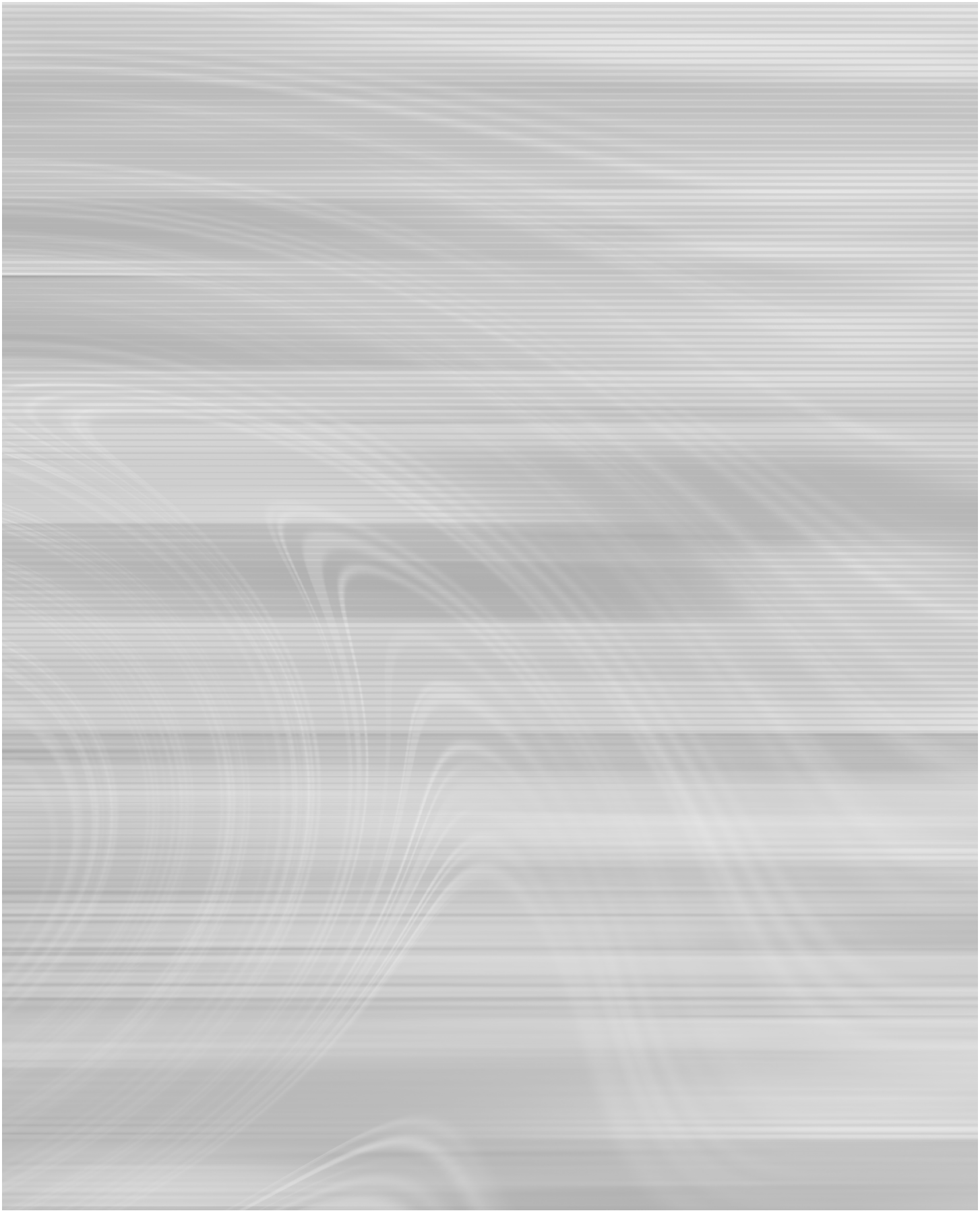
PART 7

Designing an Access Project



CHAPTER 26

Building Tables in an Access Project

If you worked through the previous chapters in this book, you created all the components and have all the knowledge necessary to produce a fully functioning desktop application in Microsoft Office Access 2007. If you built an application only for your personal use, you might never need to learn more about advanced Office Access 2007 features. But if you plan to share your application with multiple users or work with large amounts of data, it might be time to consider building an Access project.

Unlike an Access desktop database (.accdb), an Access project does not contain any tables or queries. When you define a project, you must specify a connection to a Microsoft SQL Server database. The database server provides the tables and queries—views, functions, and stored procedures—that your application will use.

As with any database design, the first step in creating an Access project is building the tables. In this chapter, you will learn how to

- Create a new project file by building a new database on the server
- Create a new project file by connecting to an existing database on the server
- Create a new table in Design view
- Select the best data type for each column
- Create check constraints to validate the data in your tables
- Define a primary key for your tables
- Add indexes to your tables
- Learn how to create relationships for your tables
- Learn how to manage your relationships and tables using database diagrams
- Set options that affect how you work in table design

Creating a New Project File

Before you can get started building a new project file, you need to make sure that you can interface with an SQL Server database that can support your Access project. The file extension that Access uses to store your project files is .adp. You must either

- Have access to SQL Server version 2000 or later and also have full Create Database and Modify Database permissions, or
- Have installed Microsoft SQL Server 2005 Express Edition. See the Appendix, "Installing Your Software," for more information on how to download and install SQL Server 2005 Express Edition.

Note

Although it is possible to use SQL Server 6.5 or later when creating an Access project (.adp), many of the tools discussed in this chapter are supported only by SQL Server 2000 or later. If you do not have access to SQL Server 2000, try installing SQL Server 2005 Express Edition instead so that you can become familiar with all the design options supported by project files (.adp) in Access 2007.

Building a New SQL Server Database

Any Access project file (.adp) must be connected to an SQL Server database to store its tables, views, functions, and stored procedures. When you create a new project file, you can also create a new database. If you don't already have a database to connect to on the server, you need to create one. Click the Blank Database button in the middle of the Getting Started screen, as shown in Figure 26-1. Access displays the Blank Database task pane where you enter a name for your new database in the File Name text box and a location to save the file beneath the File Name text box. You can modify the name of this database by typing in the File Name text box. To create a new Access project file, click the Browse button, also shown in Figure 26-1.

In the File New Database dialog box that appears, as shown in Figure 26-2, select the drive and folder you want by clicking the links on the left side and browsing to your destination folder. In this example, we selected the Development subfolder under the Documents folder on our computer. (Remember from Chapter 2, "Exploring the New Look of Access 2007," we created this Development folder as a trusted location, so Access will enable all the content.) Select Microsoft Office Access Projects from the Save As Type list to have Access create an Access project instead of the default .accdb database file. Next, go to the File Name text box, and type the name of your new Access project. Access appends an .adp extension to the file name for you. Name the new project file ContactTracking, and click OK to return to the Getting Started screen. Finally, click the Create button to create your project file.

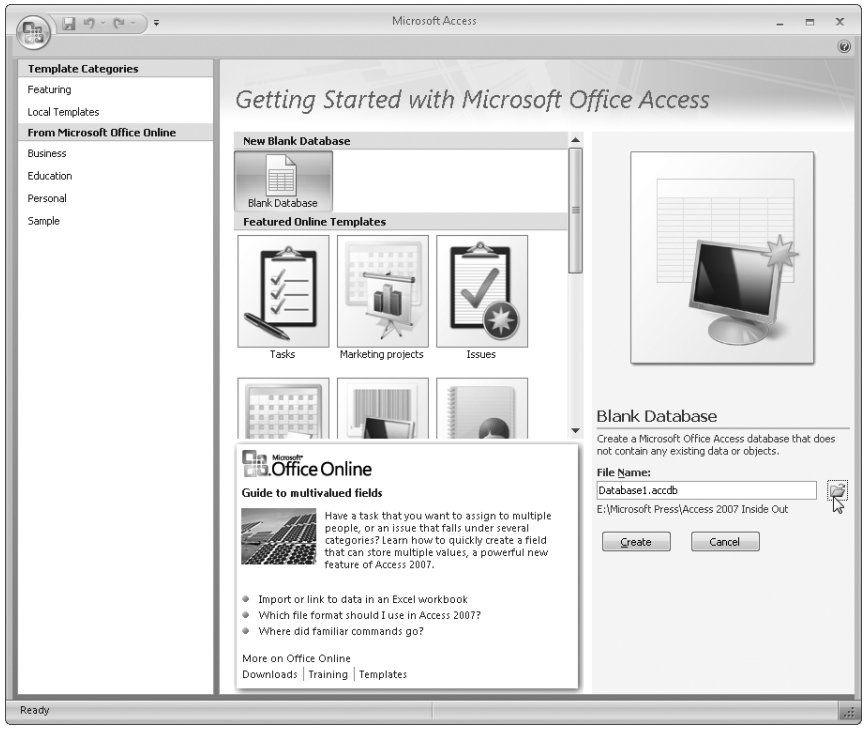


Figure 26-1 Click the Blank Database button to begin creating a new Access project.

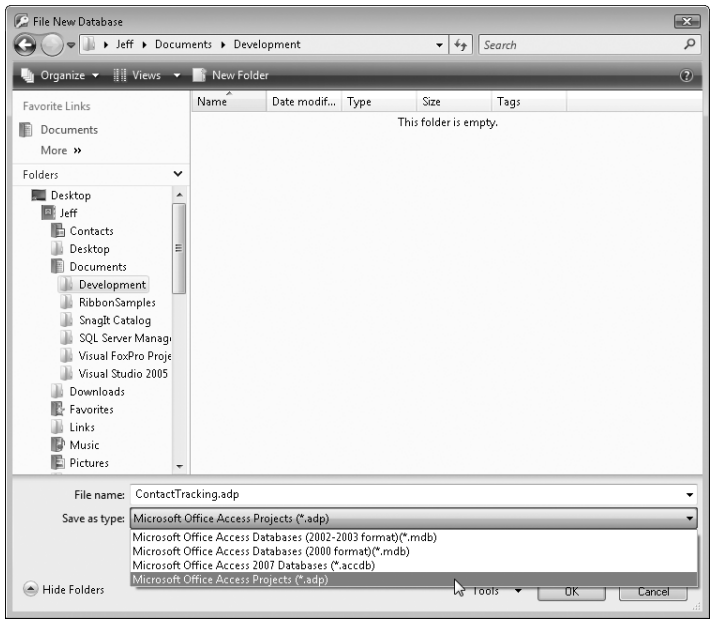


Figure 26-2 Name the new project file in the File New Database dialog box, and select Microsoft Office Access Projects from the Save As Type list.

Access displays a message box asking whether you want to connect to an existing SQL Server database, as shown in Figure 26-3. If you click Yes, Access displays a Data Link Properties dialog box where you enter the connection information for your existing SQL Server database. Because we're creating a new SQL Server database, click the No button to continue.

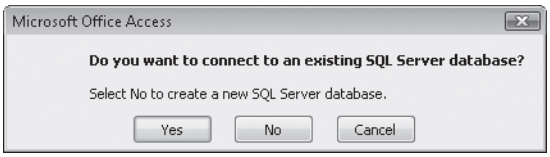


Figure 26-3 Click No to create a new SQL Server database to which to connect.

Access creates the project for you and opens it. Because this is a new project, the Microsoft SQL Server Database Wizard opens and asks you what SQL server you want to connect to and what you want to name your new database. If you are connected to a large network, it might take a while for the wizard to appear because it has to search the entire network to compile a list of all available SQL servers.

INSIDE OUT

An Access Project Is Not a Database

When you were working with .accdb files, you had all the elements of a database in one convenient file. Because an .accdb file can stand on its own, it's logical to refer to it as a *database*. Project files are different. In an Access project, the actual database is composed of all the tables, functions, views, and stored procedures that are kept on the server. Your Access project file merely connects to the database to make available these objects that are on the server, and you build the forms, reports, macros, and modules that define your application in the project file. SQL Server (or SQL Server 2005 Express Edition) has its own database name, which can be different from the project file you create in Access. It is important to keep this difference in mind, because when you talk about the database in an Access project, you are actually talking about the database stored on the server.

Figure 26-4 shows the first page of the Microsoft SQL Server Database Wizard. The first field shows the name of the server to which you want to connect. Click the arrow to display all the servers that the wizard found. If the name of the server you want to connect to is not listed, you can type the full name in the box instead. If you are using a copy of SQL Server 2005 Express Edition installed on your desktop computer, select the (local) option from the drop-down list.

In most cases, you want to select the Use Trusted Connection check box. This tells Access that you want to connect to the server using the established Windows security protocols (the default for SQL Server 2005 Express Edition). If you or an administrator has decided to enforce SQL Server security using a database server login, you need to complete the Login ID and Password fields using an account with valid Create Database

permissions on the server. You must clear the Use Trusted Connection check box to enable the Login ID and Password boxes.

Figure 26-4 Choose the server connection and database name on the first page of the Microsoft SQL Server Database Wizard.

Access automatically generates a database name by appending *SQL* to the end of the project file name you entered in the File New Database dialog box. If you want to use a different name, you can type it. For this exercise, we will use the provided ContactTrackingSQL name.

Click Next to proceed. If Access discovers that the name you chose conflicts with an existing database on the server, it suggests a new one (the same name with a number appended to the end—such as ContactTrackingSQL2). If the wizard can establish a connection to your server, you will see the page displayed in Figure 26-5. Click Finish to close the wizard, and you are now ready to begin building your new Access project.

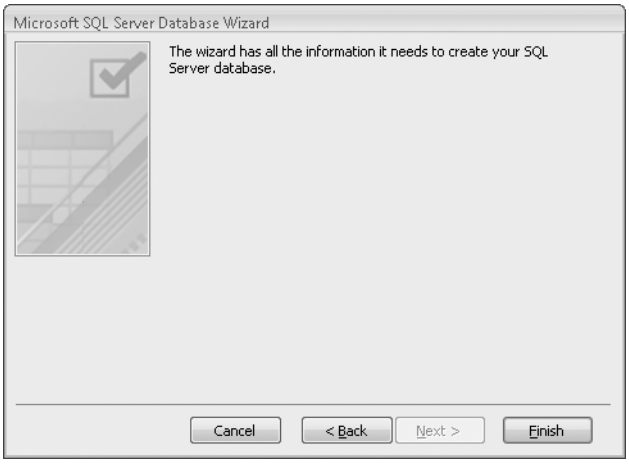


Figure 26-5 The wizard is ready to create the new database on the server.

TROUBLESHOOTING

I keep getting errors when I try to create a new database. What am I doing wrong?

In order for Access to be able to create a new database, the following must be true:

- Access must detect a valid server on the local desktop computer or on the network. If you are attempting to connect to SQL Server, check with your administrator and make sure that the server is properly registered and that your computer can link over your network to the computer on which the server resides. If you are using SQL Server 2005 Express Edition, make sure it is fully installed and configured and that you have started it. (See the Appendix for more information.)
- You must have Create Database permissions that can be verified through a trusted connection or by using SQL Server security. If you are connecting to SQL Server, check with the administrator to make sure these permissions have been created for you. On SQL Server 2005 Express Edition, these permissions should exist by default. If you are still having trouble, try using **sa** as the login ID with no password (the default System Administrator login).

Connecting to an Existing SQL Server Database

If the database has already been created for you or if you want to build a project that connects to an existing database, the first steps are similar to creating a new SQL Server database. Click the Blank Database button in the middle of the Getting Started screen, as shown in Figure 26-6. Access displays the Blank Database task pane where you enter a name for your new database in the File Name text box and a location to save the file beneath the File Name text box. You can modify the name of this database by typing in the File Name text box. Next, click the Browse button, also shown in Figure 26-6.

In the File New Database dialog box that appears, as shown in Figure 26-7, select the drive and folder you want by clicking the links on the left side and browsing to your destination folder. In this example, we selected the Development subfolder under the Documents folder on our computer. Select Microsoft Office Access Projects from the Save As Type list to have Access create an Access project instead of the default .accdb database file. Next, go to the File Name text box, and type the name of your new Access project. Access appends an .adp extension to the file name for you. For this example, name the new project file ContactTracking2, and click OK to return to the Getting Started screen. Finally, click the Create button to create your project file.

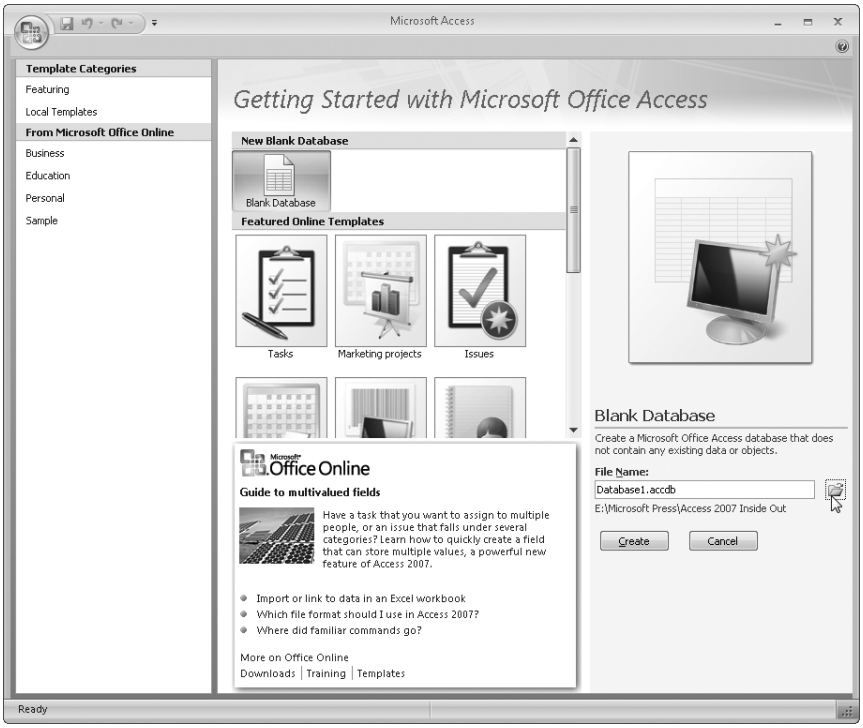


Figure 26-6 Click the Blank Database button to start creating a new Access project with existing data.

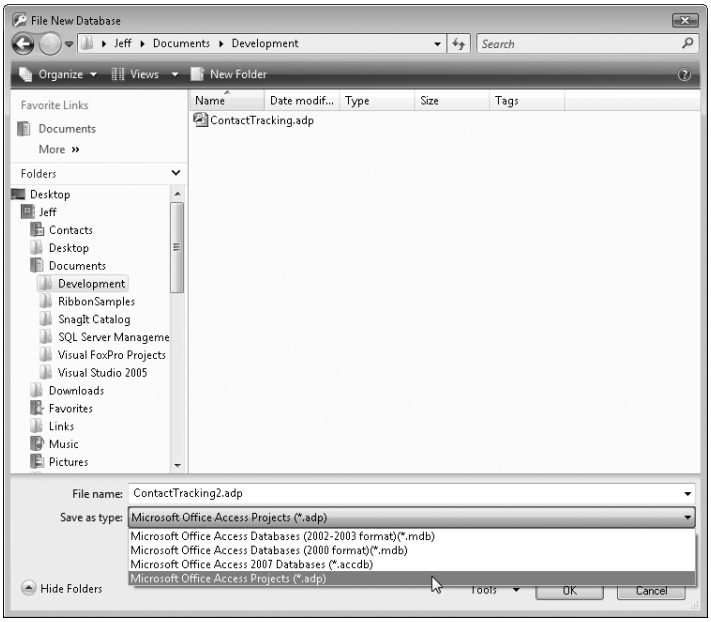


Figure 26-7 Name the new project file ContactTracking2 in the File New Database dialog box.

Access displays a message box asking whether you want to connect to an existing SQL Server database, as shown in Figure 26-8. If you click No, Access displays the Microsoft SQL Server Database Wizard where you can create a new SQL Server database. Because we're connecting to an existing SQL Server database, click the Yes button to continue.

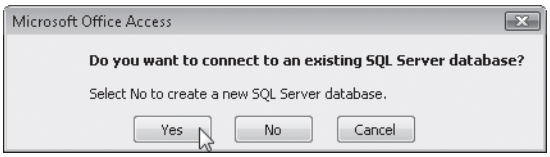


Figure 26-8 Click Yes to connect to an existing SQL Server database.

Next, Access displays the Data Link Properties dialog box, as shown in Figure 26-9. If you are connected to a large network, it might take a while for the dialog box to appear because the wizard has to search the entire network for all valid servers. After the dialog box appears, select the server to which you want to connect from the list in the first box. If the name of the server you want to connect to is not listed, you can type the full name in the box instead. If you are using a copy of SQL Server 2005 Express Edition installed on your desktop computer, select the (local) option from the list.

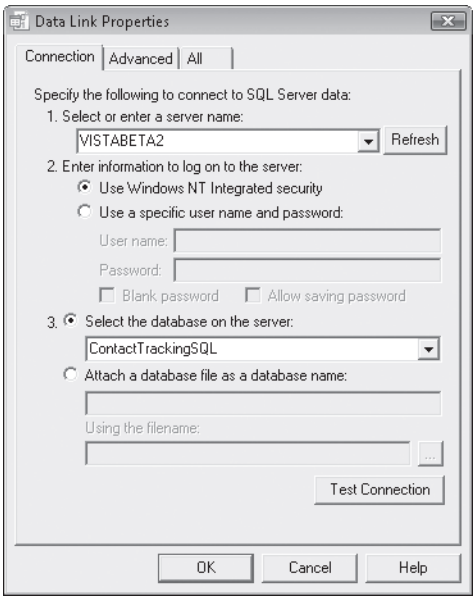


Figure 26-9 Specify how to connect to an existing database using the Data Link Properties dialog box.

> **Note**
>
> If you followed the previous instructions to create a new project with a new database, you can use the ContactTrackingSQL database that you created earlier.

Select Use Windows NT Integrated Security if you want to allow SQL Server to verify your permissions through the Windows NT settings. (In this context, Windows NT means NT 4.0, Windows 2000, Windows XP, Windows Vista, and Windows Server 2003 with SQL Server version 7.0 or later.) If you have a specific SQL Server login ID and password assigned to you, click the Use A Specific User Name And Password option instead, and enter them. When the server requires an SQL Server login, you can also specify that the password is blank (which is not the same as leaving the Password field blank), and you can ask Access to remember the login ID and password you enter.

If the database you want to connect to is already on the server, you can select the database from the second list. If the database you want is not listed, it might not exist on the server to which you have chosen to connect. Try connecting to a different server, or check with your administrator to make sure the database you want has been created.

You can also choose to attach a database file to the server by selecting the Attach A Database File As A Database Name option. (Note that you must have create permission on the server you want to use.) This allows you to connect a preexisting SQL Server master data file (.mdf) to the server and use it to build your project on. You can also use this option to attach Microsoft Access desktop database tables (.accdb) to your server, but the tables will be read-only. When you choose to attach a database, you must enter the database name you want for the attached file and specify the location of the file you want to attach. If you don't know the exact location of the database file, you can browse for it by clicking the Browse (...) button. This will display the dialog box shown in Figure 26-10.

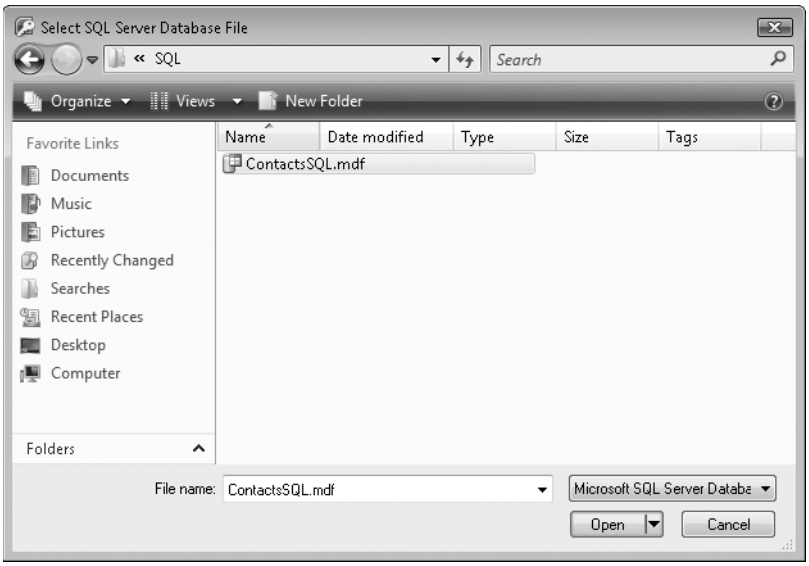


Figure 26-10 Locate a master data file to attach to SQL Server using the Select SQL Server Database File dialog box.

Use the dialog box to locate the .mdf file you want to attach, and then click the Open button. Access displays the path to the database file in the Data Link Properties dialog box, as shown in Figure 26-11. Make sure you type a unique name for your database in the text box above the Using The Filename text box. For example, if you are attaching the ContactsSQL.mdf file, enter **ContactsSQL** as the database name.

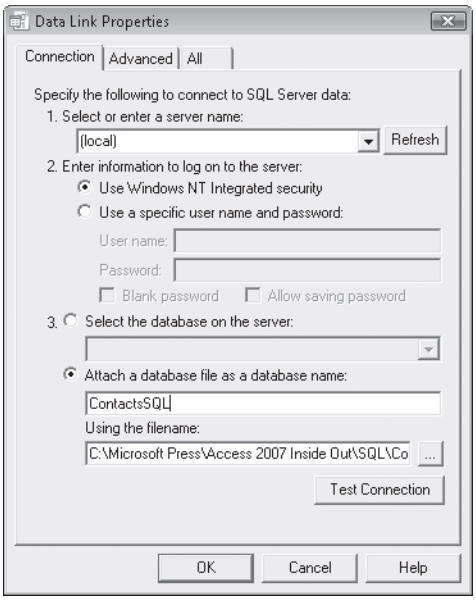


Figure 26-11 You can attach a database file as a database name using the Data Link Properties dialog box.

> **Note**
>
> If you want to work with the tables, functions, views, and stored procedures in the sample project files included on the companion CD, you will need to attach the Contacts-SQL.mdf file provided on the CD to SQL Server or SQL Server 2005 Express Edition using the preceding instructions. If you installed the sample files in the default C:\Microsoft Press\Access 2007 Inside Out folder, you can also execute the Attach Contacts.bat file that you'll find in the SQL subfolder. In Windows Vista, right-click the file, and click Run As Administrator on the shortcut menu. If you installed the sample files in a different folder, you'll need to modify the Attach Contacts.sql and Attach Contacts.bat files first to point to the correct location.

When you are ready, click the Test Connection button. If Access is able to connect, it will display a Test Connection Succeeded message. If it fails, a problem might exist

with the server connections. Contact your network administrator, or consult your SQL Server documentation for more assistance.

If you are having errors connecting to the server, see "I keep getting errors when I try to create a new database" on page 1452. If Access informs you the server is running but it can't connect to your database, see "An Access Project Is Not a Database" on page 1450 for more help.

Click OK in the Data Link Properties dialog box, and Access creates the new Access project using existing data, as shown in Figure 26-12. You are now ready to edit the existing database.

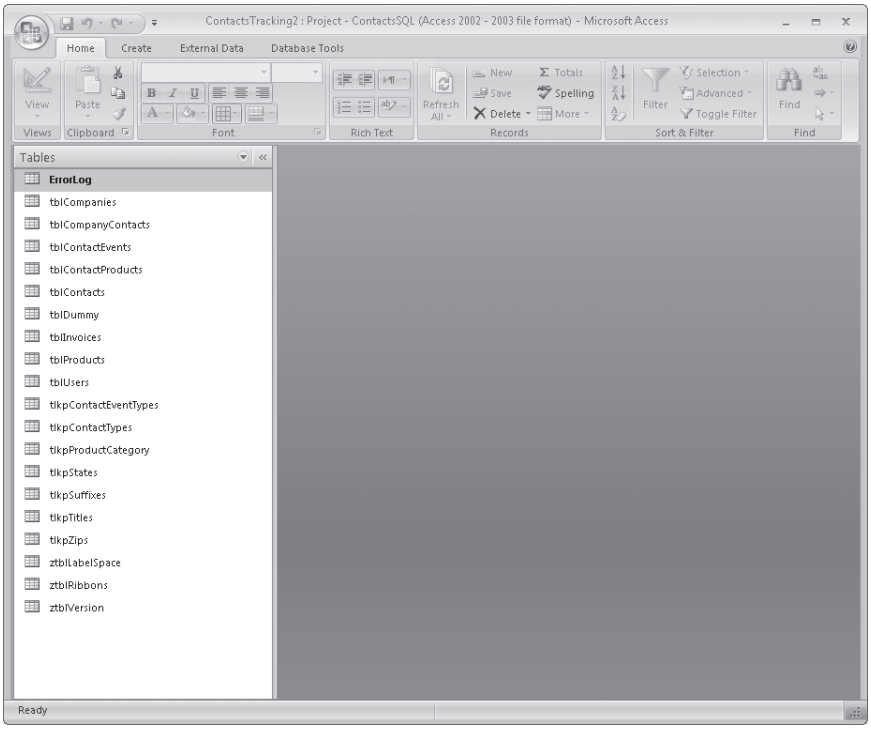


Figure 26-12 Here is the Navigation Pane of a new project connected to an existing database.

Creating a Table in Design View

Now that you have created a project file, let's take a look at creating a table. The features that allow you to create tables by using wizards or by entering data in a desktop application (.accdb) do not exist in an Access project (.adp). The only method for creating tables (other than importing them from another source or using specific code to create them) is to create them in Design view.

Creating a new table is easy. Open the project file (ContactTracking.adp) connected to the new database (ContactTrackingSQL) that you created earlier. On the Create tab, in the Tables group, click the Table Design button, and a blank table opens in Design view, as shown in Figure 26-13.

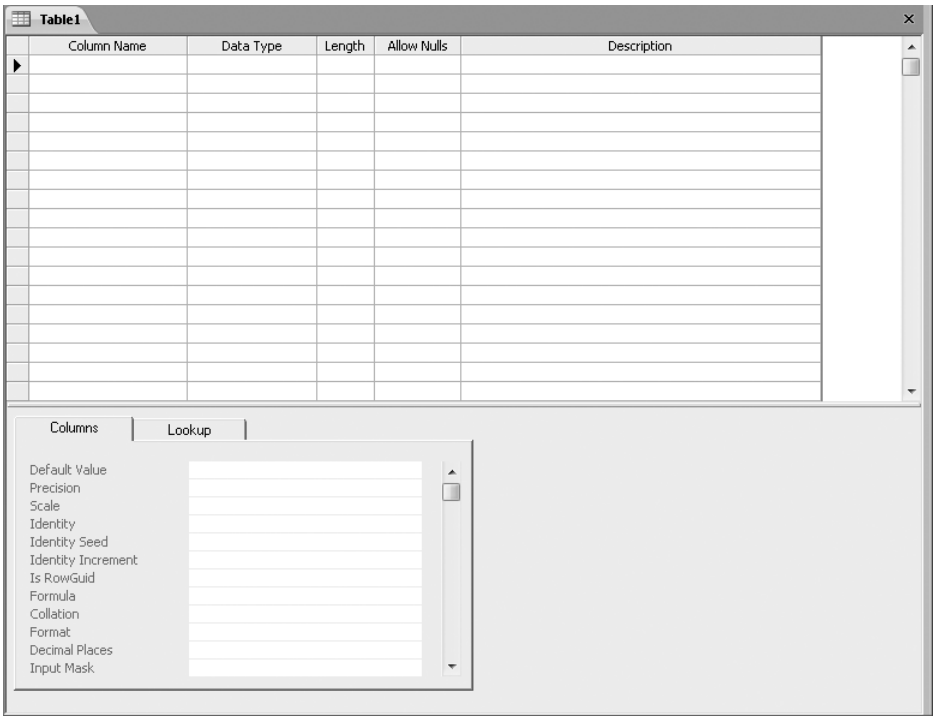


Figure 26-13 Click the Table Design button to begin creating a new table in Design view in an Access project.

Because the tables of an Access project are stored in SQL Server, the table design inherits some of the new properties and definitions supported by SQL Server. You might notice that fields are now called *columns*, and instead of records you have *rows*. This naming convention is similar to the design elements discussed in Article 1, "Designing Your Database Application," on the companion CD. Functionally, columns are the same as fields, and rows are the same as records.

Listed horizontally are some additional properties that SQL Server uses to define the columns in your table. You can also see properties listed on the tab control in the bottom section of the window. Some of the listed properties are unavailable (they appear dimmed) depending on the data types that you select for your columns.

For details about column properties, see Table 26-3, "SQL Server Table Column Properties," on page 1468.

In Chapter 4, "Creating Your Database and Tables," you learned how to create tables for an Access desktop database (.accdb) by building some of the tables in the Contact-Tracking database. Now let's learn how to build those same tables in an Access project (.adp).

Defining Columns

First, let's build a Companies table that is similar to the tblCompanies table that you'll find in the Conrad Systems Contacts sample database (Contacts.accdb). With the table in Design view, click the first row under Column Name; type the name of the first column, **CompanyID**; and press Tab. Notice that Access fills in a default data type and length. You'll learn how to change this default later in this chapter. Click the arrow in the Data Type cell or press Alt+Down Arrow to open the list of data type options, as shown in Figure 26-14.

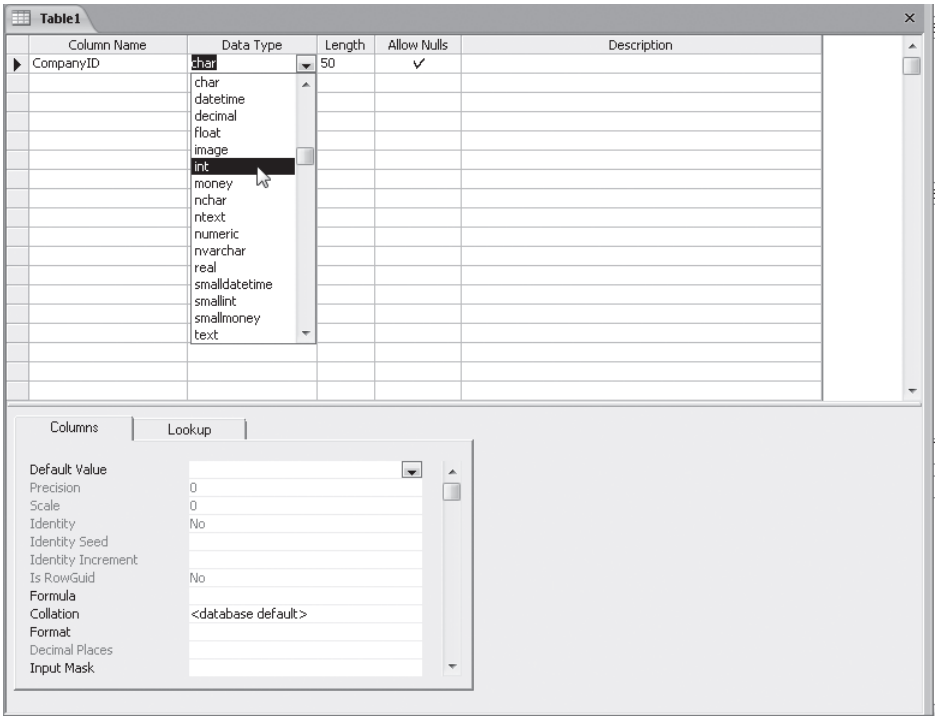


Figure 26-14 You can select SQL Server data types from the drop-down list of data type options.

INSIDE OUT

Rules for Identifiers and Column Names

As you learned in Chapter 4, it is a good idea to follow a sensible naming convention when defining fields in your Access .accdb tables. One of the primary reasons for this was so that it would be easier to interface with SQL Server later. It's possible to convert an existing desktop database (.accdb) to a project with connected tables, views, functions, and stored procedures in SQL Server. If you plan your field (column) names carefully, this process happens much more smoothly.

All objects in an SQL Server database, including tables, views, functions, and stored procedures, must have names that follow a set of naming conventions called the *Rules for Identifiers*. These rules also apply to naming your columns in a table in an Access project. (Remember, the table is actually stored in SQL Server.)

A column name can be up to 128 characters long and can include any combination of letters, numbers, and the symbol characters @, _, #, or $. The column name must begin with a letter, _, @, or #. The name also should not contain spaces and cannot be a Transact-SQL reserved word (such as Select, Alter, or Create).

Microsoft *SQL Server Books Online* provides an invaluable reference to all the features of Microsoft SQL Server 2005. You can download a free copy (a set of help files) from *www.microsoft.com/technet/prodtechnol/sql/2005/downloads/books.mspx*. In *SQL Server Books Online*, you can find a list of all the reserved words in Transact-SQL.

You should give your columns meaningful names and use the same name throughout for a column that occurs in more than one table. You should avoid using column names that might also match any name internal to Office Access 2007, Microsoft Visual Basic, or SQL Server. For example, all objects have a Name property, so it's a good idea to qualify a column containing a name. For example, use CustomerName, CompanyName, VendorName, or something similar. You should also avoid names that are the same as built-in functions, such as Date, Time, Now, or Space.

You will see that if you type a column name that doesn't meet the Rules for Identifiers criteria, the name appears in delimited brackets ([ ]). *Delimited* means that the object name does not meet the criteria of the Rules for Identifiers naming convention. Delimited object names must always be encased in brackets ([ ]) or double quotes ("), but you can use double quotes only if the server has been set up to accept double quotes surrounding identifiers (field names). When names are delimited, SQL Server can still recognize and use them, but it must spend extra processing time converting those delimited names to *limited* names that meet the criteria for the Rules for Identifiers naming convention. Also, whenever you refer to objects with delimited names in forms, queries, or other procedures, you must continue to enclose them in brackets or double quotes. As you can see, you will have an easier time building your database, and it will run more efficiently if you name all your objects using the Rules for Identifiers.

Column Data Types

Before you define all the columns for the Companies table, you need to understand the data types available in SQL Server. Table 26-1 describes all the data types supported by the Access project table design facility. For the most part, the listed data types are similar to the ones supported by an Access desktop database (.accdb). A table stored in SQL Server provides a much wider selection of data types than does an Access desktop database (.accdb). This allows you to be more exact in the amount of space that each column must consume and thus gives you the opportunity to save space and processing time for your database.

Table 26-1 SQL Server Data Types

Data Type	Length (Bytes)	Description	Equivalent Desktop Database Data Type
bigint	8	Fixed-point integer[1] from -2^{63} to $+2^{63} - 1$.	(None)
binary	Fixed, up to 8000	Fixed-length binary data.	(None)
bit	1	True/false values. SQL Server can store up to eight columns of the bit data type in one byte.	Yes/No
char	Fixed length, up to 8000	Non-Unicode (single-byte character set) fixed-length character values.	(None)
datetime	8	Date/time value from January 1, 1753, to December 31, 9999, precise to 0.03 seconds.	Date/Time
decimal	5, 9, 13, or 17, depending on precision	An alias for numeric. Fixed-precision numeric data from -10^{38} to $+10^{38}$. Precision (the number of digits) can be up to 38, and Scale (the number of digits to the right of the decimal point) can be as large as the specified precision.	Number (Decimal)

[1] An Access 2007 project does not support the bigint data type. If you attempt to open a table in a project file that has a bigint data type included in the design, Access will not be able to open the table in Design view.

Data Type	Length (Bytes)	Description	Equivalent Desktop Database Data Type
float	8	Floating-precision numeric data from -1.79×10^{308} to $+1.79 \times 10^{308}$. Although you can define any precision from 1 to 53 bits in SQL Server, the table design facility in a project automatically sets precision to 53 (8 bytes) when you select float.	Number (Double)
image	16 plus length of the image	OLE Object data.	OLE Object
int	4	Fixed-point integer from –2,147,483,648 to +2,147,483,647.	Number (Long Integer)
money	8	Currency data from -2^{63} to $+2^{63} - 1$, with four decimal places.	Currency
nchar	Fixed, up to 8000 (4000 characters)	Unicode (double-byte character set) fixed-length character values.	(None)
ntext	16 plus length of the text (maximum 1 billion characters)	Varying-length Unicode character values.	Memo
numeric	9	An alias for decimal. Fixed-precision numeric data from -10^{38} to $+10^{38}$. Precision (the number of digits) can be up to 38, and Scale (the number of digits to the right of the decimal point) can be as large as the specified precision.	Number (Decimal)
nvarchar	Varying, up to 8000 (4000 characters)	Varying-length Unicode character values.	Text

Data Type	Length (Bytes)	Description	Equivalent Desktop Database Data Type
real	4	Floating-precision numeric data from –3.4 × 10^{38} to +3.4 × 10^{38}. Although you can define any precision from 1 to 53 bits in SQL Server, the table design facility in a project automatically sets precision to 24 (4 bytes) when you select real.	Number (Single)
smalldatetime	4	Date/time value from January 1, 1900, to June 6, 2079, precise to one minute.	(None)
smallint	2	Fixed-point integer from –32,768 to +32,767.	Number, Integer
smallmoney	4	Currency data from –214,748.3648 to +214,748.3647, with four decimal places.	(None)
sql_variant	8016	Data type that can store values of different data types, except for text, ntext, image, and timestamp types. When you define a column using the sql_variant data type, each row in your table can have a different type of data in that column.	(None)
text	16 plus length of the text (maximum 2 billion characters)	Varying-length non-Unicode character values.	Memo

Data Type	Length (Bytes)	Description	Equivalent Desktop Database Data Type
timestamp	8	Database-wide unique number that changes each time a row is updated. SQL Server uses timestamp values to identify when the data was last affected and in what order the rows were affected. You can define only one timestamp column per table, and you should not define an index for a timestamp column because the values are always changing.	(None)
tinyint	1	Fixed-point integer from 0 to 255.	Number, Byte
unique-identifier	16	Globally Unique Identifier (GUID). Declaring the uniqueidentifier data type is useful if the data in your table needs to be uniquely identified apart from all the other data in all the other tables in all the other databases that are networked with your database. You can specify only one column per table as the uniqueidentifier data type.	Number, GUID
varbinary	Varying, up to 8000	Varying-length binary data.	(None)
varchar	Varying, up to 8000	Varying-length non-Unicode character values.	Text
xml	Varying, up to 2 gigabytes	Stores well-formed and optionally schema-bound XML fragments as a series of UTF-16 encoded bytes.	Memo

Note

You can also create user-defined data types in Microsoft SQL Server 2005. For example, you might want to define a data type named StateProvince that is always two characters and must not be null. If you or your administrator has created user-defined data types in your database, Access displays them in the list of available data types in a table's Design view.

INSIDE OUT

Understanding sql_variant

An sql_variant is a special data type that can store a variety of other data types. To do this, SQL Server stores an additional piece of information with the column called *metadata* (information that describes other data). Each row in an sql_variant column can store different data types except for text, ntext, timestamp, image, and sql_variant. Because sql_variant can hold a variety of data types, it can also behave differently than you might expect when comparing and converting the values it contains. It will also take SQL Server longer to work with sql_variant data types because it has to interpret the metadata in each row to learn what type of data is stored there. If you know that the data in your column will always be the same type, you should use the specific data type declaration for that column instead of sql_variant.

Completing the Columns in the Companies Table

You now know enough about column data types to finish the basic design of the Companies table. Earlier, you entered CompanyID as the first column in the table. Now, select int as the data type—equivalent to a long integer in a desktop database—for this column. You can see that Access sets the length property for you. For some of the remaining columns that are character data types (including char, nchar, ntext, nvarchar, text, and varchar), you will be able to set the data length you want.

The property in the table's Design window, Allow Nulls, specifies whether Null values can be entered in this column. This is similar to the Required property in Access .accdb files but works in the opposite fashion. By default, Allow Nulls is always selected (true). If you want to require data in this column, you should click the Allow Nulls property to remove the check mark. (If you are tabbing from one column to the next, you can also press the Spacebar to toggle the Allow Nulls property.) Because CompanyID needs to be a unique identifier with a value in every row, remove the check mark from Allow Nulls.

The Description property for each column allows you to enter a descriptive phrase. Access displays this description on the status bar (at the bottom of the Access window) whenever you select this column in a view query or a function query in Datasheet view

or in a form in Form view or Datasheet view. For this example, enter **Unique Company ID** in the Description property for the CompanyID column. You don't need to worry about setting any of the custom properties on the Columns or Lookup tab in the lower part of the window for now.

Tab down to the next line, enter **CompanyName** for the Column Name property, and then choose nvarchar as the data type. Enter **50** in the Length property to restrict the length of data entered to no more than 50 characters. Every row should have a value in the CompanyName column, so clear Allow Nulls for this column. In Description, enter **Company Name**.

Use the information listed in Table 26-2 to complete the design of the table shown in Figure 26-15.

Table 26-2 Column Definitions for the Companies Table

Column Name	Data Type	Length	Allow Nulls	Description
CompanyID	int	4	No	Unique company ID
CompanyName	nvarchar	50	No	Company name
Department	nvarchar	50	Yes	Department
Address	nvarchar	255	Yes	Address
City	nvarchar	50	Yes	City
County	nvarchar	50	Yes	County
StateOrProvince	nvarchar	20	Yes	State or province
PostalCode	nvarchar	20	Yes	Postal/ZIP Code
PhoneNumber	nvarchar	30	Yes	Phone number
FaxNumber	nvarchar	30	Yes	Fax number
Website	ntext	16	Yes	Web site address
ReferredBy	int	4	Yes	Contact who referred this company

When you are finished entering all the column definitions, click the Save button on the Quick Access Toolbar to save the table. Access displays the Choose Name dialog box, as shown in Figure 26-16. Name the table **Companies**, and click OK.

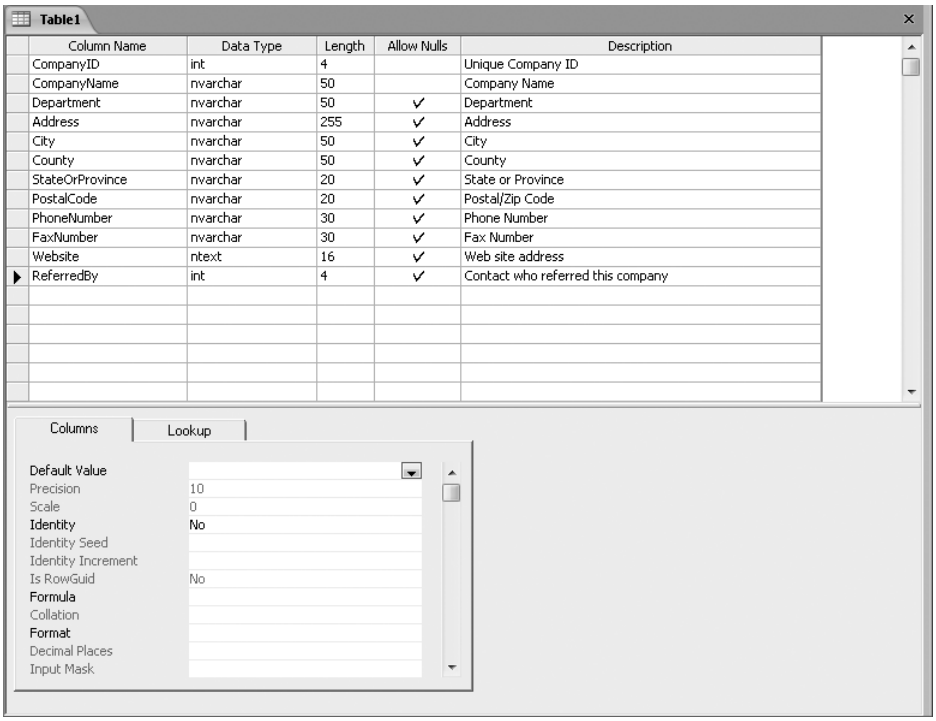


Figure 26-15 Your completed column definitions for the Companies table should look like this.

Figure 26-16 Enter a name for the new table in your Access project.

You haven't defined a primary key for this new Companies table, so Access displays a warning and asks you whether you want to add a primary key now. You'll return to this table later to define the primary key, so click No to finish saving the table, and then close the table definition window.

Understanding Column Properties

You can further define how your database will store and handle your data by specifying additional column properties. Table 26-3 displays a complete listing of the properties you might need when creating columns. Depending on which data type you select, certain properties become available in the lower half of the table's Design window to customize your column definition. All the properties are displayed, but only some of them are relevant depending on the data type you choose. (For example, it doesn't make

much sense to specify the number of decimal places for the text nvarchar data type.) Unavailable properties appear dimmed.

Table 26-3 SQL Server Table Column Properties

Property	Description
Default Value	The default value entered in the column if no other value is entered. You can type a default value directly or select from a list of global default values. SQL Server enters the Default Value property for a column when it saves the row, not when you first begin to enter data in the row. If you want to display a default value for data entry in a column, create a form, bind it to the table, and then set the default value of the control in the form.
Precision	For decimal and numeric data types, the maximum number of digits allowed. The default value is 18, and you can specify an integer value from 1 to 38. For other numeric data types, SQL Server sets the Precision property depending on the data type.
Scale	For decimal and numeric data types, the maximum number of decimal digits stored. Scale must be less than or equal to Precision. See also Decimal Places.
Identity	For fixed-point data types (integer and decimal), this converts the column to a system-generated number that functions much like AutoNumber in Access desktop databases. An identity column cannot be null and cannot have a default value. In addition, you can specify the starting value (Seed) and Increment value for Identity. The default value is No. Set Identity to Yes to create an identity column or Yes (Not For Replication) to create an identity column that maintains its value when the data is replicated.
Identity Seed	The starting value for an identity column. The default value is 1.
Identity Increment	The amount that each value in the Identity property is incremented for new rows. You can enter an integer from 1 to 2,147,483,647. The default value is 1.
Is RowGuid	If the data type of the column is uniqueidentifier, setting this to Yes tells SQL Server to use it as a globally unique identifier for replication. The default setting is No.
Formula	The formula for a computed column.
Collation	The collating (sorting) sequence that SQL Server applies by default to a text column when its values are returned as rows in the results of a query. Click the property, and then click the Build (...) button to open a dialog box to choose from other collation sequences available on your computer or on the server. The default is the collation sequence defined for your database.
Format	The display format for the column. For details about custom formats, see "Setting Control Properties" on page 651.

Property	Description
Decimal Places	The number of decimal places *displayed*. (This is not the same as Scale, which controls the decimal places *stored*.) The default value is [Auto], which indicates that the number of decimal places displayed automatically adjusts depending on the precision of the number stored in the column. See also Scale.
Input Mask	An input mask that controls and formats how data is entered. For details about input masks, see "Defining Input Masks" on page 170.
Caption	You can enter a fully descriptive column name that Access displays in form labels and in report headings. Because you should create column names with no embedded spaces, you can use the Caption property to specify a name that includes spaces for Access to use in labels and headers associated with this column in queries, forms, and reports.
Indexed	Identifies whether the column is indexed. The default is No. You can specify Yes (Duplicates OK) to create a nonunique index or Yes (No Duplicates) to create a unique index. See also "Adding Indexes" on page 1471.
Hyperlink	The text column value can be displayed as a hyperlink. The default is No.
IME Mode and IME Sentence Mode	On computers with an Asian-language version of Windows and an appropriate Input Method Editor (IME) installed, these properties control the conversion of characters in kanji, hiragana, katakana, and hangul character sets.
Furigana	You can specify an alternate column name for this property, and SQL Server copies the Furigana (Japanese Text) equivalent of the typed text into the named column.
Postal Address	On a Japanese-language system, this property allows you to specify a control or a column that displays an address based on an entered postal code. You can also use this feature to display a bar code based on an entered address.

The table design facility also lets you define Lookup properties for columns identical to those available in an Access desktop database. You can define Lookup properties for the bigint, bit, char, decimal, float, int, nchar, ntext, numeric, nvarchar, real, smallint, text, tinyint, and varchar data types. You can specify a Display Control for each of the columns as a text box, list box, or combo box. For the bit data type, you can also specify a check box as the Display Control. This allows you to display related information instead of the actual data when you view a table, form, view, or function in Datasheet view. When you create new forms, the controls on the form bound to columns inherit the lookup from the table design. The same suggestions we made about lookups in Chapter 5, "Modifying Your Table Design," also apply here. For more information, see "Taking a Look at Lookup Properties" on page 240.

TROUBLESHOOTING

I'm using SQL Server 7.0, so why can't I see all the column properties or create lookups?

This book assumes that you are working with SQL Server 2000 or later in these chapters on Access projects. It is possible to build projects in earlier versions of SQL Server, but not all the features discussed here are supported, including many column properties and the ability to specify lookups at the table level.

Defining a Primary Key

As you know from the principles of good table design (see Article 1 on the companion CD), it is important to define a primary key to uniquely identify each row in a table. Doing so allows you to define relationships with other tables and reduces redundant data. Also, you cannot update a table that does not have a primary key. Defining a primary key for a table in an Access project is very similar to defining a primary key in an Access desktop database (.accdb).

Remember, you didn't define a primary key for the Companies table. Open that table again in Design view. To define a primary key, select the column that you want to make into a primary key—in this case, the CompanyID column. Then click the Primary Key button in the Tools group on the Design tab under Table Tools. Access automatically creates a UNIQUE index on the selected column and creates a primary key named PK_Companies. Access displays a key symbol to the left of the selected column to acknowledge your definition of the primary key, as shown in Figure 26-17. Be sure to save your changes. Note that when you need to define multiple columns as the primary key, you can select a group of columns by holding down Ctrl and clicking each one.

Because an index is created when you define the primary key, you also have the ability to modify the properties of the primary key as an index. To modify the primary key, you need to access the Indexes/Keys tab of the table properties either by right-clicking any column in Design view and clicking Indexes/Keys on the shortcut menu or by clicking the Indexes button in the Show/Hide group on the Design tab. To learn more about modifying the index properties of the primary key, see the next section.

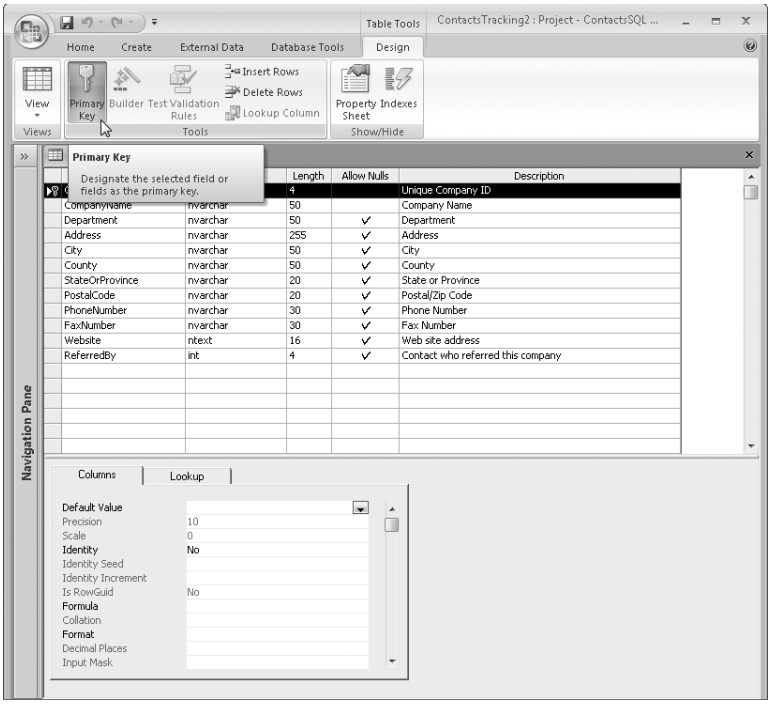


Figure 26-17 Define the CompanyID column as the primary key for the Companies table.

Adding Indexes

Creating indexes in a table allows for faster access to the information in the table when it is being queried, much the same way that you use the index in this book—you find the term you want and jump directly to the pages containing that term. You don't have to leaf through all the pages to find the information you want to see.

Similar to an Access desktop database, SQL Server implements the primary key by creating an index on it. As noted earlier, you must define a primary key to be able to update an SQL Server table from your Access project. You must also define primary keys for your tables to be able to create relationships between tables.

However, you might often use criteria on other columns in a table to select data using a query (view, function, or stored procedure), and you can increase the efficiency of these queries by adding one or more indexes to those columns. Access projects support two basic types of indexes: *clustered* and *nonclustered*. When you define a clustered index on a table, SQL Server physically sequences the rows in the table based on the values in

the index. This, in effect, becomes the default ordering for rows that you fetch from the table. As you can imagine, you can create only one clustered index on a table. If a clustered index doesn't already exist, SQL Server makes the primary key a clustered index when you define it. You can define up to 249 nonclustered indexes to help make fetching your data using criteria faster.

INSIDE OUT

Too Many Indexes Is Not a Good Thing

Indexes don't always speed up the performance of your database. They occupy extra disk space and can also slow down INSERT, UPDATE, and DELETE actions on the table because the index has to be updated each time such an action occurs. A good rule of thumb is to create indexes only on columns in the table that you know you will use often in criteria.

Assume that your users will often query the Companies table using criteria on the City column. To speed up this search, let's create an index on the City column. Open the Companies table in Design view, and then open the Properties window on the Indexes/Keys tab by right-clicking any column in Design view and clicking Indexes/Keys on the shortcut menu. This opens the Properties window and selects the Indexes/Keys tab, as shown in Figure 26-18.

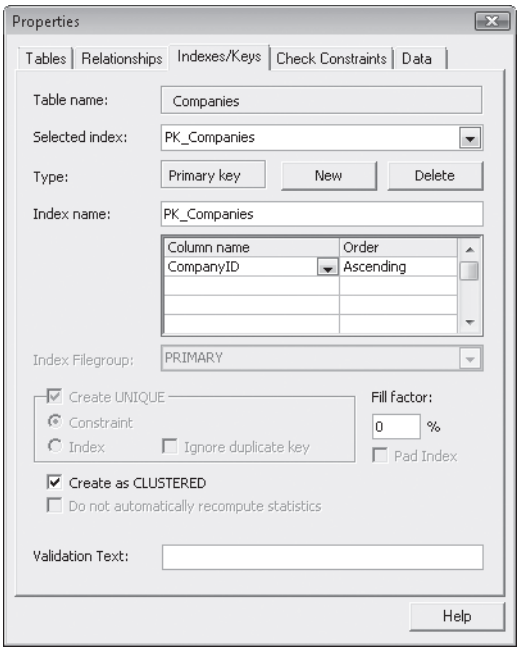


Figure 26-18 You can view the indexes of your SQL Server table on the Indexes/Keys tab of the table's Properties window.

The Indexes/Keys tab always shows you the first index created on the table—usually the one associated with the primary key. If there were more than one index on this table, you could view each of them by selecting them from the Selected Index list. Right now, the only index on the Companies table is the primary key index. To begin defining a new index, click the New button. By default, Access starts a new index called IX_Companies on the first column of the table (the CompanyID column in this case).

To create the index on the City column, start by typing a more descriptive name for the index in the Index Name field. Click the Index Name text box, and change the name to IX_Companies_City by adding **_City** to the end of the index name. When you look up the index in the future, it will be easier to identify its purpose.

Below the Index Name field is a box with two headings labeled Column Name and Order. Here you specify which columns you want to be a part of the index and the order in which each column should be sorted. You can include up to 16 columns in an index. You can specify Ascending or Descending order for each column in the index. If the index contains multiple columns, each column will be sorted in the order it was added to the index. Right now, CompanyID is the only member of the index. Click the CompanyID name to display a list, and replace CompanyID by selecting City from the list. Leave the sort order as Ascending. Because the City column is the only member of this index, you don't need to specify any other columns under Column Name.

The Index Filegroup is the filegroup in which SQL Server saves the index information. SQL Server manages indexes using filegroups and, by default, stores them in the PRIMARY filegroup. If you need to define a large quantity of indexes in the database, you or an administrator might decide to create additional filegroups to better organize your indexes. If additional filegroups are available, you will be able to select them from the drop-down list. For the ContactTracking database, the PRIMARY filegroup should be sufficient for storing all your indexes.

If you select the Create UNIQUE check box, each value entered in the index or key column must be unique. You then have the option to specify whether you are creating a unique index or a unique constraint. If you select Constraint, SQL Server creates an index along with a constraint that checks to make sure the value is unique before adding or updating a row in the table. The benefit of the constraint is that you can supply validation text that is more descriptive than the generic message SQL Server generates for the unique index. However, creating a unique index offers some extra features not available when creating a unique constraint. For example, because a unique index is a physical index, SQL Server sorts the key values in the order specified with the column name, which might enhance the performance of some queries. A unique index also allows you to select the Ignore Duplicate Key check box. If you select the Ignore Duplicate Key check box, then during a transaction to update or add rows, SQL Server discards any rows that would create duplicates. Because it is possible for the City column to contain duplicate values, you do not want to create a unique index on the City column. For this index, do not select the Create UNIQUE check box.

CAUTION!

When you select the Ignore Duplicate Key check box for a unique index, an Update operation that would create a duplicate value will instead delete the row to be updated. This happens because an Update operation is actually a Delete of the affected rows followed by an Insert of the changed rows. The Delete will succeed, but the Insert will be discarded because it is a duplicate.

The Fill Factor and Pad Index options allow you to specify additional information to fine-tune the index performance in the database. The Fill Factor percentage indicates what percent of each index page the database should leave empty when building and updating the index. If space is left on each index page, then it is possible for SQL Server to add new entries to the middle of the index when data is inserted without having to build new pages for the index. Specifying a fill factor greater than the default 0% (no room left on each page) is useful only for tables with large indexes that are updated often. If you specify a fill factor greater than 0%, then you also have the option to select the Pad Index check box. Pad Index uses the Fill Factor percent to pad the interior of each index node—each node (group of index pages with similar key values) in the index will have additional space to grow. Like Fill Factor, Pad Index is useful only for tables with large indexes that are updated often. The index on the City column does not need to take advantage of the Fill Factor or Pad Index option, so leave Fill Factor at the default of 0%.

Selecting the Create As CLUSTERED check box creates the index as a clustered index. Remember, you can create only one index per table as a clustered index because a clustered index dictates the physical order of the rows in the table. The primary key (CompanyID) is already defined as clustered, so do not select the Create As CLUSTERED check box.

The next option is Do Not Automatically Recompute Statistics. To enhance performance, SQL Server creates and maintains statistics on the distribution of data values in your tables. If you select this option and the server already has computed statistics for this index on the table, SQL Server reuses the existing statistics instead of re-creating them. Using this option might speed the creation of an index on a very large table, but the statistics might not be as accurate because of the changes in the index. As a result, the index might not be as efficient as it could be. Because this is the first time you have created an index on the City column, no existing statistics are available to be recomputed. Selecting this check box won't do anything, so leave it blank.

If you create a unique constraint, it is important to enter a descriptive message in the Validation Text box. If the validation of the unique constraint fails, then SQL Server displays the message in the Validation Text box instead of the cryptic message it displays by default. Be sure to state the names of the columns in the index. For example, instead

of entering a message such as **Duplicate value in Companies table**, be more specific by entering **A duplicate value for City was entered in the Companies table**. This lets users know how to correct the data input. The City index is not a unique constraint, so you do not need to enter any validation text.

When you have completed the index on the City column, the information on the Indexes/Keys tab should look like Figure 26-19. You can close the Properties window or click New to add more indexes. If you create an index by mistake, select it from the Selected Index list, and click the Delete button to remove it. Be sure to save the table so that the index will be created on the table.

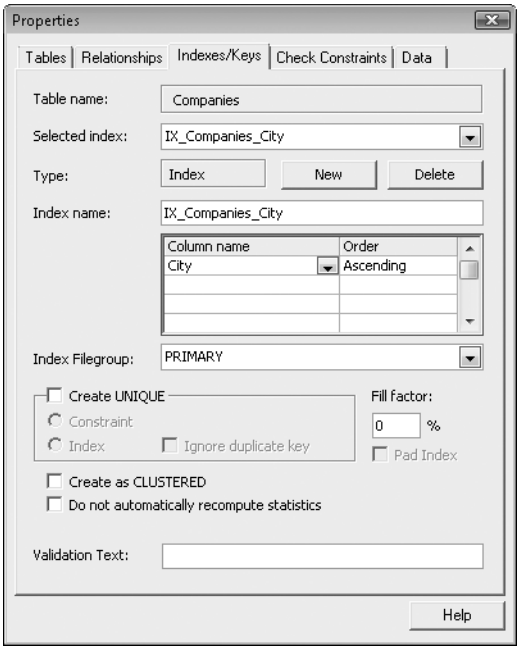


Figure 26-19 The completed IX_Companies_City index should look like this.

Creating Additional Tables in Contact Tracking

So far, we've discussed how to build the Companies table in an Access project. Before you can define constraints and relationships, you'll need to build several more tables. Let's start by building the Contacts table and the CompanyContacts table. Table 26-4 shows you the columns you need for the Contacts table.

Table 26-4 Column Definitions for the Contacts Table

Column Name	Data Type	Length	Allow Nulls	Description
ContactID	int	4	No	Unique contact ID
LastName	nvarchar	50	No	Last name
FirstName	nvarchar	50	Yes	First name
MiddleInit	nvarchar	1	Yes	Middle initial
Title	nvarchar	10	Yes	Person title
Suffix	nvarchar	10	Yes	Person suffix (Jr., Sr., II, and so on)
ContactType	nvarchar	50	Yes	Description of the contact type
BirthDate	datetime	8	Yes	Birth date
DefaultAddress	smallint	2	Yes	Specify Work or Home as the default address
WorkAddress	nvarchar	255	Yes	Address
WorkCity	nvarchar	50	Yes	City
WorkStateOrProvince	nvarchar	20	Yes	State or province
WorkPostalCode	nvarchar	20	Yes	Postal/ZIP code
WorkCountry	nvarchar	50	Yes	Country
WorkPhone	nvarchar	30	Yes	Work phone
WorkExtension	nvarchar	20	Yes	Phone extension
WorkFaxNumber	nvarchar	30	Yes	Fax number
HomeAddress	nvarchar	255	Yes	Address
HomeCity	nvarchar	50	Yes	City
HomeStateOrProvince	nvarchar	20	Yes	State or province
HomePostalCode	nvarchar	20	Yes	Postal/ZIP code
HomeCountry	nvarchar	50	Yes	Country
HomePhone	nvarchar	30	Yes	Home phone
MobilePhone	nvarchar	30	Yes	Mobile phone
EmailName	ntext	16	Yes	E-mail name
Website	ntext	16	Yes	Web site address
Photo	image	16	Yes	Photo of contact
SpouseName	nvarchar	75	Yes	Spouse name
SpouseBirthDate	datetime	8	Yes	Spouse birth date
Notes	ntext	16	Yes	Notes

Column Name	Data Type	Length	Allow Nulls	Description
CommissionPercent	float	8	Yes	Commission when referencing a sale
Inactive	bit	1	No	Contact is inactive

Define ContactID as your primary key for this table by clicking the ContactID column name and then clicking the Primary Key button in the Tools group on the Design tab to define the key. Save the table, and name it Contacts.

Next you need to build the linking table that will act as the "glue" between the Companies table and the Contacts table—CompanyContacts. Table 26-5 shows the columns you need to define to create the CompanyContacts table.

Table 26-5 Column Definitions for the CompanyContacts Table

Column Name	Data Type	Length	Allow Nulls	Description
CompanyID	int	4	No	Company/organization
ContactID	int	4	No	Person within company
Position	nvarchar	50	Yes	Person's position within the company
DefaultForContact	bit	1	No	Is this the default company for this contact?
DefaultForCompany	bit	1	No	Is this the default contact for this company?

Define the combination of CompanyID and ContactID as the primary key for this table by clicking the selection button next to CompanyID and then holding down the Ctrl key and clicking the button next to ContactID. Click the Primary Key button in the Tools group on the Design tab to define the key, and save the table as CompanyContacts. Now that you have three different tables created in the project, you'll learn how to define some constraints that will control how users can enter data in them.

Defining Check Constraints

Constraints are a way of limiting the values that can be entered in a column or group of columns. Constraints are handy because you can use them to enforce data consistency from your users and ensure the relational integrity of your database. You might already be familiar with some of these constraints. Access projects use a group of constraints to create elements in the table design interface that are familiar to Access desktop database (.accdb) users. These elements include the ability to specify a default value in a column, declare primary and foreign keys, and create a table index as a unique index.

Access projects also include the ability to create additional custom constraints called *check constraints* for the columns you create in your tables. Check constraints are similar

to Access desktop database table and field validation rules. You can define constraints to ask the database to check the values of a column (or a group of columns) before the database saves a row to make sure they meet the criteria you specified. Check constraints differ from validation rules because they are applied to the whole row when the row is added or updated. If you enter a value that fails a check constraint criterion, you won't see the validation message until you try to save the row. SQL Server evaluates check constraints in the order that you create them on the table. Let's take a look at how to create a check constraint.

Open the Contacts table in Design view, right-click any column to display the shortcut menu, and then click Constraints. This opens the Properties window with the Check Constraints tab selected. Because you haven't created any check constraints yet, the options in the window will be unavailable and empty. To begin creating a new check constraint, click the New button. This starts a definition for a new check constraint, as shown in Figure 26-20.

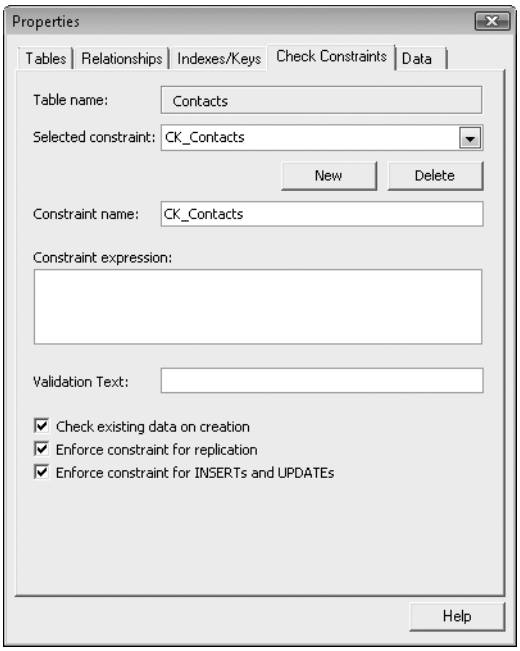


Figure 26-20 On the Check Constraints tab in the Properties window, click New to create a new check constraint on the Contacts table.

The Constraint Name is the name of the current constraint. By default, Access has created a constraint named CK_Contacts. This isn't a very descriptive name for the constraint. Unfortunately, you cannot rename the constraint until you enter a valid expression in the Constraint Expression box, so start by typing a valid expression.

A constraint expression can be any valid SQL expression that evaluates to true or false. One of the rules you need to enforce for the Contacts table is that no contact

can have a commission percent greater than 90 percent. To define this rule, you must enter an expression in the Constraint Expression box that evaluates to false if anyone enters a commission percent greater than 90 percent for a contact. To create the commission percent expression, type the following in the Constraint Expression box: **CommissionPercent <= 0.9**.

When you attempt to save a value in CommissionPercent and this expression evaluates to false (you entered a commission percent greater than 90 percent), Access displays the validation message, and the row will not be saved. You can use many other types of expressions to validate the data that is being added or updated. For now, let's finish creating the commission percent check constraint and make sure it works.

Now that you have a valid expression in the Constraint Expression box, you can change the name of the check constraint to something more meaningful. Click the Constraint Name box, and enter **CK_Contacts_CommissionPercent** as the constraint name. This is a useful name because when you look at objects in the database, CK tells you that this object is a check constraint, Contacts tells you that it is applied to the Contacts table, and CommissionPercent tells you which column is being validated.

> **Note**
>
> You might notice that if you try to move the focus out of the Constraint Expression box while it contains a partially complete or incorrect expression, Access immediately generates an error stating that the expression could not be validated. You can click Yes to keep working on the expression, or you can click No to move the focus away from the Constraint Expression box. If the Constraint Expression box is blank, Access offers you the option to delete the constraint. In this case, click No to continue working on the constraint. Be careful: Access cannot save the constraint unless the expression is valid.

Because all check constraints are evaluated only when the entire row is updated or inserted, it is important to make sure that the validation text is descriptive enough to tell the user where the problem is so that it can be fixed. In the Validation Text box, type **You cannot specify a commission greater than 90%**. This will alert the user that the commission percent value is too great if the validation of the constraint fails.

If you want to make sure that all existing contacts meet the commission percent constraint, leave the Check Existing Data On Creation check box selected. When you select this check box, SQL Server will not save the constraint if any existing rows fail to meet the constraint expression. The Enforce Constraint For Replication check box allows you to apply the constraint to any copies of the database that you distribute using replication. This option is useful if you ever plan to replicate your database. For now, you can clear this check box. In order for SQL Server to apply the constraint whenever users insert or update data, you must leave the Enforce Constraint For INSERTs And UPDATEs check box selected. When you are done creating the commission percent check constraint, it should look like Figure 26-21. If you need to add check constraints,

click the New button. You can also delete a check constraint by choosing it from the Selected Constraint list and then clicking the Delete button.

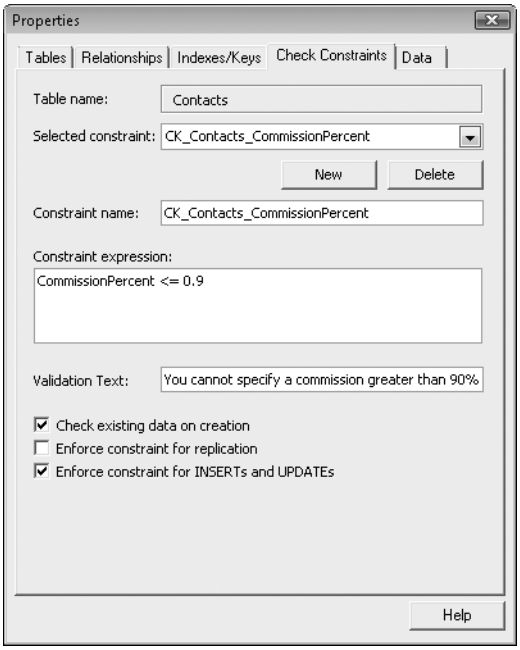


Figure 26-21 Your completed commission percent check constraint on the Contacts table should look like this.

When you are finished creating check constraints, close the Properties window, and save the table. When you save the table, the check constraints are saved as well. Now open the table in Datasheet view, and see whether the check constraint is working correctly. Enter **1** in the ContactID field, enter your last name in the LastName column, tab to the CommissionPercent column, and then try entering a commission percent of **100%**. Press the Down Arrow key to move to a different record, and you should see the error message in Figure 26-22.

Figure 26-22 SQL Server displays your custom message when validation fails on the check constraint for commission percent.

Note

In the initial release of Access 2007, Access fails to display the defined validation text when validation fails on a check constraint. Instead, Access displays the SQL Server error number and cryptic message. Microsoft is aware of this and is investigating the issue.

Creating Additional Constraint Expressions

Now that you know how to create a check constraint, let's take a look at some other types of expressions that you can use to validate your data:

- The IN keyword allows you to specify a list of values that the data must match. For example, if you want to make sure the suffix values in the Contacts table are limited to Jr., Sr., PhD., or II, use an expression like this:

  ```
  [Suffix] IN ('Jr.', 'Sr.', 'PhD.', or 'II')
  ```

- You can use the LIKE keyword to specify the formatting of data. If you want to make sure the WorkPostalCode column in Contacts is five numeric digits, use an expression like this:

  ```
  [WorkPostalCode] LIKE '[0-9][0-9][0-9][0-9][0-9]'
  ```

- You can also check multiple columns in one constraint by using expressions joined with the AND and OR operators. All constraint expressions must evaluate to true or the database won't save the row. If you want the constraint to pass when the constraint expression evaluates to false, prefix the expression with the NOT operator. For example, in the Contacts table, if you want to make sure that WorkStateOrProvince information is always supplied whenever WorkCity is supplied, use an expression like this:

  ```
  (NOT ([WorkStateOrProvince] Is Null)) OR ([WorkCity] Is Null)
  ```

When the user supplies a city (Springfield) but not a state or province (IL or MA), both NOT ([WorkStateOrProvince] Is Null) and ([WorkCity] Is Null) will be false, and the constraint will fail. Note that this constraint allows the user to enter only the state or province without a city, but this does not allow the user to enter a city without also specifying a state or province.

INSIDE OUT

Enter Multiple Constraints Carefully

Although you can join multiple constraint expressions together with AND and OR operators, it is generally a good idea to keep the expressions separate unless one column value relies on another column value for validation. Use parentheses around multiple comparison expressions and Boolean operators to ensure that your expression evaluates as you expect.

Another good use of check constraints is to purposefully take advantage of the ability to provide validation text. The current design of the Companies table requires the input of a company name in every row by setting the Allow Nulls property for CompanyName to False (no check mark). When you enter a row without a company name, you get a message from SQL Server similar to the one shown in Figure 26-23.

Figure 26-23 SQL Server returns this message when validation for the Allow Nulls property fails.

Instead of using the Allow Nulls property to require data entry, you could create a check constraint that uses an expression like this:

```
Not [CompanyName] Is Null
```

Then you can include the validation text: **You must supply a Company / Organization name**. Now your users see a much easier to understand message when they forget to enter a company name, and you will still be requiring an entry in the CompanyName column for every row added.

> **Note**
>
> If you decide to use a check constraint to require data entry in certain columns, be sure to set the Allow Nulls property for those columns. SQL Server always checks the Allow Nulls setting first. If that fails, SQL Server never checks any constraints. So, if a user attempts to save a row that incorrectly contains a Null, the user sees the cryptic SQL Server message instead of the one you specified. However, you cannot use this technique for a primary key column because SQL Server requires that no columns in a primary key contain a Null value (the Allow Nulls property must always be cleared).

Defining Relationships

Now that you have learned how to build tables in an Access project, you can start defining relationships between them. As explained in Chapter 4, defining relationships is valuable because it enforces referential integrity between the tables. It also makes building queries and forms on those tables easier. The same is true in an Access project linked to SQL Server.

Defining Relationships in Table Design View

After you've built the tables, you're ready to start defining relationships. In an Access project, you can define a relationship in Design view. All you need are two tables—one

that has a primary key and another that contains a related foreign key. Open the CompanyContacts table in Design view, right-click any column, and click Relationships. This opens the Properties window with the Relationships tab selected. Because you haven't created any relationships yet, options in the window will be unavailable and empty. To create a new relationship, click the New button, and a new relationship appears, as shown in Figure 26-24.

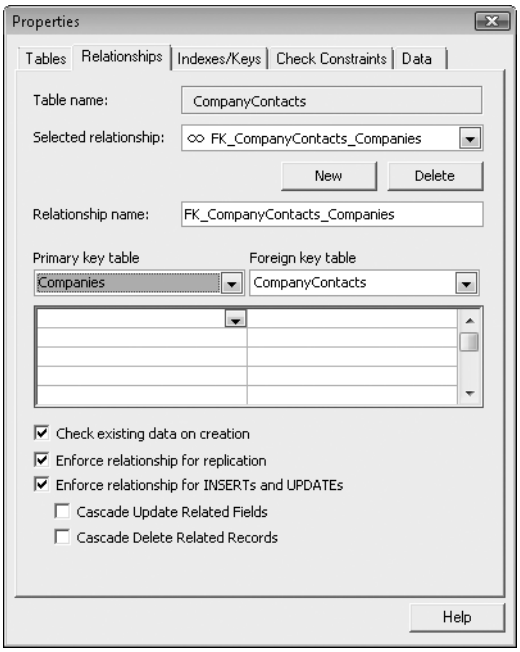


Figure 26-24 Use the Relationships tab in the Properties window to create a new relationship in the CompanyContacts table.

Relationship Name displays the name of the relationship that you are currently editing. By default, Access has named the new relationship FK_CompanyContacts_Companies. Fortunately, this is a good name for the relationship you are about to create. FK identifies this as a foreign key constraint (or relationship). The two table names identify the members of the relationship. The infinity symbol preceding the name in the Selected Relationship box indicates that you are editing or creating this relationship from the table on the many side of the relationship. If you had opened the Companies table before beginning to define this relationship, the relationship design window would show you a primary key symbol instead. Because you are actually defining a constraint on the table on the many side of the relationship, that table name appears first in the name.

Underneath Relationship Name are two lists that allow you to specify which tables are members of the relationship. By default, Access will specify the current table as the foreign key table and the first table listed alphabetically in the table list as the primary key table. Because you have created only three tables so far, the Companies table will be

listed as the primary key table. These are the two tables you want to use in the relationship, so you don't have to change the selected tables. If you have already created some other tables in your project, then the Companies table might not be listed as the primary key table. If it isn't, simply select it from the list.

Below the table listings are two lists that you can use to specify which columns will be members of this relationship. You want to create a one-to-many relationship between Companies and CompanyContacts. The column that they share in the relationship is the CompanyID column. To create the relationship, select CompanyID from the first list under both Primary Key Table and Foreign Key Table.

After you have selected the columns you want for this relationship, you need to specify a few more options before you save the table and this new relationship. When you select the Check Existing Data On Creation check box, SQL Server will examine the data in the existing tables and make sure that none of it violates the constraints of the relationship. If SQL Server finds any problems, it displays an error when you try to save the table, and the save operation will fail. Selecting this check box is a good idea for any relationship you create, because it ensures that no integrity problems exist with the current data in the tables. If you already have any data in your tables, make sure you select the Check Existing Data On Creation check box.

Selecting the Enforce Relationship For Replication check box makes sure that the foreign key constraint (the relationship) is copied to any replicated databases. For now, you can clear the Enforce Relationship For Replication check box.

The next option, Enforce Relationship For INSERTs And UPDATEs, allows you to enforce referential integrity for this relationship, ensuring that any added or updated data does not violate the constraint of the relationship. If you select the Enforce Relationship For INSERTs And UPDATEs check box, you also have the option of selecting the Cascade Update Related Fields and/or Cascade Delete Related Records check boxes. These two options allow you to control what happens to data in related rows when data in the table containing the primary key changes. If you select the Cascade Update Related Fields check box, whenever you update data in the primary key columns, SQL Server also updates any related data in the foreign key column of the related table. If you select the Cascade Delete Related Records check box, whenever you delete a row in the table containing the primary key, SQL Server deletes all related rows in the table containing the foreign key. You should carefully consider selecting Cascade Delete Related Records because selecting this check box will always delete foreign rows when you delete any rows in your primary key table. For this relationship, make sure that you leave the Enforce Relationship For INSERTs And UPDATEs check box selected. Also, select the Cascade Update Related Fields check box so that SQL Server will propagate any changes you make to the primary key to the related tables. Leave the check box for Cascade Delete Related Records cleared.

When you are finished creating the relationship between the Company table and the Company Contacts table, your Properties window should look like the one shown in Figure 26-25. If you need to add relationships, click the New button. You can also delete a relationship by choosing it from the Selected Relationship list and then clicking the Delete button. Be sure to save your table to save the changes you made to relationships.

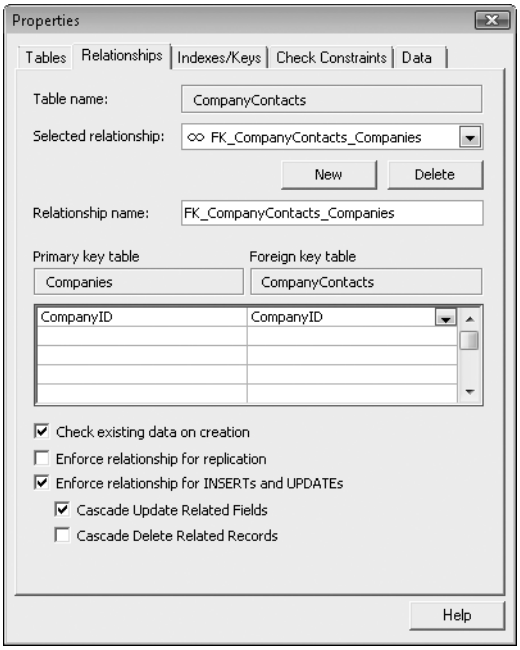


Figure 26-25 You've now completed defining a relationship between the Companies table and the CompanyContacts table.

Defining Relationships Using Database Diagrams

Database diagrams are similar to the Relationships window in an Access desktop database (.accdb), but diagrams are more versatile. You can create multiple database diagrams to organize your table relationships into different visual groups. You can also edit any portion of your table(s) directly from the Diagram window.

To demonstrate how database diagrams work, let's create a new one and use it to define the relationship between the CompanyContacts table and the Contacts table. To create a new database diagram, on the Create tab, in the Other group, click the arrow on the New Object button, and then click Diagram. Access opens a blank Diagram window and displays the Add Table dialog box, as shown in Figure 26-26.

To add a table to the Diagram window, select it from the list, and then click Add (or double-click the table name). Add Companies, CompanyContacts, and Contacts to the Diagram window. When you are finished, click the Close button.

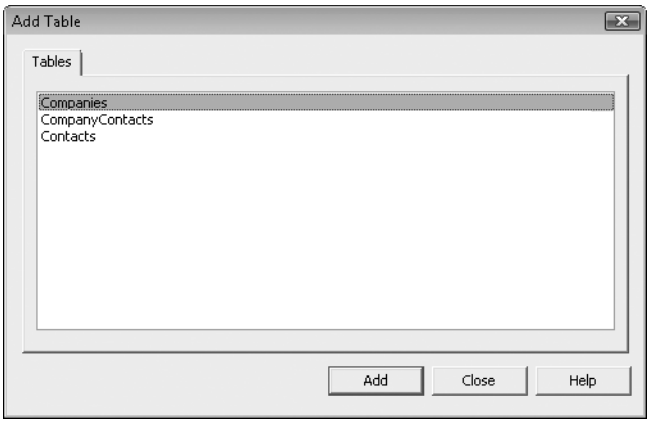


Figure 26-26 You first need to add tables to the Diagram window to begin designing relationships.

Because you have already defined the relationship between the Companies table and the CompanyContacts table, you'll see the relationship represented in the Diagram window as a line drawn from one table to the other, as shown in Figure 26-27. Note that you might need to click the header of each table and drag it within the Diagram window to position the tables as shown. The key represents the one side (or primary key side), and the infinity symbol represents the many side (or foreign key side). A one-to-one relationship would be represented with keys on each end of the line.

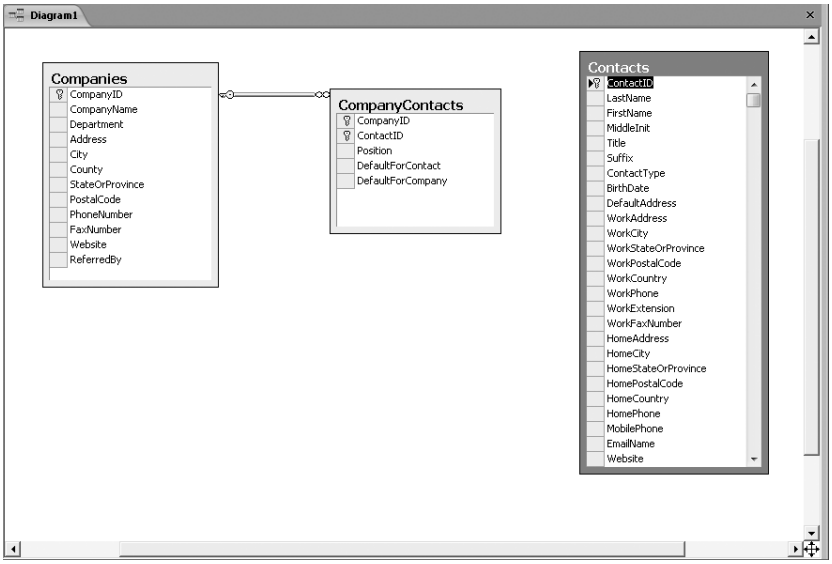


Figure 26-27 The Diagram window shows your three tables and any existing relationship lines.

You can right-click any of the tables shown in the Diagram window to see a list of options that you can use to control how the table is displayed or to edit its properties.

For example, you can select Column Properties, Column Names (the default), Keys (show only the columns that are keys), Name Only (just the table names), or Custom View. You should already be familiar with what many of these options do because they have the same functionality as the options you saw in a table's Design view. If you want to add any more tables, right-click the diagram, and click Add Table. You can remove tables from the diagram by right-clicking the table header and selecting Hide Table. (Hiding the table does not delete it.)

If you open a table's properties (by clicking Properties on the shortcut menu or by double-clicking any of the table headers), you will notice the addition of the Tables, Columns, and Lookup tabs. You can select the table you want to edit on the Tables tab and set certain table properties. After you select a table, you can choose a column to edit on the Columns tab. The options you see on the Lookup tab are the same as on the Lookup tab you see in the bottom half of a table's Design view. By using these tabs and manipulating how much of the table is displayed in the diagram, you can easily design or redesign any part of your tables directly from the Diagram window.

Now let's define the relationship between CompanyContacts and Contacts. In the Diagram window, you can do this simply by dragging the primary key from one table and dropping it onto another. To do this, click the selector for the ContactID column name in Contacts, and drag and drop the column onto the CompanyContacts table. When you release the mouse button, Access displays the dialog box shown in Figure 26-28 so that you can define the relationship.

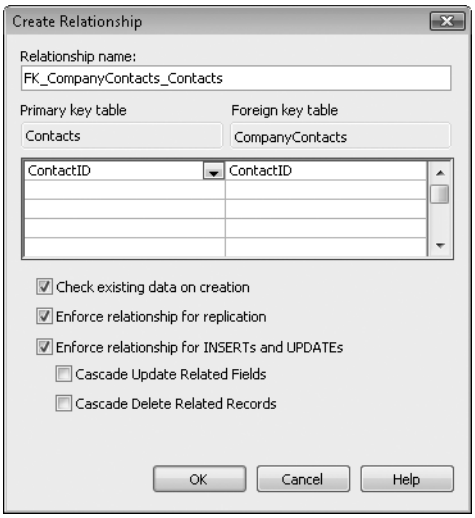


Figure 26-28 You can use the Create Relationship dialog box to define a relationship from the Diagram window.

This dialog box is very similar to what you see on the Relationships tab of a table's properties. Relationship Name displays the default name for this relationship. If you want, you can type a different name. If the keys in each table share the same name, Access automatically picks them out for you. Otherwise, you can choose the primary

and foreign keys from the lists under each table name. You can also specify the relationship criteria, as described earlier. For this relationship, make sure that the ContactID column is the only column specified for both tables. Leave the Check Existing Data On Creation and Enforce Relationship For INSERTs And UPDATEs check boxes selected. Also select the Cascade Update Related Fields check box. Make sure all other check boxes are cleared. When you are done, click the OK button to create the relationship. The Diagram window will now show the relationship between the CompanyContacts table and the Contacts table, as shown in Figure 26-29. Note that the asterisk on Contacts and CompanyContacts indicates that an update is pending for both tables.

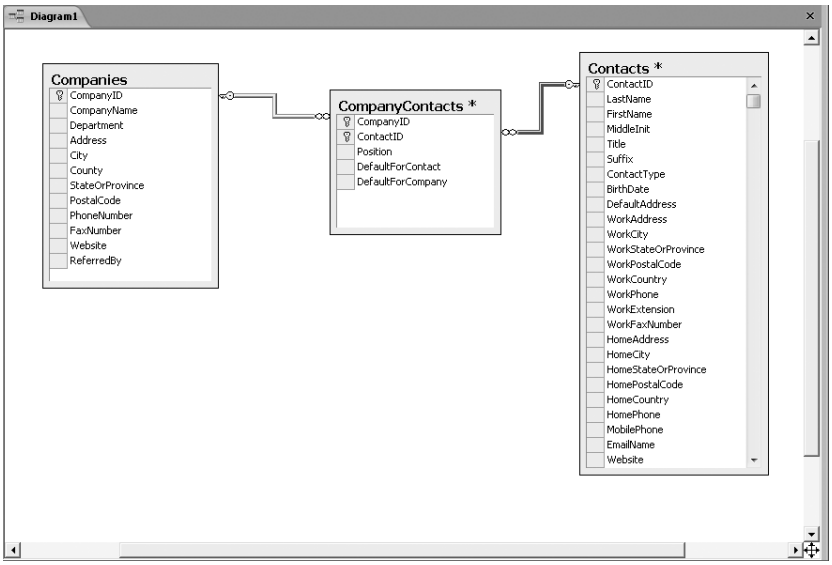


Figure 26-29 The Diagram window now displays the two relationships you created.

You have now created two relationships that represent the many-to-many relationship between the Companies table and the Contacts table. The relationships you create in a Diagram window won't be saved until you save the diagram. When you close the Diagram window or click the Save button on the Quick Access Toolbar, Access prompts you to provide a name for the diagram. (If you close the Diagram window without first saving your changes, Access asks you whether you want to save the diagram.) Go ahead and name this diagram **dgmContactTracking**. Access also prompts you to save the pending changes to the two tables. Click Yes to save the new relationship.

Setting Table Design Options

Now that you understand the basic mechanics of defining tables and relationships in your Access project, let's take a look at several options that allow you to customize how you design your tables. To view the database options, click the Microsoft Office Button, and then click Access Options.

You can find the first options that affect table design under the General heading in the Advanced category, as shown in Figure 26-30. As we mentioned in Chapter 4, we highly recommend that you select the All Databases check box under Use Four-Digit Year Formatting. When you select this check box, Access displays all year values in date/time formats with four digits instead of two. This is important because when you see a value in two-digit date format, such as 15 MAR 12, you won't be able to easily tell whether this is March 15, 1912, or March 15, 2012. Although you can affect the display of some formats in your Regional And Language Options in Windows Control Panel, you won't affect them all unless you set four-digit formatting in Access.

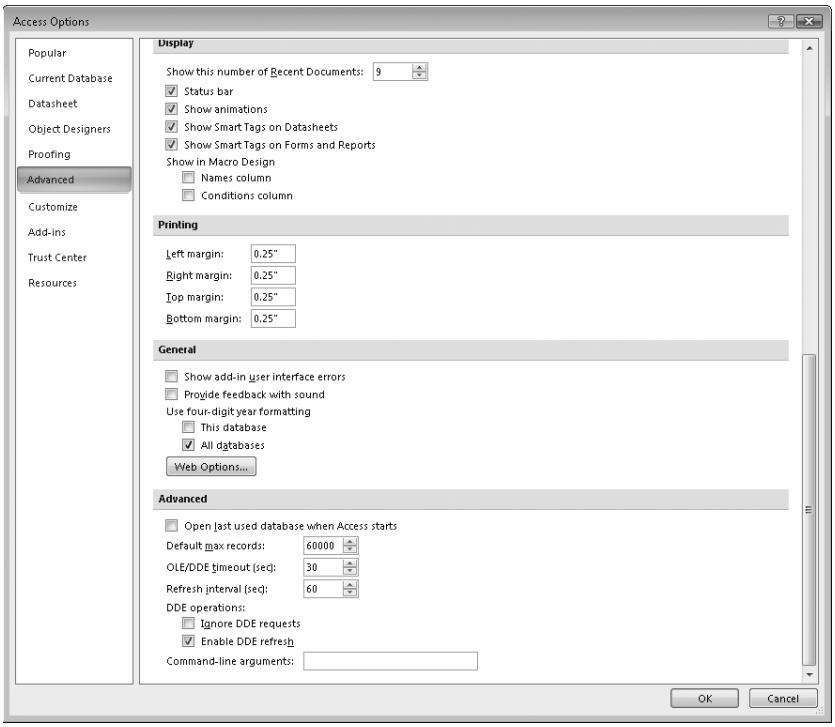


Figure 26-30 You can change settings that affect table design in the General section of the Advanced category in the Access Options dialog box.

Under Use Four-Digit Year Formatting, you have two options. If you select the This Database check box, the setting creates a property in the database you currently have open and affects only that database. If you select the All Databases check box, the setting creates an entry in your Windows registry that affects all databases that you open on your computer. We recommend that you select the All Databases check box.

The next category that contains useful settings that affect table design is the Object Designers category. Select that category to see the settings shown in Figure 26-31.

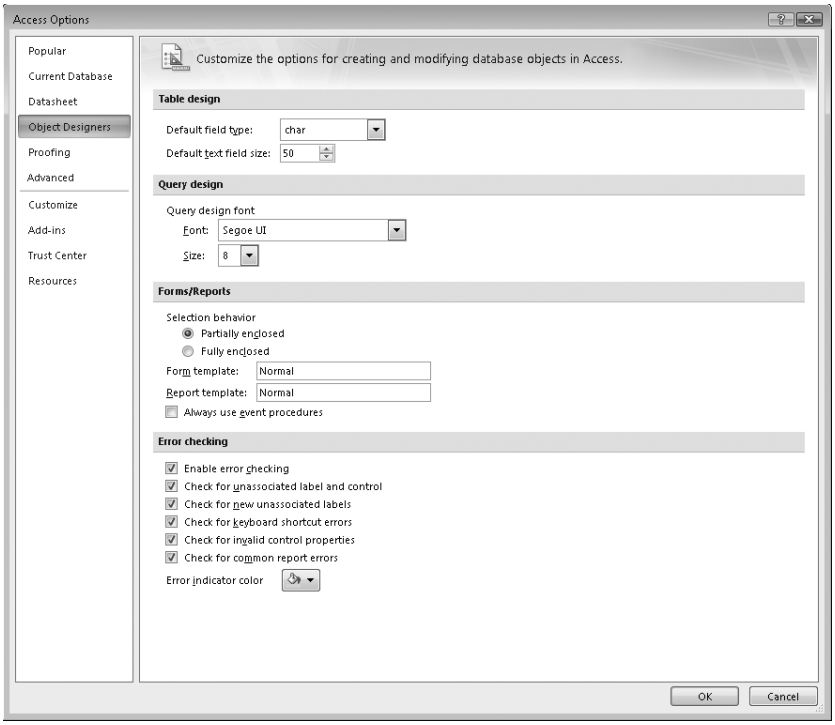


Figure 26-31 You can also change settings that affect table design in the Object Designers category in the Access Options dialog box.

Under the Table Design heading of this category, you can set the default field sizes for text fields. When you choose a data type that supports text (char, nchar, varchar, nvarchar), Access automatically fills in the length you choose. For Default Field Type, you can choose the field type that Access selects when you type a new column name in a table's Design view and then tab to the Data Type column.

Now that you've started to get comfortable with creating an Access project and tables, you can read the next chapter to learn how to create the different types of queries (functions, stored procedures, and views) supported by Access projects.