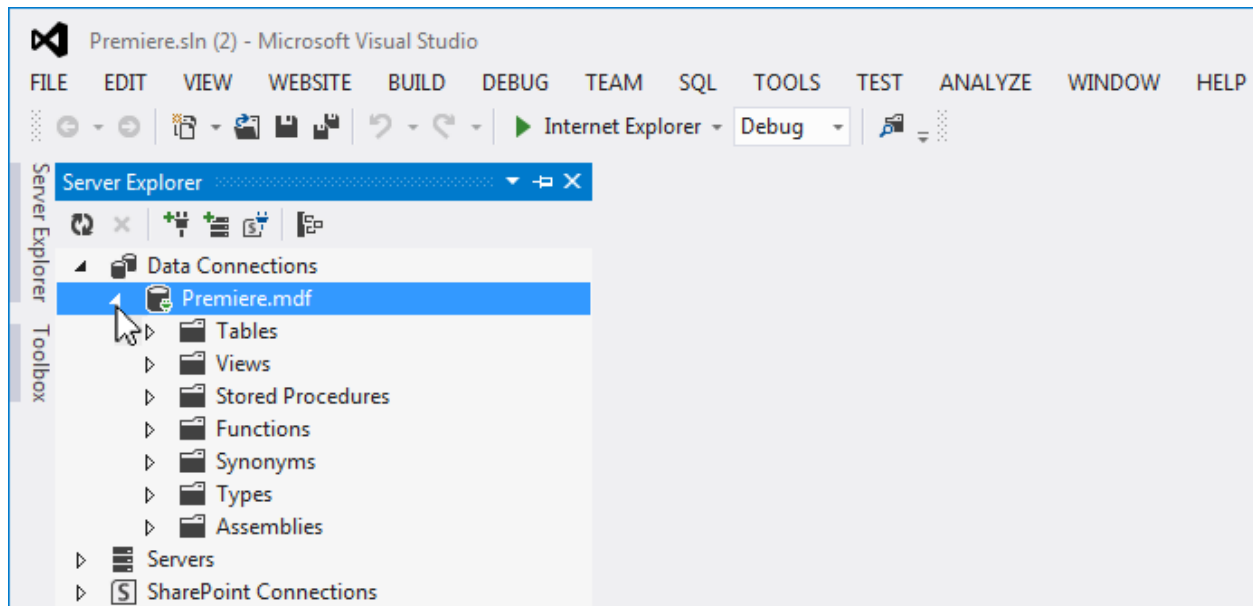
This opens the “Add New Item” dialog window. Scroll down to find and select **SQL Server Database**, give the database a name like “Premiere.mdf” (it is not necessary to include the “.mdf” extension in the filename as SQL Server will add that to the name automatically) in the image below, and click the **<Add>** button.



Visual Studio has a special folder named “App_Data” that often is used for storing SQL Server and other databases in Web Site applications. This is a good idea so the developer should confirm that the database be placed in the “App_Data” folder by clicking the **<Yes>** button.



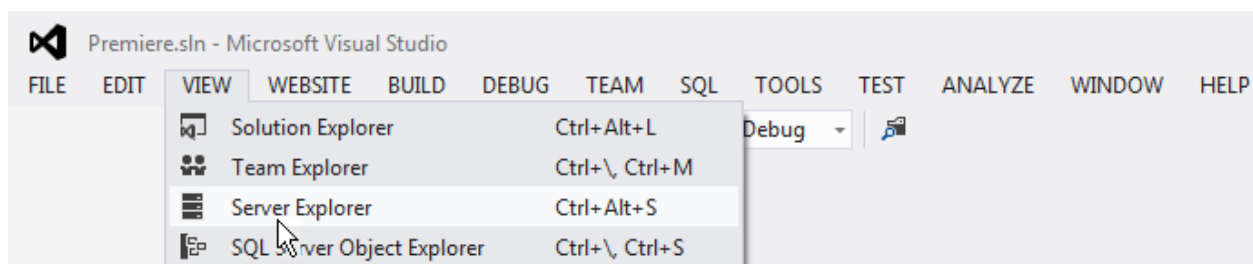
The “Server Explorer” window is the **server management console** for Visual Studio. It is used for viewing open data connections and exploring databases. It displays a categorized list of all the different types of objects that are contained within the databases.



If you do not see the “Server Explorer” window immediately, look for the “Server Explorer” icon and title in the left frame of the Visual Studio window. Click the mouse on this icon and the “Server Explorer” window slides out from the frame and remains there until the mouse pointer click somewhere else.

If you want to keep the “Server Explorer” window visible at all times, click the pushpin (📌) icon (known as **AutoHide**) so it is pointing down. Otherwise if the pushpin is pointing to the left, the “Server Explorer” window slides back into the Visual Studio frame when you move the mouse pointer away from it. Each time you want to work with any of the database objects again, just point the mouse over the “Server Explorer” icon and the window slides back out into view once more. Many database developers prefer to keep Server Explorer hidden providing a larger work area.

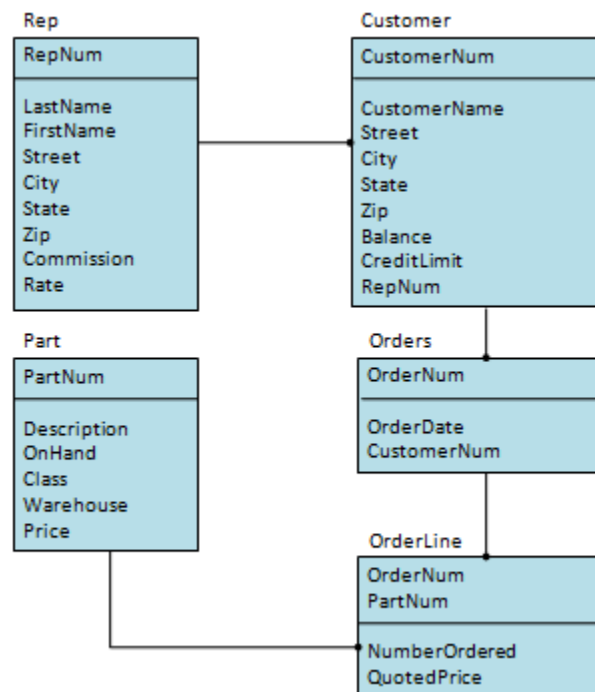
If you still cannot see “Server Explorer” the window may not be open. To see almost any window that currently is not visible, go the “View” menu on the “Standard” toolbar. To open the “Server Explorer” window, click on “View” and select “Server Explorer” from the drop-down menu.



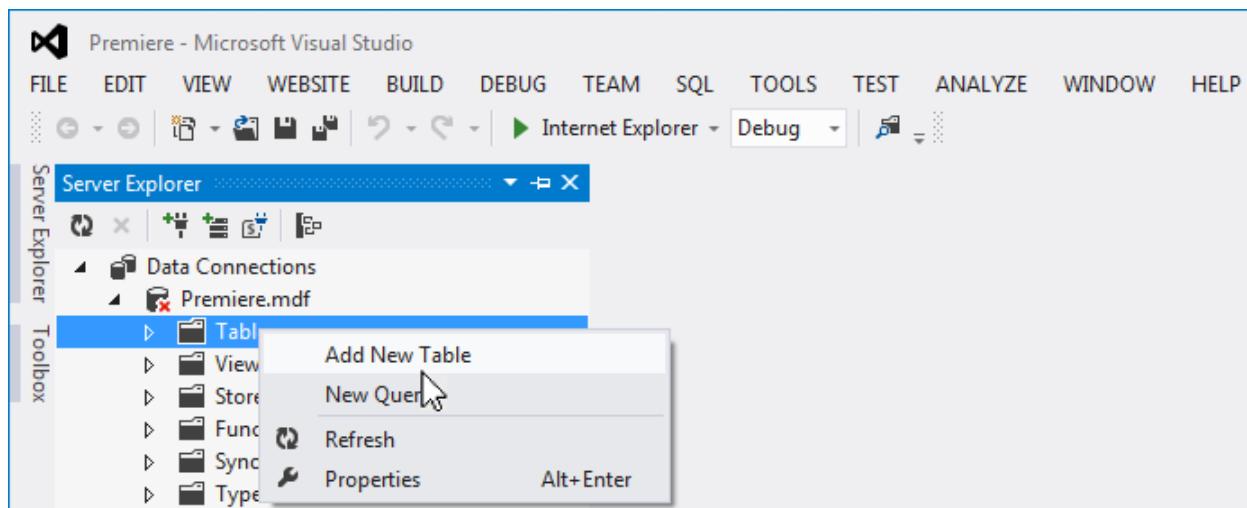
Our specific interest right now is tables and how they are created and then manipulated. Other object types such as **Views** and **Stored Procedures** will be discussed in later chapters. To “drill down” and view lists of the SQL Server database objects (including Tables) click the “expand” [►] arrow in front of the object type. Right now there are no tables or other objects in any of the object categories; if you were to click the expand arrow in front of the “Tables” group, the list of tables would be empty. As you add objects to the database they all will become visible in the “Server Explorer” list.

Creating Tables

A **table** is a collection of **records** (rows) which contain data elements that are related to each other. Consider the “Premiere Products” database which is presented in Chapter 1 of the Pratt and Adamski textbook. It consists of a series of related tables that are represented in the **E-R diagram** shown below. The diagram shows the tables within the database, the fields within those tables, which fields make up the primary key that uniquely identifies records in the table, and the relationships (including one-to-many relationships) between the tables. All of these attributes must be implemented in SQL Server in the physical design of the database.

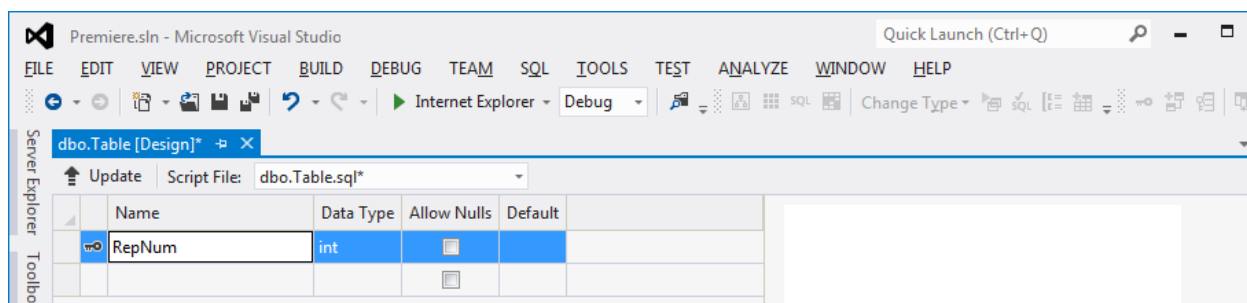


To create a new table in a SQL Server database, right-click “Tables” in the list of SQL Server object types and select “Add New Table” from the “drop-down” **shortcut menu**.



The “Table Design” window appears (see below). To design the structure of a table, three items of information are needed and must be entered for each field’s configuration:

- **Column Name**—the name given to the field, e.g. the **fieldname**
- **Data Type**—a property that specifies what type of data the field can hold (see below); to update this property, click its value which expands the “drop-down list” and select the field’s data type
- **Allow Nulls**—identifies fields that can be left blank when a record is saved; if this checkbox is not “checked” the record cannot be “committed” (saved) to the table until a value is entered for the field (column)



A good habit is to start by **naming** the table. Do this by updating the “CREATE TABLE” statement in the T-SQL window (we will talk more about this window later) at the bottom of the Table Designer by changing the word “Table” to the actual table name, e.g. “Rep” in this instance as in the image below.

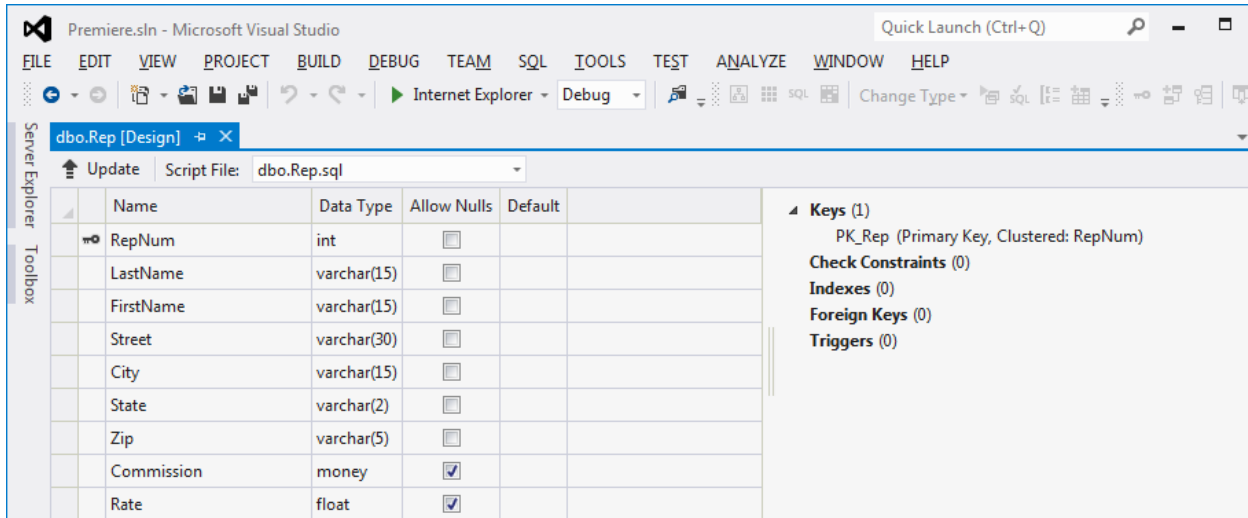


A **data type** is a classification identifying one of various types of data, such as floating-point, integer, Boolean or string, that determines (limits) the possible values that can be stored for that field. Each field in a SQL Server table must have a data type. Some of the SQL Server data types are:

- **varchar**—variable length string for which the **Size** specifies the maximum number of characters that can be stored in each field (there are several other string types—char, nchar, nvarchar, and text—but you always should use varchar)
 - Format: varchar(*n*)
 - *n* is the length of the string between 1 to 4000 characters; this field size value also can be updated in the **Length** property found in the list of “Column Properties” that is displayed at the bottom of the “Table Design” window
- **int**—an integer (a number with no decimals) in the range -2,147,483,648 to 2,147,483,647
- **float**—for use with **floating point** numeric data, that is numbers that have decimals
- **money**—represents monetary or currency (really a floating-point/decimal data type)
- **date**—for entering dates in the range Jan 1, 1 A.D. through December 31, 9999 A.D.; supports entry of dates in a wide variety of formats
- **datetime**—for entering times on a specific date in the range from 12:00 midnight on January 1, 1753 to 11:59:59.997 p.m. on December 31, 9999; supports entry of dates and times in a wide variety of formats including the date without the time and vice versa; the default time for a date entered without the time is 12:00 midnight (displayed as 00:00:00 in universal time)
- **bit**—used to represent Boolean values (only stores values one (1) which is **TRUE** and zero (0) which is **FALSE**)

With this information we are ready to design the tables that make up the “Premiere.mdf” database. The following image shows the definition of each of the “Rep” table which will be added as a **Table** object to the “Premiere.mdf” database. The completed structure of every table includes the fields/columns (“Name”) as well as the “Data Type” and “Allow Nulls” properties of each table is as follows in the image below.

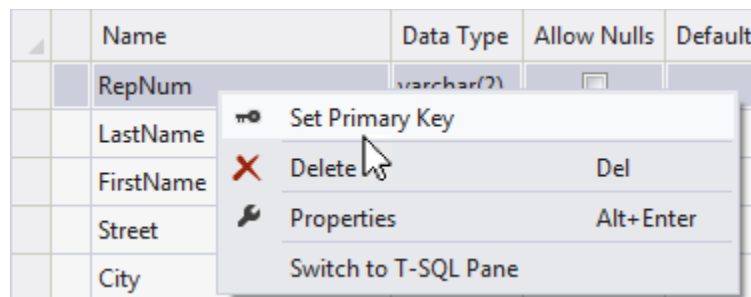
The “Rep” Table



The **primary key** is the field (or in some cases two or more fields joined together) that uniquely identify records in a relational database table. No two records (rows) in the table can have the same value (or combination of values) in those fields (columns). Also the primary key is the mechanism used by SQL Server to determine the default **sort order** for the table when it is displayed. All tables in a relational database should have a primary key.

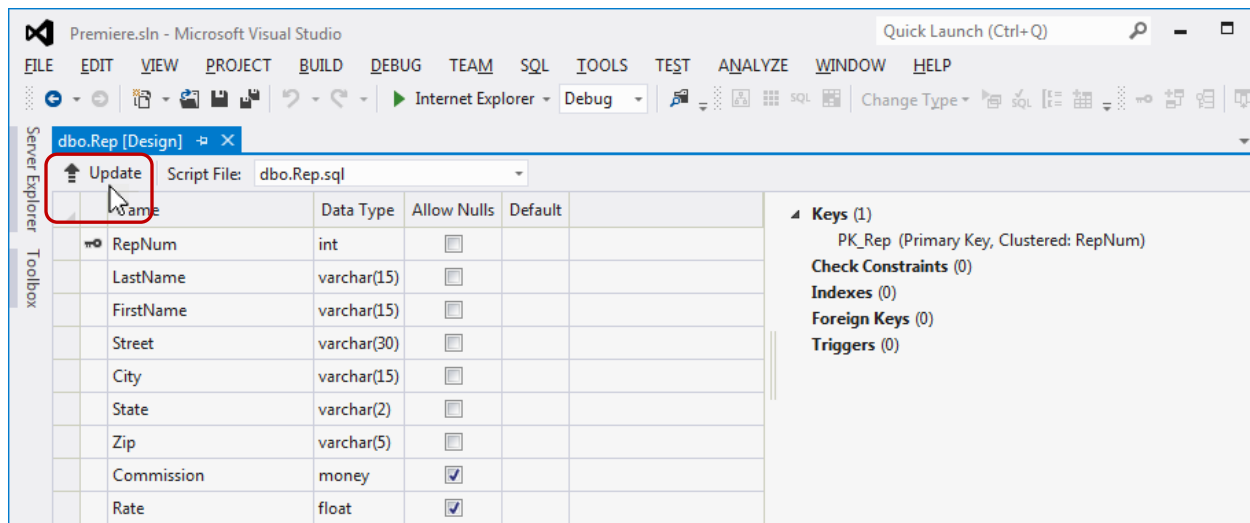
By default every new table has a field named “Id” which is defined initially as the primary key. This column has been renamed “RepNum” in the “Rep” table. Setting this column as the primary key implies that no two “Rep” records will have the same “RepNum” value.

To define the primary key for a table, right-click the column name of the field to be defined as the primary key (the “RepNum” field in the image below) and select the command “Set Primary Key” from the short-cut menu.

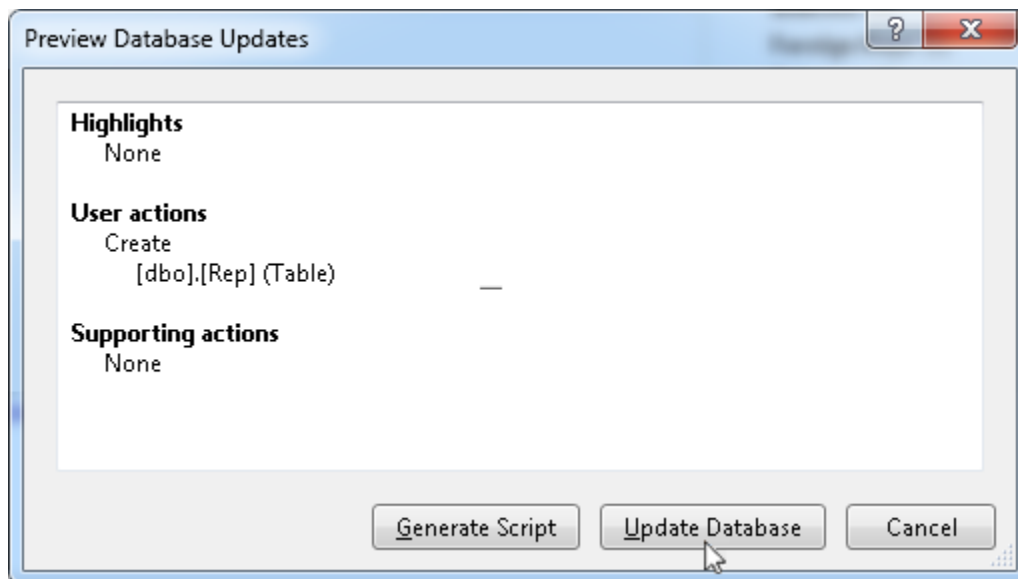


To remove the primary key designation for a field, right-click the column name and select the command “Remove Primary Key” from the short-cut menu.

When the design of a table is complete, the table must be saved. To do this click the “Update” button just above the list of column names. The purpose of this operation is to execute the “CREATE TABLE” command that appears in the “T-SQL” window at the bottom half of the “Table Design” window (more will be discussed concerning T-SQL at a later time).



After a brief time the “Preview Database Updates” dialog window will open. As long as there are no errors in the design of the table, the “User actions” list will indicate the operation(s) that will take place, which in this case is creating the “Rep” table. Click the <Update Database> button to create (save) the table design.



The following images show the additional tables that will be added to the “Premiere Products” database.

The “Customer” Table

Premiere.sln - Microsoft Visual Studio

Quick Launch (Ctrl+Q)

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP

Internet Explorer - Debug

Change Type

dbo.Customer [Design]

Update Script File: dbo.Customer.sql

Name	Data Type	Allow Nulls	Default
CustomerNum	int	<input type="checkbox"/>	
CustomerName	varchar(35)	<input type="checkbox"/>	
Street	varchar(30)	<input type="checkbox"/>	
City	varchar(15)	<input type="checkbox"/>	
State	varchar(2)	<input type="checkbox"/>	
Zip	varchar(5)	<input type="checkbox"/>	
Balance	money	<input checked="" type="checkbox"/>	
CreditLimit	money	<input checked="" type="checkbox"/>	
RepNum	int	<input type="checkbox"/>	

Keys (1)
PK_Customer (Primary Key, Clustered: CustomerNum)
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)

The “Orders” Table

Premiere.sln - Microsoft Visual Studio

Quick Launch (Ctrl+Q)

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP

Internet Explorer - Debug

Change Type

dbo.Orders [Design]

Update Script File: dbo.Orders.sql

Name	Data Type	Allow Nulls	Default
OrderNum	int	<input type="checkbox"/>	
OrderDate	date	<input type="checkbox"/>	
CustomerNum	int	<input type="checkbox"/>	

Keys (1)
PK_Orders (Primary Key, Clustered: OrderNum)
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)

The “OrderLine” Table

Premiere - Microsoft Visual Studio

Quick Launch (Ctrl+Q)

FILE EDIT VIEW PROJECT BUILD DEBUG TEAM SQL TOOLS TEST ANALYZE WINDOW HELP

Internet Explorer - Debug

Change Type

dbo.OrderLine [Design]

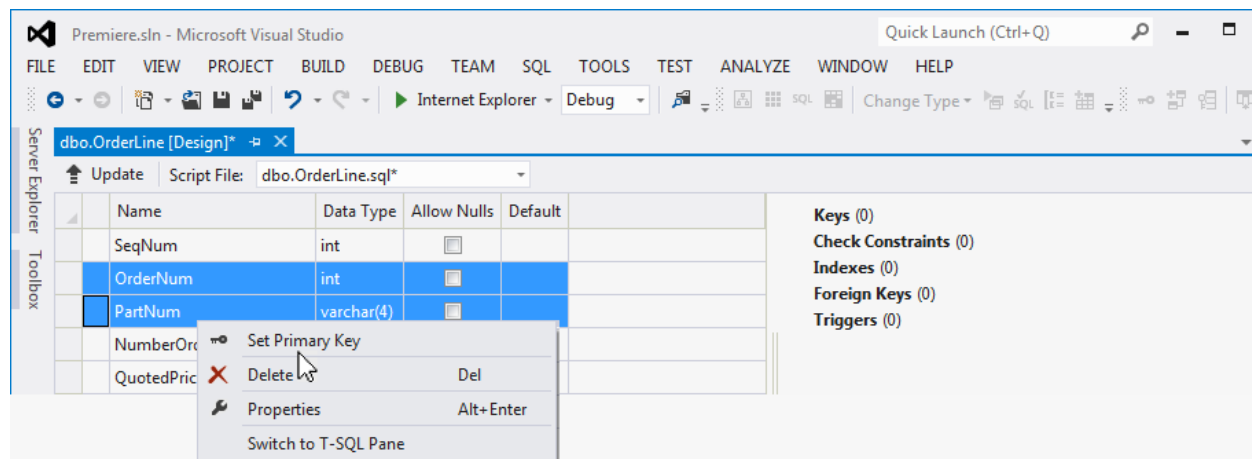
Update Script File: dbo.Table.sql

Name	Data Type	Allow Nulls	Default
SeqNum	int	<input type="checkbox"/>	
OrderNum	int	<input type="checkbox"/>	
PartNum	varchar(4)	<input type="checkbox"/>	
NumberOrdered	int	<input type="checkbox"/>	
QuotedPrice	money	<input type="checkbox"/>	

Keys (1)
PK_OrderLine (Primary Key, Clustered: OrderNum, PartNum)
Check Constraints (0)
Indexes (0)
Foreign Keys (0)
Triggers (0)

As per the E-R diagram on page 6, the “OrderLine” table has a **concatenated** primary key. The reason for this is that neither the “OrderNum” nor “PartNum” columns uniquely identify each record. There could be more than one record in this table with identical order numbers; also there could be more than one record with identical part numbers. However the same part number would appear twice on the same order. Therefore the “OrderNum” nor “PartNum” columns together uniquely identify each record.

To create a concatenated primary key, hold the <Shift> key on the keyboard down while clicking each of the column names that make up the primary key. Then right-click either column name and select the command “Set Primary Key” from the drop-down menu.




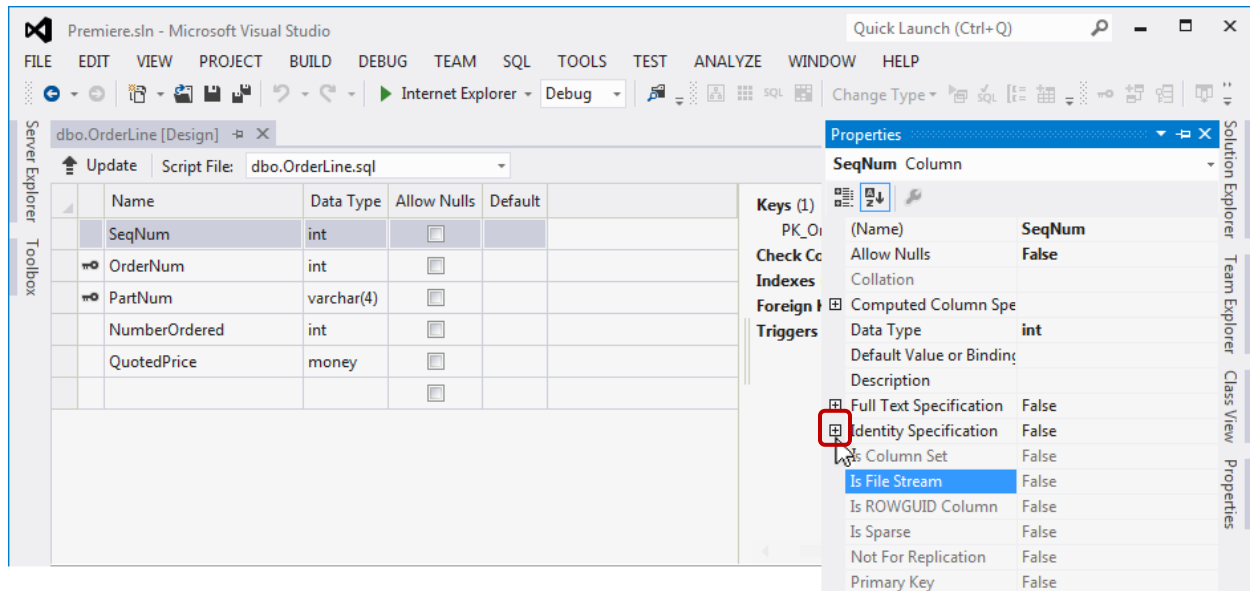
SQL Server lets the developer create integer (**int**) fields that automatically generate numbers **in sequence**, in the form of a “counter” field. This is accomplished by setting values for the **Identity Specification** properties for the field. In the “Premiere.mdf” database from the textbook, no field in any table has its Identity Specification property turned on. Therefore in this lesson an arbitrary “SeqNum” field that does not exist in the textbook (or the E-R diagram on page 6) has been added to the definition of the “OrderLine” table so as to demonstrate this feature.

To turn on the Identity Specification for a field in a table and therefore implement “auto number” generation, the following properties are set:

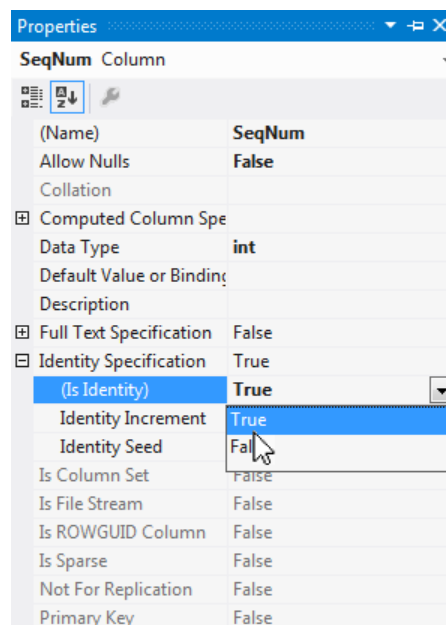
- **(Is Identity)**—set this value to “Yes” to turn “on” the Identity Specification for the field
- **Identity Increment**—the value that will be **added** (+) automatically to the value of each new field when a new record is inserted into the table, frequently a value of one (1)
- **Identity Seed**—value assigned to the Identity Specification field for the **first record** in the table

The following images show how to set the Identity Specification to generate auto numbering field for the “SeqNum” field in the “OrderLine” table. To set the Identity Specification field values:

1. Click the row in Table “Design” window for the field which will be specified as the “Identity Specification”
2. Click “Properties” at the right side of the Visual Studio IDE to open the “Properties Window” and find the “Identity Specification” property in the “Column Properties” list; click the “expand”() icon to drill down to its properties



3. Update the “(Is Identity)” property by clicking the drop-down arrow and selecting “Yes” from the list



4. Set the “Identity Increment” to the amount by which the sequence increments for each new record (default is to count by 1’s)
5. Set the “Identity Seed” to the starting value for the first record (default is to start at 1)

The screenshot shows the 'Properties' window for a column named 'SeqNum'. The 'Identity Specification' is set to 'True'. The 'Identity Increment' is set to '1' and the 'Identity Seed' is set to '1'. Both '1' values are highlighted with a red box.

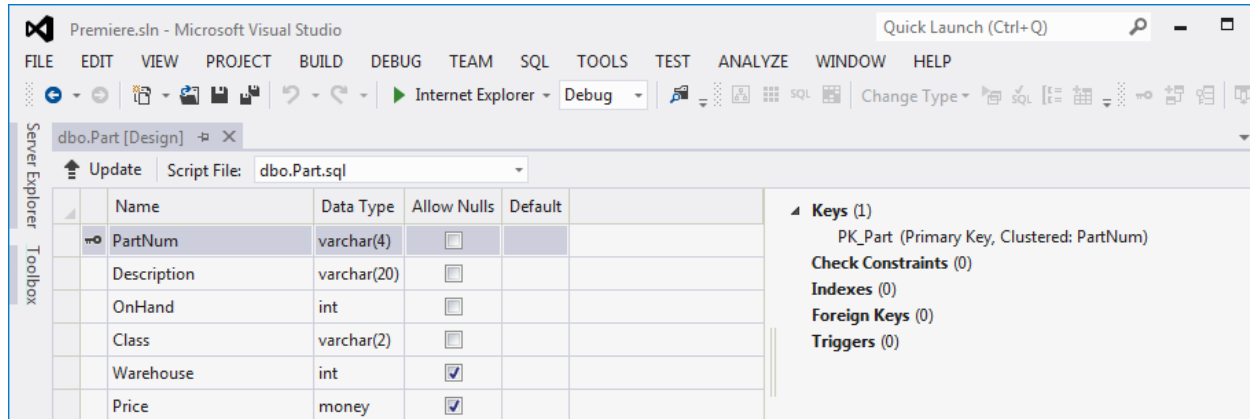
Properties	
SeqNum Column	
(Name)	SeqNum
Allow Nulls	False
Collation	
Computed Column Specification	
Data Type	int
Default Value or Binding	
Description	
Full Text Specification	False
Identity Specification	True
(Is Identity)	True
Identity Increment	1
Identity Seed	1
Is Column Set	False
Is File Stream	False
Is ROWGUID Column	False
Is Sparse	False
Not For Replication	False
Primary Key	False

When new records are inserted into a table with the Identity Specification turned “on” the user **will not be able to manually type** values into the Identity Specification field. Instead when the new record is committed, the new value will be assigned automatically to the field. In the following example, values for the “SeqNum” were calculated automatically by SQL Server and inserted into the field when the user moved the insertion point (cursor) to another new or an existing record.

The screenshot shows a query window titled 'OrderLine: Query...ATA\PREMIERE.MDF'. The table has columns: SeqNum, OrderNum, PartNum, NumOrdered, and QuotedPrice. The SeqNum values are automatically generated by SQL Server, starting from 1 and increasing by 1 for each new record. The last row is a new record with NULL values.

	SeqNum	OrderNum	PartNum	NumOrdered	QuotedPrice
▶	1	21608	AT94	11	21.9500
	2	21610	DR93	1	495.0000
	3	21610	DW11	1	399.9900
	4	21613	KL62	4	329.9500
	5	21614	KT03	2	595.0000
	6	21617	BV06	2	794.9500
	7	21617	CD52	4	150.0000
	8	21619	DR93	1	495.0000
	9	21623	KV29	2	1290.0000
*	NULL	NULL	NULL	NULL	NULL

The “Part” Table



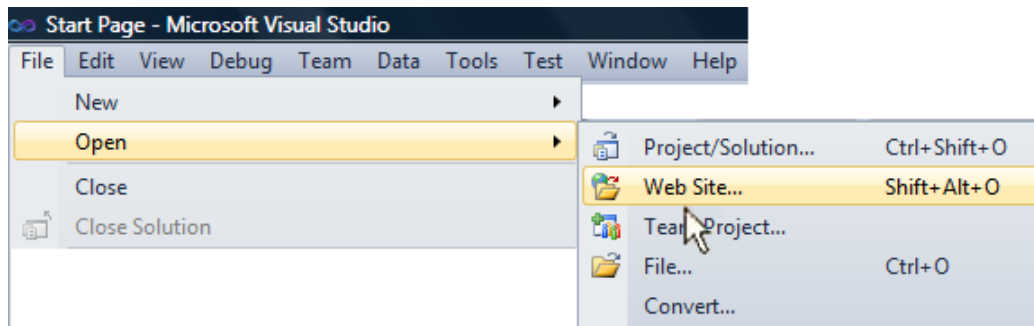
Here again is reminder that whenever changes of any type are made to any table, those changes must be saved to the database design. This includes adding or deleting columns, renaming a column or modifying its data type and/or size and also its “Allow Nulls” property, as well as modifying its “Foreign Keys” properties. This always is completed by clicking the “Update” button just above the list of column names. When the “Preview Database Updates” dialog window opens, click the **<Update Database>** button to create/update the table design.

Working with Existing Databases and Tables

To gain access to a SQL Server database stored within a Visual Studio Web Site, the designer must first launch Visual Studio. Unlike most files and applications in Microsoft Windows, you **cannot double-click the icon** representing the Visual Studio Web Site (actually there is not file for the Web Site, just a folder in which the Web Site is stored) or the SQL Server database.

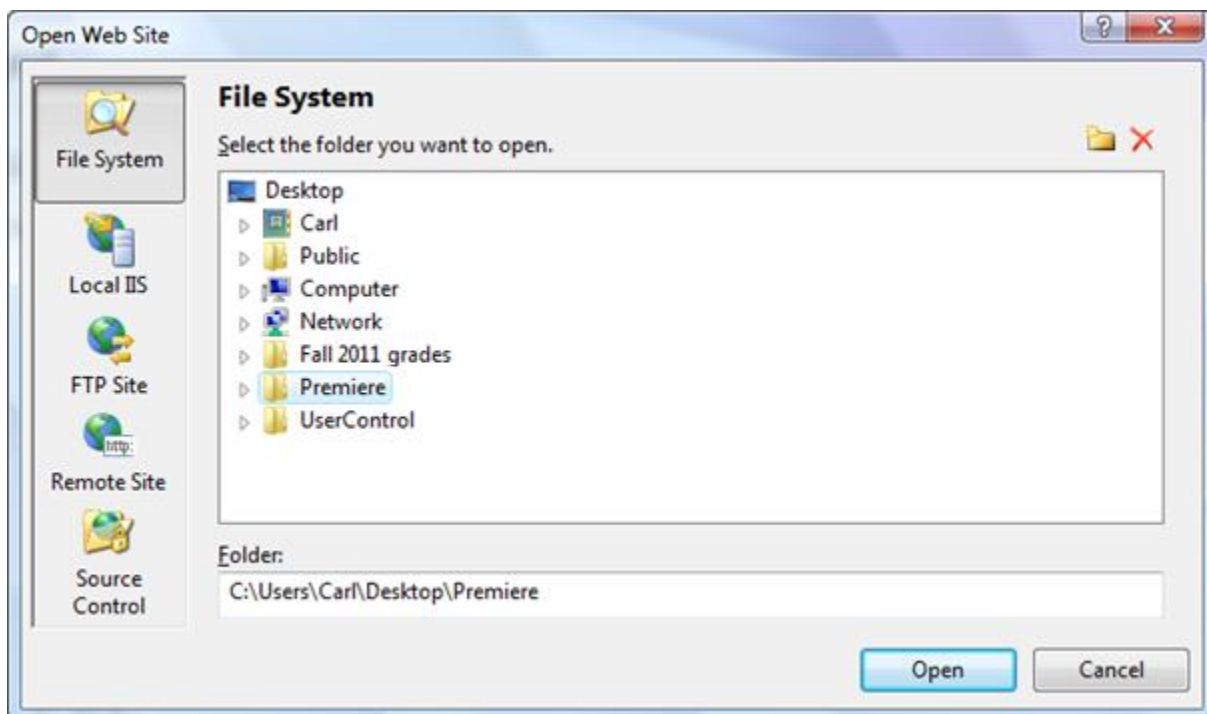
To open an existing Web Site that contains the SQL Server database:

1. From the **File** menu select the **Open** command
2. Then from the “Open” submenu, select **Web Site...**

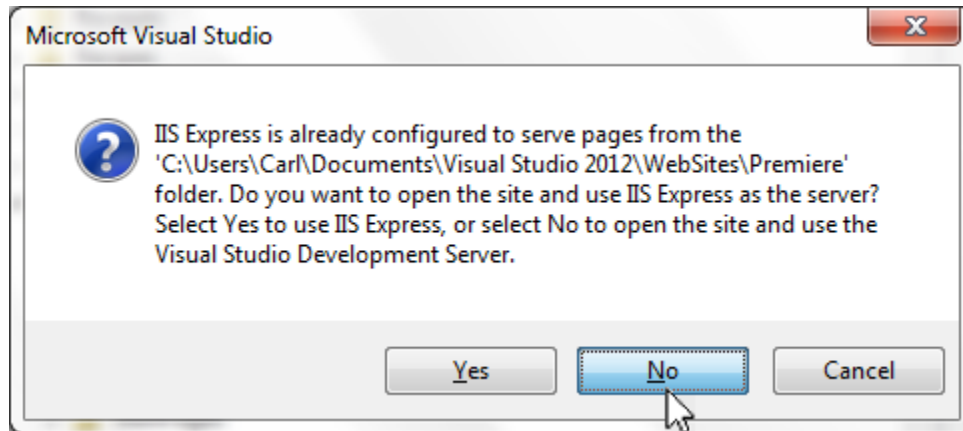


Then in the “Open Web Site” dialog window:

1. Confirm that the <**File System**> button is selected in the left column of the window
2. Find and select (click on) the folder that contains the Web Site
3. Click the <**Open**> button

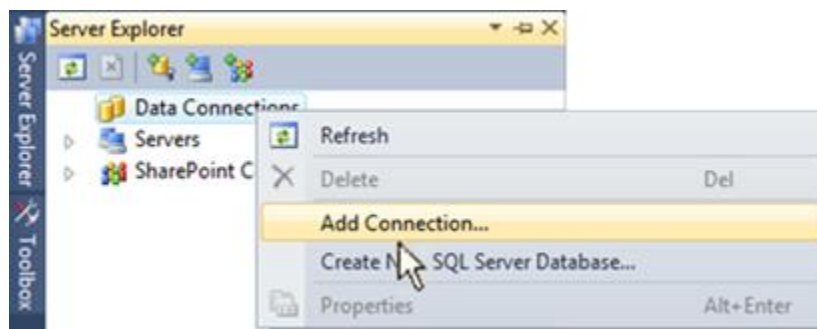


4. Sometimes when a database is being opened, the following dialog window will appear to ask if you want to use something called “IIS Express as the server?” There is no reason for us to discuss IIS Express but rather always be using the Visual Studio Development Server. Always click the <No> button when you see this window.

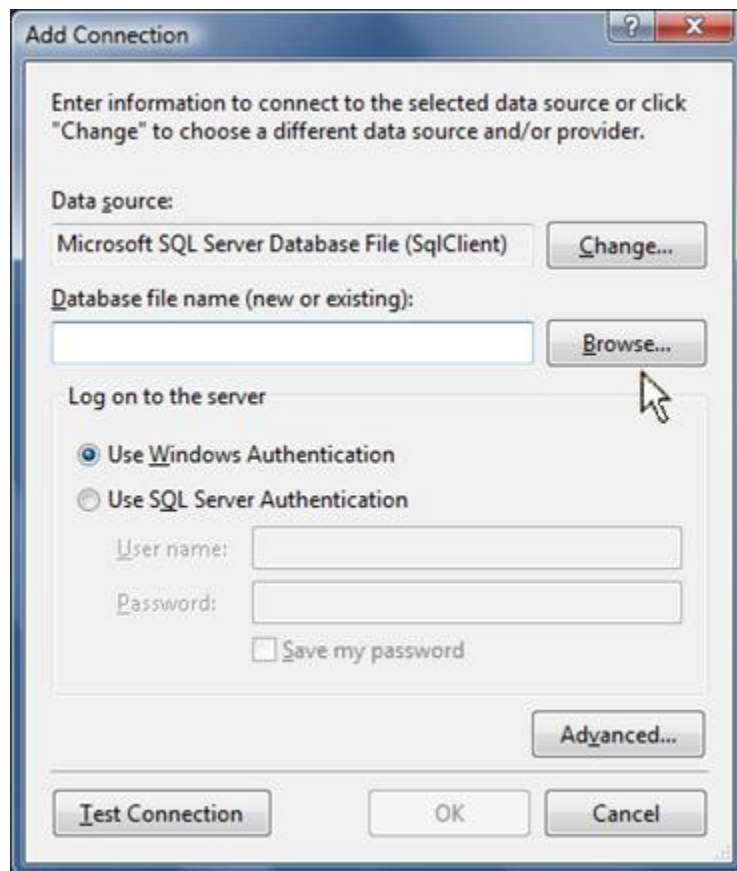


At this point you should have access to “Server Explorer” and the SQL Server database. Occasionally however the database may not be visible in “Server Explorer”. If this happens to you:

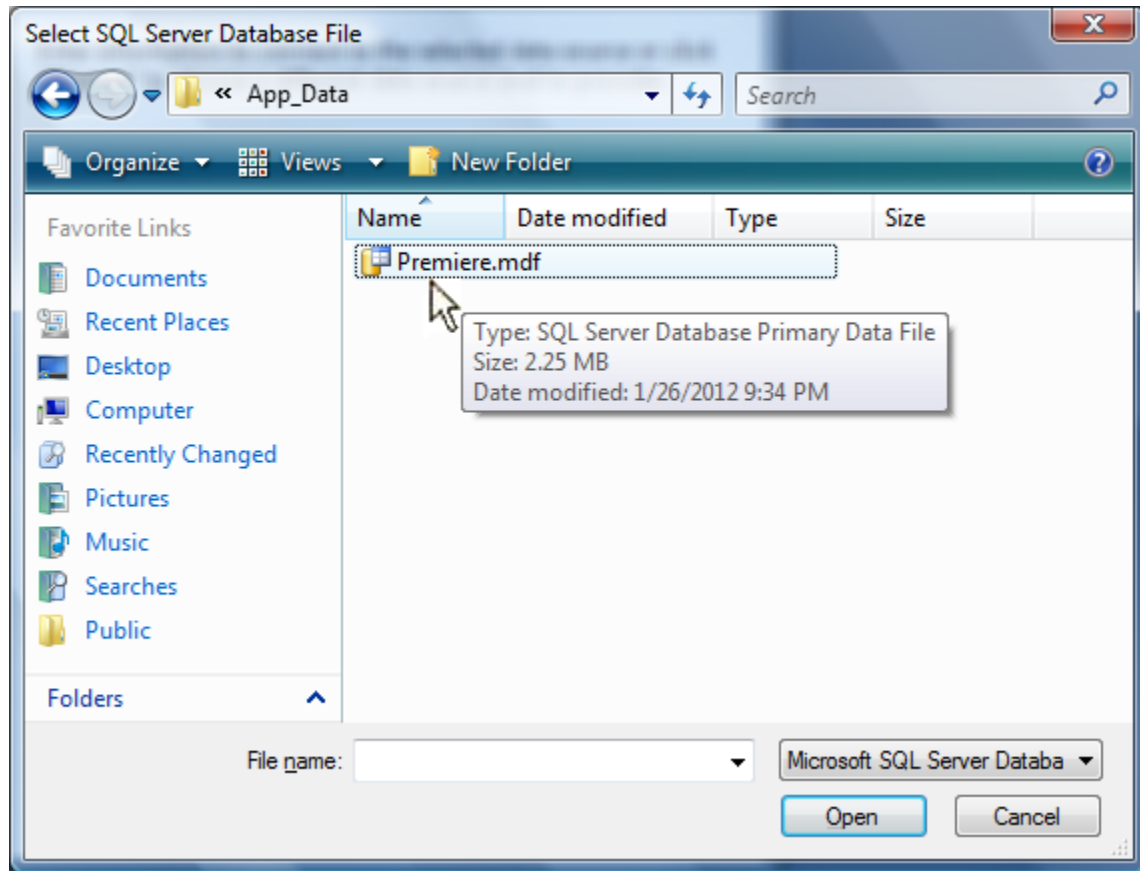
1. In “Server Explorer” right-click on **Data Connections** and select the command **Add Connection...**



2. In the “Add Connection” dialog window select “Microsoft SQL Server Database File (SqlClient)” as the **Data source:** and click the <**Browse...**> button to find the folder that contains the SQL Server database



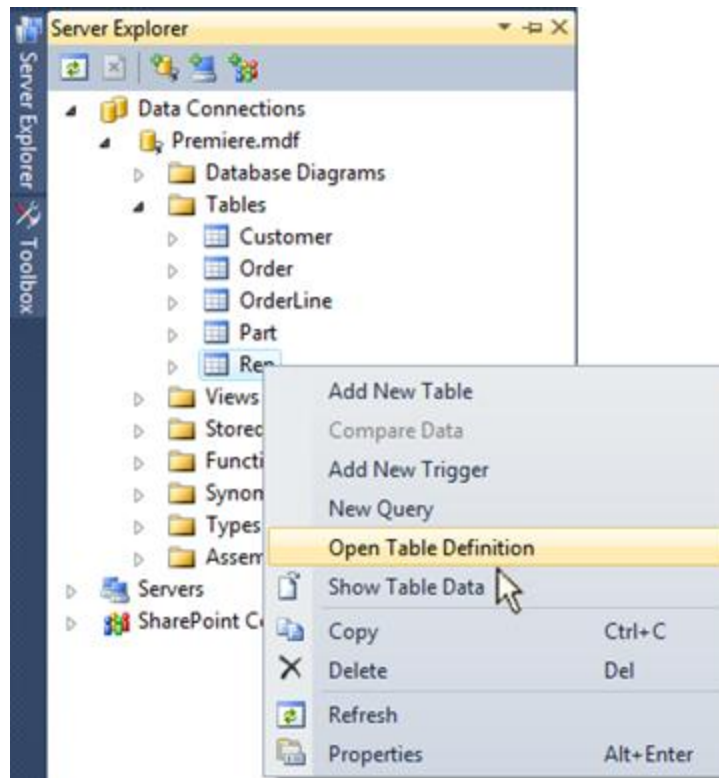
3. In the “Select SQL Server Database File” dialog window, select the database name and click the <Open> button



4. Once you return to the “Add Connection” dialog window click the <OK> button to connect to the SQL Server database

When the database is displayed in the “Server Explorer” window it may be necessary to click the “expand” [>] arrow in front of the database name to view its tables and other objects. Click the arrow in front of “Tables” again to expand down to view the list of table names. It is possible even to click the arrow in front of an individual table name and expand down to view a list of the table’s fieldnames.

To open the definition of any table to continue working on its structure (changing field names and/or data types, setting relationships between tables, etc.), right-click the table name and select “Open Table Definition” from the drop-down menu (see next page). You also can double-click on any table name to open its definition window.



Again here is one final reminder that when any changes are made to existing table structures, you must save redefined tables. Do this by clicking the “Update” button just above the list of column and then clicking the <Update Database> button in the “Preview Database Updates” dialog window to create/update the table design.

Relationships

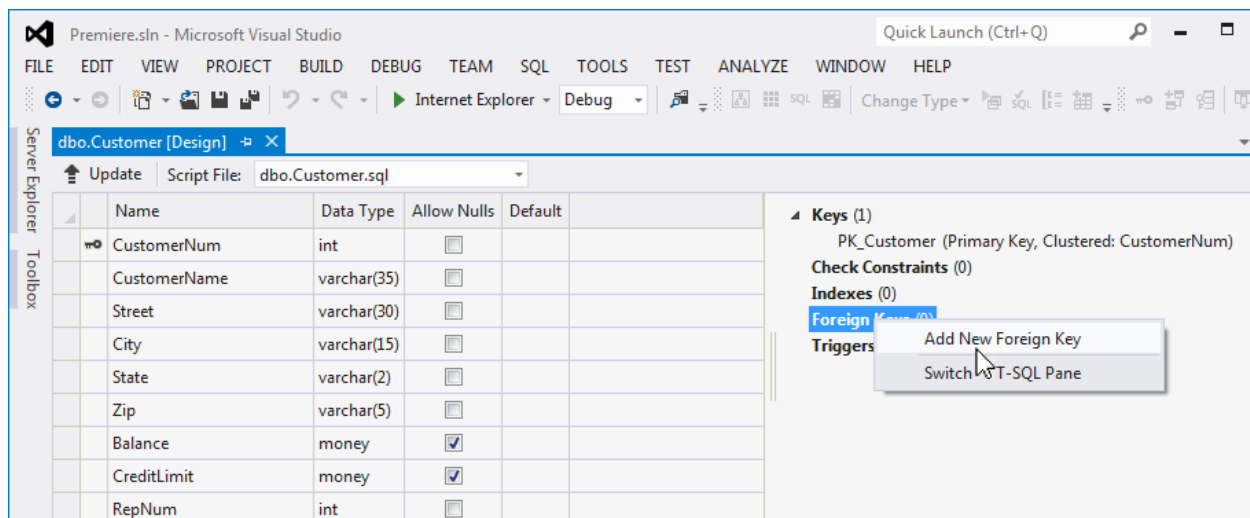
Relationships are the association between entities within the multiple tables which must be enforced in the database. These are the links that are specified by lines drawn between two tables in an E-R diagram. In a **one-to-many relationship** between two tables, one entity in one table can be related to multiple entities in another table.

Consider for example the “Rep” and “Customer” tables in the “Premiere.mdf” database. Since it is the primary key, no two records may have a duplicate value for the “RepNum” field in the “Rep” table (the **one**-part of the relationship). However in the “Customer” table there may be multiple (the **many**-part of the relationship) identical values for the “RepNum” (e.g. multiple customers may be served by the same sales representative). Additionally all values for the “RepNum” field in the “Customer” table must match one of the “RepNum” field values in the “Rep” table.

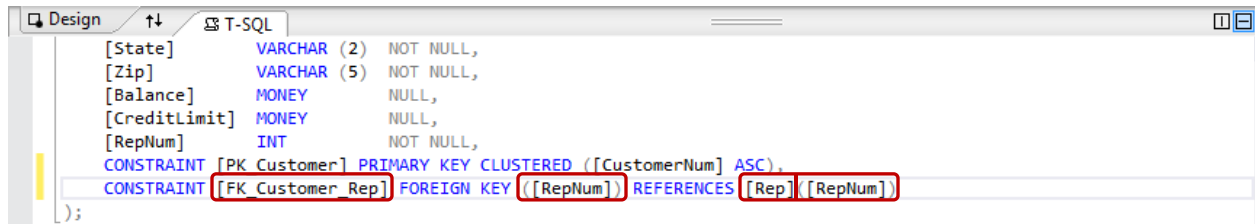
To enforce these relationships, the designer always should enter the “Relationship” function from the table that contains the **many**-part of the relationship (e.g. which is called the “foreign key” table and which is the opposite of the “primary key” table). In the “Premiere.mdf” database example this is the “Customer” table, not the “Rep” table. Additionally the data type and field size of the columns from both tables in the relationship must be the same, e.g. if one field is data type **int**, the other field must be also.

To create and enforce relationships (always create the relationships between tables **before** entering any data):

1. Open the design of the “foreign key” table, the one that contains the many part of the relationship
2. Right-click on “Foreign Keys” in the list of table objects on the right and select “Add New Foreign Key” from the short-cut menu



3. Then scroll down in the “T-SQL” window at the bottom of the Table Designer to find the newly inserted FOREIGN KEY constraint statement. Here make the following changes:
 - Modify the CONSTRAINT name to reflect the tables that make up the relationship; in this case the name is “FK_Customer_Rep” which indicates that “Customer” is the foreign key (the many part of the relationship) and “Rep” is the referenced table (the one part of the relationship). Although this step is not required, it is a good idea for “documenting” the foreign key constraint
 - Modify the FOREIGN KEY value which in this case is “RepNum” of the “Customer” table
 - Modify the table name for REFERENCES which in this case is the “Rep” table
 - Modify the primary key for the REFERENCES which in this case is “RepNum”, the primary key in the “Rep” table

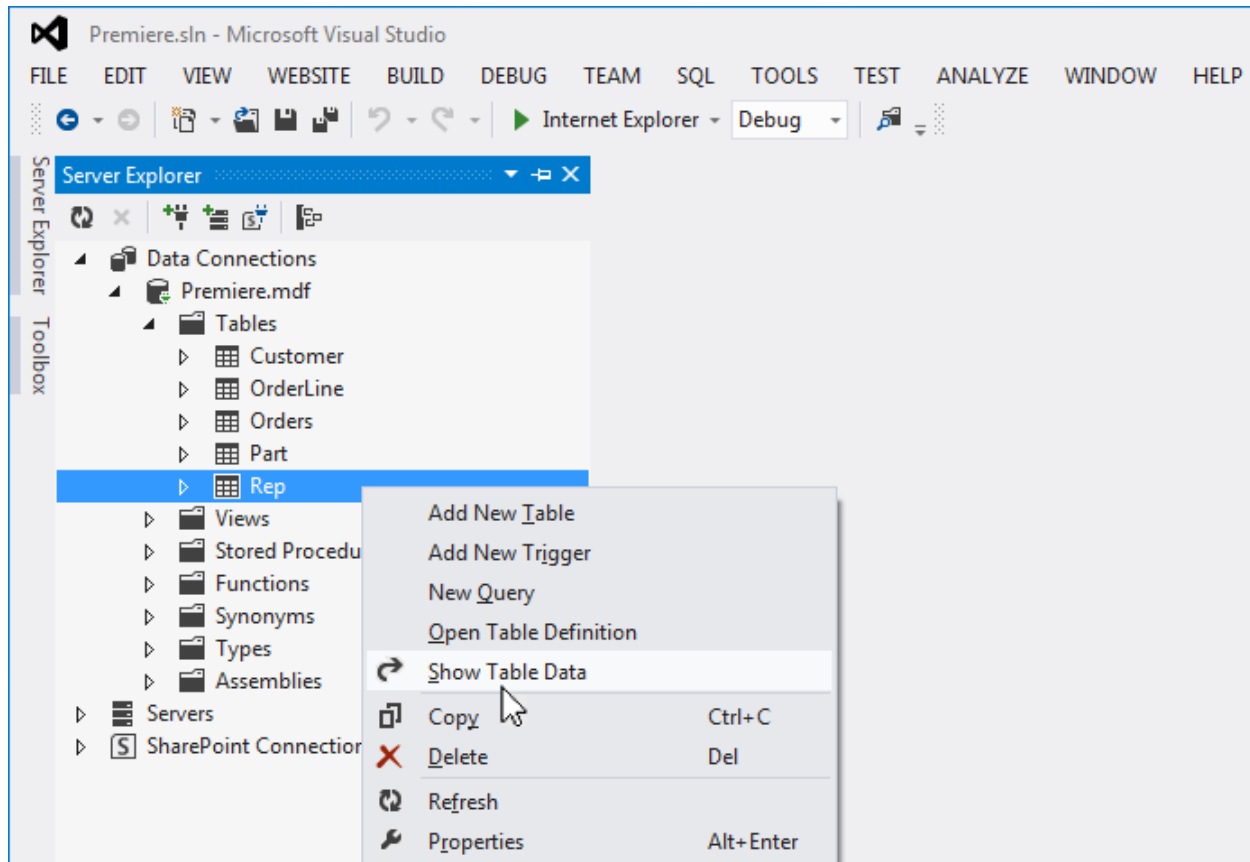


This process should be repeated for all tables that have linked relationships and are viewable in the database design as represented in the E-R diagram. Save each table when you are done creating the relationship before closing the table design. If the developer forgets to execute the save operation, he or she will be prompted to do so when closing the table design.

Working with Table Data

When the database design is complete, then records can be inserted into the tables. To open a SQL server database table to add records, as well as to view and work with the data:

1. (If necessary) click the “expand” [▶] arrow in front of “Tables” in the list of database objects to drill down and view the table object names
2. Right-click the table name and select “Show Table Data” from the “drop-down” shortcut menu



3. Key the data into the fields (columns) one record at a time
4. Data is committed (saved) to the table automatically moving to a new record, so it does not need to be saved manually
5. Close the table when finished with it

The "Part" Table

Part: Query(carl-a...ATA\PREMIERE.MDF) X						
	PartNum	Description	OnHand	Class	Warehouse	Price
▶	AT94	Iron	50	HW	3	24.9500
	BV06	Home Gym	45	SG	2	794.9500
	CD52	Microwave Oven	32	AP	1	165.0000
	DL71	Cordless Drill	21	HW	3	129.9500
	DR93	Gas Range	8	AP	2	495.0000
	DW11	Washer	12	AP	3	399.9900
	FD21	Stand Mixer	22	HW	3	159.9500
	KL62	Dryer	12	AP	1	349.9500
	KT03	Dishwasher	8	AP	3	595.0000
	KV29	Treadmill	9	SG	2	1390.0000
*	NULL	NULL	NULL	NULL	NULL	NULL

The "Rep" Table

Rep: Query(carl-a...ATA\PREMIERE.MDF) X									
	RepNum	LastName	FirstName	Street	City	State	Zip	Commission	Rate
▶	20	Kaiser	Valerie	624 Randall	Grove	FL	33321	20542.5000	0.05
	35	Hull	Richard	532 Jackson	Sheldon	FL	33553	39216.0000	0.07
	65	Perez	Juan	1626 Taylor	Fillmore	FL	33336	23487.0000	0.05
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

The "Customer" Table

Customer: Query(...TA\PREMIERE.MDF) X									
	CustomerNum	CustomerName	Street	City	State	Zip	Balance	CreditLimit	RepNum
	148	Al's Appliance and Sport	2837 Greenway	Fillmore	FL	33336	6550.0000	7500.0000	20
	282	Brookings Direct	3827 Devon	Grove	FL	33321	431.5000	10000.0000	35
	356	Ferguson's	382 Wildwood	Northfield	FL	33146	5785.0000	7500.0000	65
	408	The Everything Shop	1828 Raven	Crystal	FL	33503	5285.2500	5000.0000	35
	462	Bargains Galore	3829 Central	Grove	FL	33321	3412.0000	10000.0000	65
	524	Kline's	838 Ridgeland	Fillmore	FL	33336	12762.0000	15000.0000	20
	608	Johnson's Department Store	372 Oxford	Sheldon	FL	33553	2106.0000	10000.0000	65
	687	Lee's Sport and Appliance	282 Evergreen	Altonville	FL	32543	2851.0000	5000.0000	35
	725	Deerfield's Four Seasons	282 Columbia	Sheldon	FL	33553	248.0000	7500.0000	35
	842	All Season	28 Lakeview	Grove	FL	33321	8221.0000	7500.0000	20
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

The "Orders" Table

Orders: Query(car...ATA\PREMIERE.MDF) X			
	OrderNum	OrderDate	CustomerNum
▶	21608	10/20/2013	148
	21610	10/20/2013	356
	21613	10/21/2013	408
	21614	10/21/2013	282
	21617	10/23/2013	608
	21619	10/23/2013	148
	21623	10/23/2013	608
*	NULL	NULL	NULL

The "OrderLine" Table

OrderLine: Query...ATA\PREMIERE.MDF) X					
	SeqNum	OrderNum	PartNum	NumOrdered	QuotedPrice
▶	1	21608	AT94	11	21.9500
	2	21610	DR93	1	495.0000
	3	21610	DW11	1	399.9900
	4	21613	KL62	4	329.9500
	5	21614	KT03	2	595.0000
	6	21617	BV06	2	794.9500
	7	21617	CD52	4	150.0000
	8	21619	DR93	1	495.0000
	9	21623	KV29	2	1290.0000
*	NULL	NULL	NULL	NULL	NULL

To delete a record, click the "row select" box and then press the <Delete> key on your computer keyboard.

Part: Query(carl...ATA\PREMIERE.MDF) X						
	PartNum	Description	OnHand	Class	Warehouse	Price
	AT94	Iron	50	HW	3	24.9500
	BV06	Home Gym ...	45	SG	2	794.9500
	CD52	Microwave Ove...	32	AP	1	165.0000
	DL71	Cordless Drill	21	HW	3	129.9500
	DR93	Gas Range	8	AP	2	495.0000
	DW11	Washer	12	AP	3	399.9900
	FD21	Stand Mixer	22	HW	3	159.9500
▶	KL62	Dryer	12	AP	1	349.9500
	KT03	Dishwasher	8	AP	3	595.0000
	KV29	Treadmill	9	SG	2	1390.0000
*	NULL	NULL	NULL	NULL	NULL	NULL

The following “confirmation” dialog window will be displayed. Click the <Yes> button to confirm the deletion (***do not actually delete any records right now***).

