MICROSOFT® ACCESS® TUTORIAL

APPENDIX M

Preview

After a database has been designed properly it must be implemented and the applications that make the database useful to the end users must be developed. Microsoft Access is a great tool for prototyping a database's implementation and its applications. Therefore, we will show you how to:

- Create the database.
- Create and modify tables.
- Enter data into the tables.
- Import tables.
- Define the relationship(s) between the tables.
- Create queries.
- Create forms.
- Create reports.
- Link application components with macros.

This tutorial[1] assumes that you know how to perform basic operations in the Microsoft Windows environment. Therefore, you should know what a folder is, how to maximize or minimize a folder, how to create a folder, how to select a file, how you maximize and minimize windows, what clicking and double-clicking indicate, how you create a folder, how you drag, how to use drag and drop, how you save a file, and so on.

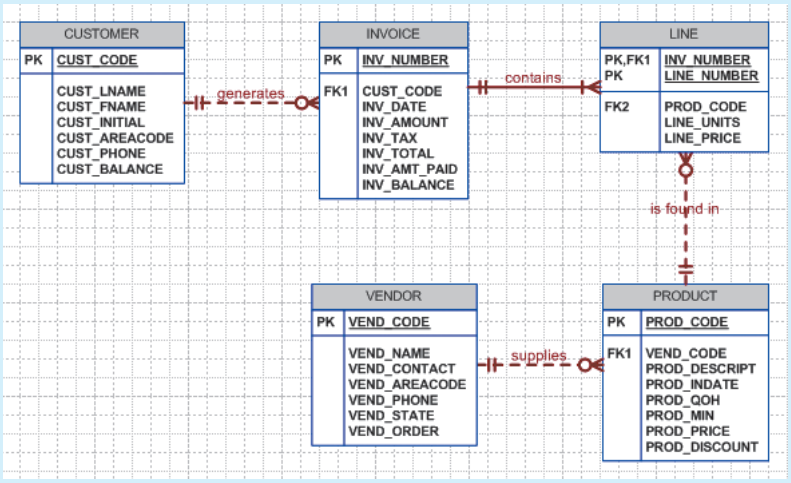
[1]This tutorial is intended as a supplement to *Database Systems: Design, Implementation, and Management, 10th ed.* Many thanks to Christian Couch for his extensive help with the updates to this tutorial.

Contents

1.1 Create the Database

If you have studied our ***Database Systems: Design, Implementation, and Management*** text, you already know the important ground rule: The database design is the crucial first step in the journey that lets you successfully implement and manage a database. The database design shown in Figure M.1 in the form of an Entity Relationship Diagram (ERD) will be the basis for this tutorial.

FIGURE M.1 The SaleCo database ERD



SOURCE: Course Technology/Cengage Learning

NOTE

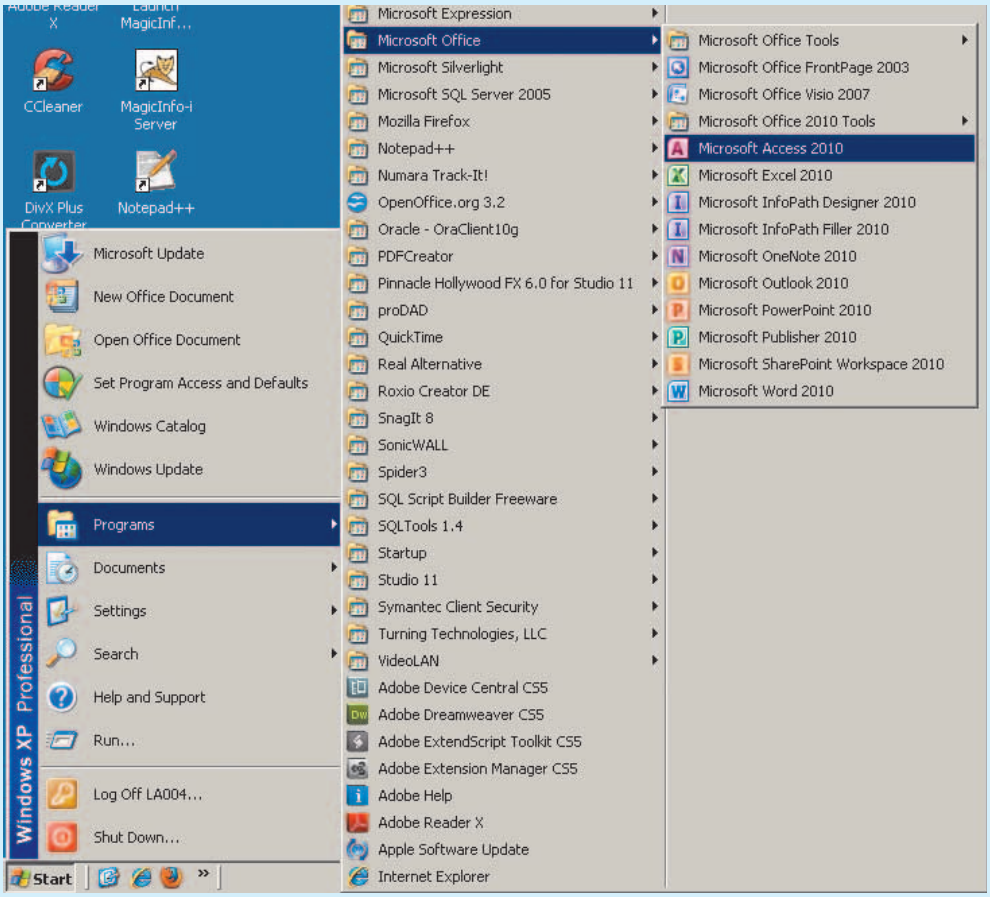
If necessary, review the text's Chapters 4 through 6 to ensure that you have a sound basis for database design work. You can study Appendix A, "Designing Databases with Visio Professional: A Tutorial," if you want to create the database design shown in Figure M.1.

Given the database design shown in Figure M.1, you are ready to implement it in Microsoft Access. The first step is to start the Microsoft Access DBMS software, which is part of the Microsoft Office *Professional* suite. Use the same routine you use for all Microsoft Office components. In other words, to start Microsoft Access, follow the sequence **Start/All Programs/Microsoft Office/Microsoft Access**. The *sequence* is the same for all MS Office versions. However, you will see slight variations in the labeling of the MS Office components. For example, Figure M.2 shows Office 2007 labels. (And, naturally, your screen is likely to differ—in detail—from the one you see in Figure M.2. After all, you may be running a different version of Windows and you may have a different version of the Microsoft Office suite. Nevertheless, the basics remain the same.)

After you have completed the just-described sequence, you will see the screen in Figure M.3. (Only a small portion of the screen is shown.)

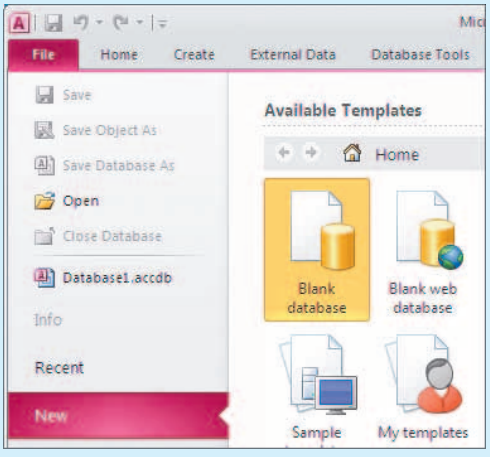
After you select the **New...** option shown in Figure M.3, the screen will change to show Figure M.4. Place the cursor over the **Blank database...** option.

FIGURE M.2 Starting Microsoft Access



SOURCE: Course Technology/Cengage Learning

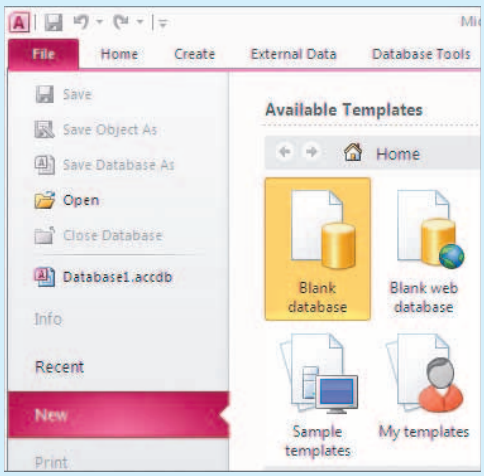
FIGURE M.3 Creating a new database



SOURCE: Course Technology/Cengage Learning

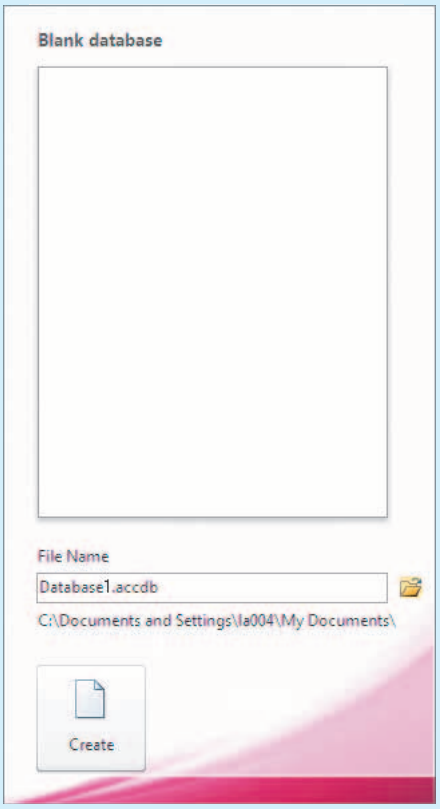
When you select the **Blank database...** option shown in Figure M.4, Figure M.5 will appear. Note that the default folder is **My Documents** and that the default database name is **Database1.accdb**.

FIGURE M.4 Selecting a blank template



SOURCE: Course Technology/Cengage Learning

FIGURE M.5 Default folder and database name

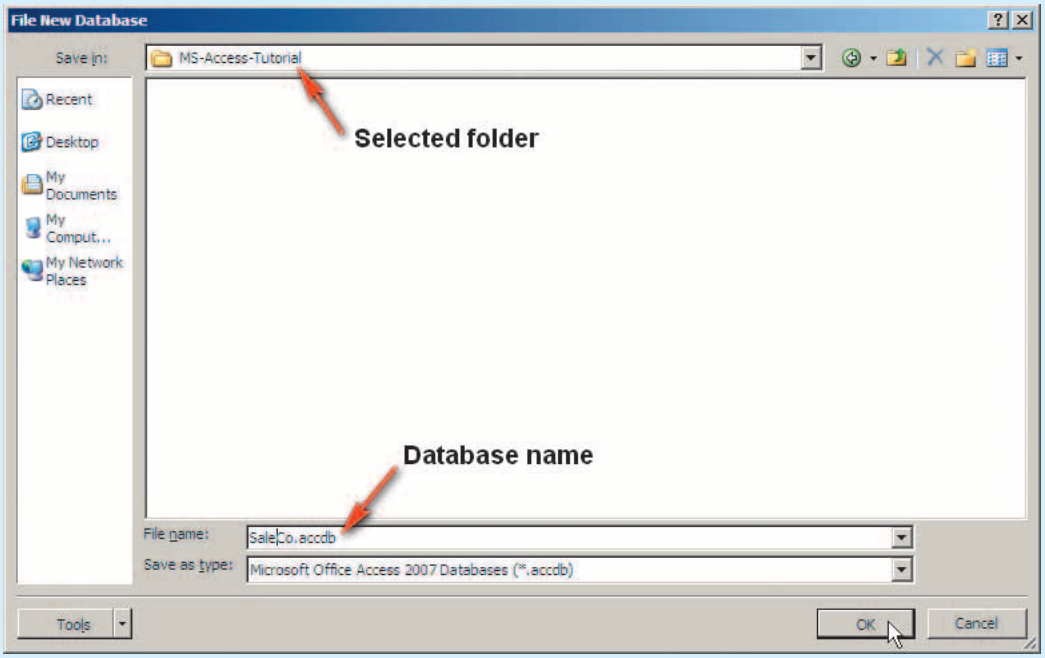


SOURCE: Course Technology/Cengage Learning

Select the folder in which you want to store the database. In the example you see in Figure M.6, the folder is named **MS-Access-Tutorial** and the database name is **SaleCo**.

After selecting the folder and database name as shown in Figure M.6, click the **OK** button, then click the Create button to generate the **SaleCo** database you see in Figure M.7.

FIGURE M.6 Select the database folder and name

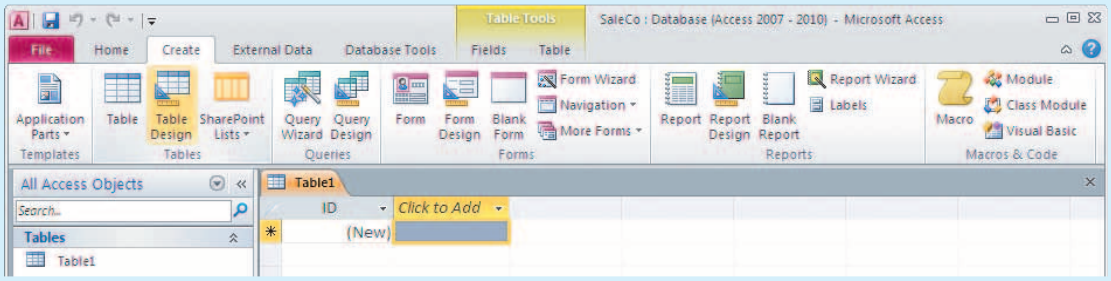


SOURCE: Course Technology/Cengage Learning

NOTE

The **SaleCo** database is also found on the textbook Premium Website for the text. However, *that* database includes all the components you will learn about in this tutorial. If you want to copy the **SaleCo** database from the Website, make sure you place it in a different folder to avoid over-writing the database components that you will create in this tutorial.

FIGURE M.7 The SaleCo database

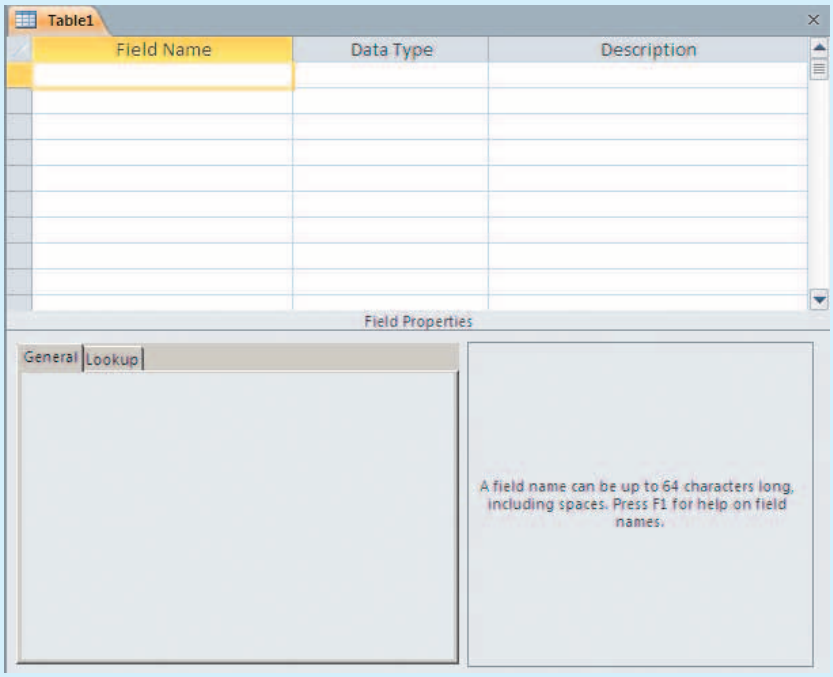


SOURCE: Course Technology/Cengage Learning

1.2 Creating the Table Structures

A table should automatically be created when you open the database. If not, go ahead and create the first table structure by clicking the **Table Design** button highlighted in Figure M.7. This will generate the table structure you see in Figure M.8. (You can click the Maximize button to maximize the screen or you can simply drag the sides to suit your needs.)

FIGURE M.8 The initial table in Design view



SOURCE: Course Technology/Cengage Learning

You are now ready to type in the first attribute name. (See Figure M.1 again to see what attributes are included in the CUSTOMER table.) Figure M.9 shows the entry of the CUST_CODE attribute.

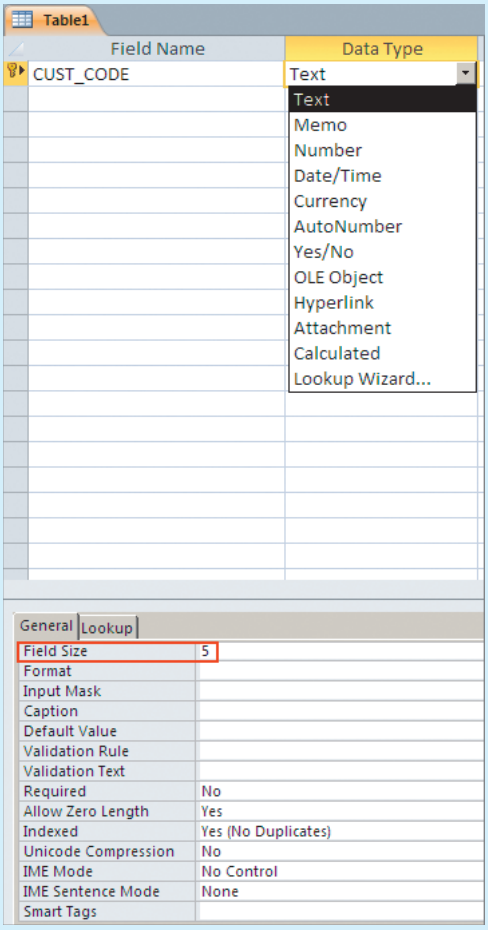
The CUST_CODE values will all consist of a five-character text string, so the **Field Properties** box default **Field Size** of 255 should be reset to 5. (Just mark the "255" and type in the value "5" to get the job done.)

We suggest that you save your work frequently, so go ahead and save the CUSTOMER table structure. You will be prompted to enter a table name, so type the CUSTOMER name in the **Save As** dialog box, as shown in Figure M.10.

However, note that your save routine displays the message box shown in Figure M.11.

If you select the **Yes** button in the PK reminder shown in Figure M.11, MS Access will automatically create a PK attribute for you … and that is not what you want. (Remember, you want the CUST_CODE to be your PK.) If you select **Cancel**, you will be returned to the table design format without first saving the table. Since you want to save the table, select **No** to ensure that the table will be saved, albeit without a PK. (We will show you how to set the PK later.)

FIGURE M.9 The initial CUSTOMER table field entry



SOURCE: Course Technology/Cengage Learning

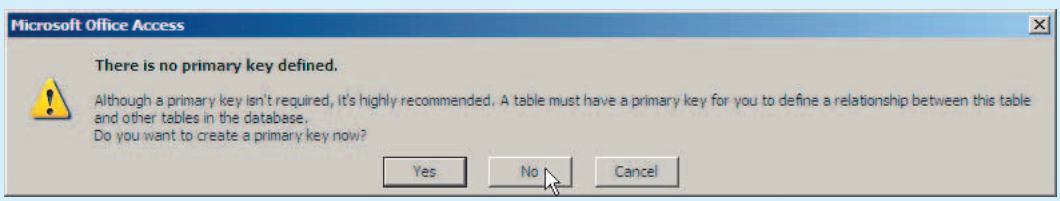
FIGURE M.10 Save the CUSTOMER table

SOURCE: Course Technology/Cengage Learning

After you click the **No** button, MS Access will return you to the table as you see in Figure M.9, and the new CUSTOMER table will be listed as shown in Figure M.12.

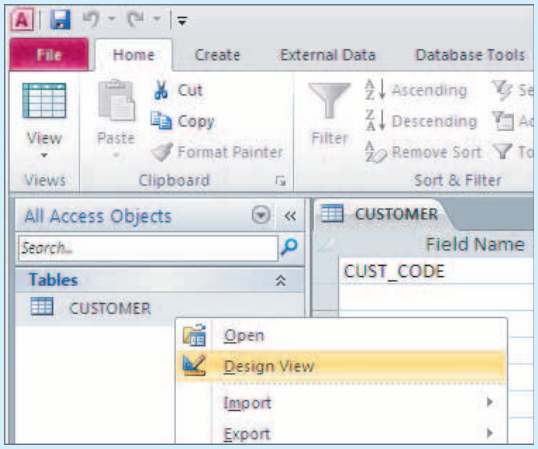
To continue working on the CUSTOMER table, right-click the CUSTOMER table name and then select the **Design View** option to return to the table field entry screen. If you take a look at Figure M.13, you'll see that we have entered a CUST_CODE description and that we have decreased the **Field Size** to 5.

FIGURE M.11 Primary key reminder



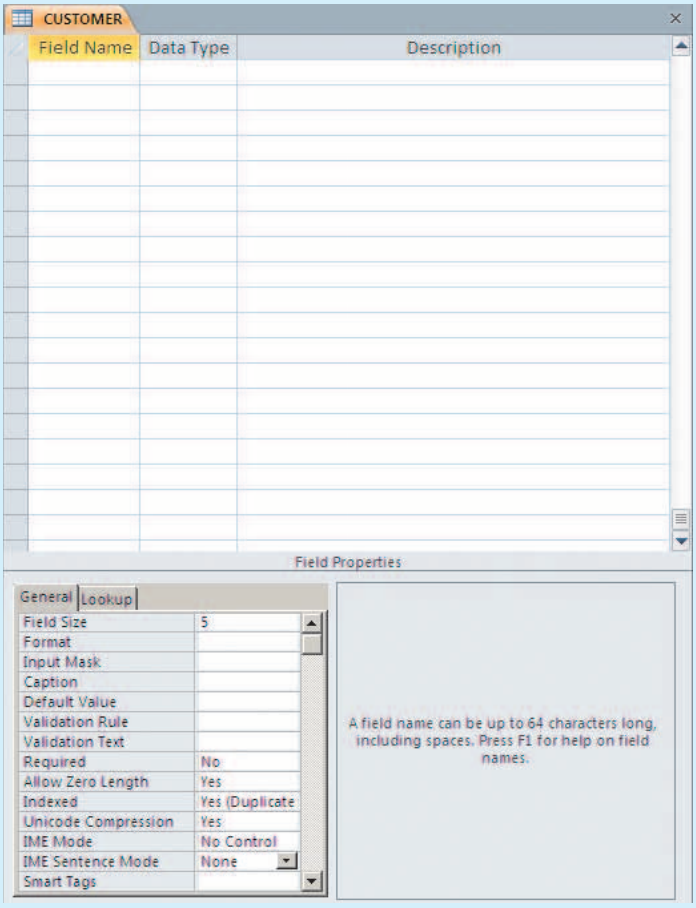
SOURCE: Course Technology/Cengage Learning

FIGURE M.12 The saved CUSTOMER table



SOURCE: Course Technology/Cengage Learning

FIGURE M.13 Additional CUSTOMER table field entries

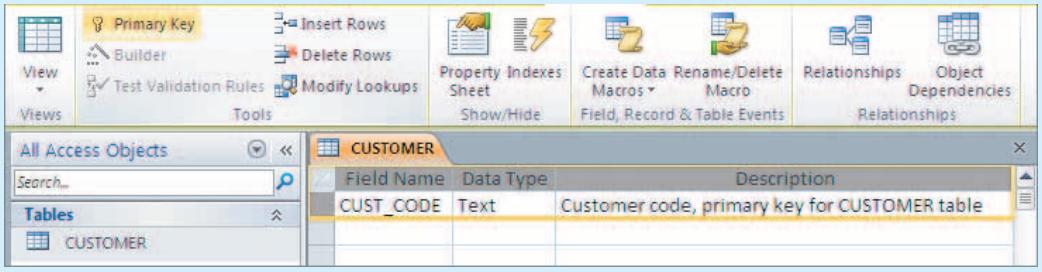


SOURCE: Course Technology/Cengage Learning

As you examine Figure M.13, note that the **Field Properties** box entry for the CUST_CODE indicate that the **Indexed** default value is set to **Yes (Duplicates OK)**. However, this will be corrected once you define CUST_CODE to be the PK for this table. That job is done in two steps:

1. Click in the narrow column just to the left of the CUST_CODE attribute in the **Field Name** column. Note that this action inverts the screen for the first field name row.
2. Click the key symbol on the button bar along the top of the screen to set the PK.

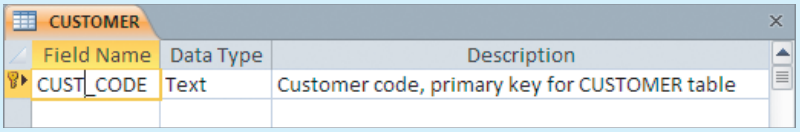
FIGURE M.14 Setting the primary key



SOURCE: Course Technology/Cengage Learning

The completion of step 2 generates the screen you see in Figure M.15. Note that the narrow column to the left of the **Field Name** column now shows the key symbol to indicate that the CUST_CODE attribute is now the PK. (We suggest that you save the table again.)

FIGURE M.15 Primary key is set



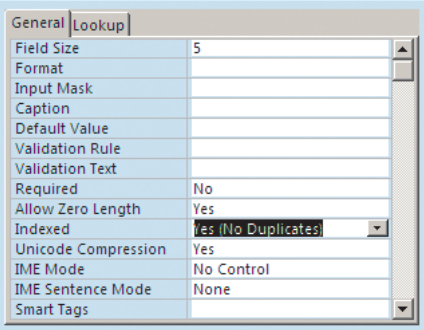
SOURCE: Course Technology/Cengage Learning

If you look at the CUST_CODE **Field Properties** box again, you will see—check Figure M.16—that the CUST_CODE **Indexed** property has been reset to **Yes (No Duplicates)**. Note that this new property enforces entity integrity, because no duplicate values will be permitted.

Using the procedures you have just learned, go ahead and create the remaining attributes for the **SaleCo** database's CUSTOMER table. (Use the ERD in Figure M.1 as your guide.) Note that the CUST_BALANCE is a **currency** value in Figure M.17. Make sure that you use a **Text** format for all of the remaining attributes and that you limit the field lengths for CUST_LNAME, CUST_FNAME, CUST_INITIAL, CUST_AREACODE, and CUST_PHONE to 20, 20, 1, 3, and 8, respectively. (Phone numbers will be entered with the format 999-9999.) Given the self-describing field names, there is little need to continue the entry of more detailed descriptions. For example, the CUST_LNAME is clearly a customer's last name, so no further description is necessary.

Make sure that you save the table again. You are now ready to enter some data. Incidentally, note that the CUST_BALANCE default value in Figure M.17 is 0. Therefore, the CUST_BALANCE (currency) value will be shown as 0.00 until you change it.)

FIGURE M.16 The CUST_CODE field properties

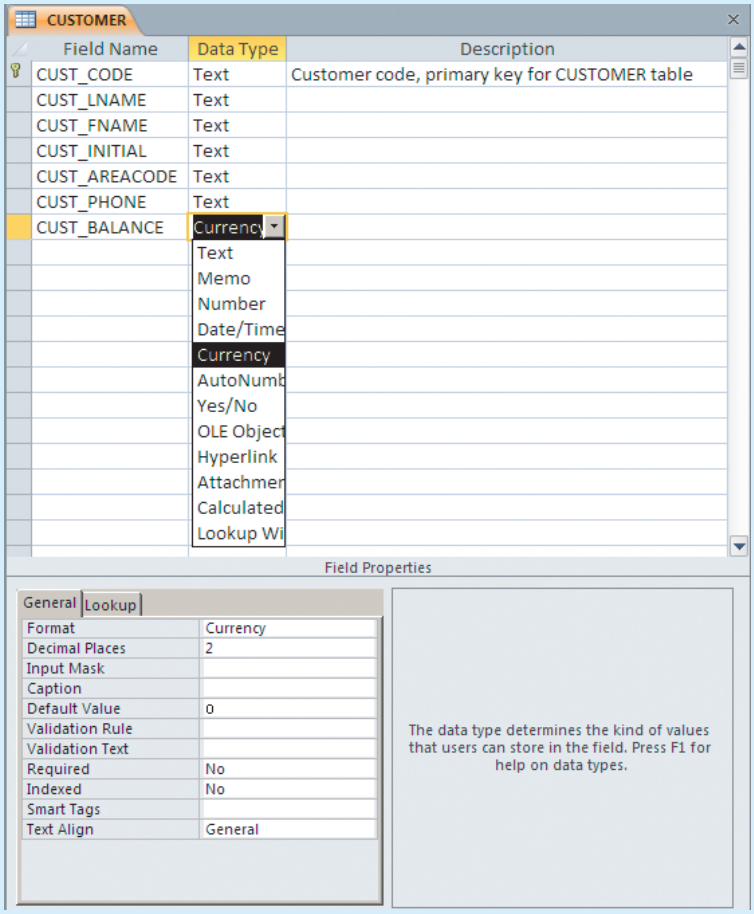


SOURCE: Course Technology/Cengage Learning

1.2.1 Data Entry

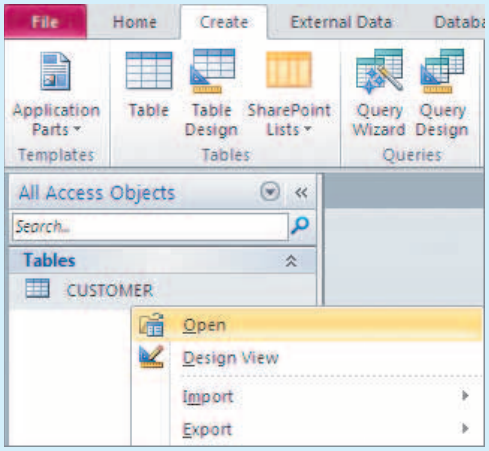
Close the CUSTOMER table's design view to return to the table options screen, then right-click the CUSTOMER table and click the **Open** table option shown in Figure M.18 to generate Figure M.19. (You can also double-click the CUSTOMER table to see Figure M.19.) Once the table is open, go ahead and enter the first customer's data as shown in Figure M.20.

FIGURE M.17 The completed CUSTOMER table



SOURCE: Course Technology/Cengage Learning

FIGURE M.18 Opening the CUSTOMER table for data entry



SOURCE: Course Technology/Cengage Learning

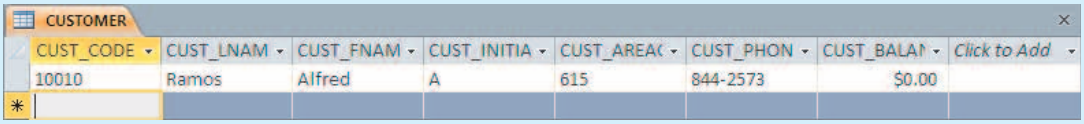
FIGURE M.19 The CUSTOMER Data Select View

SOURCE: Course Technology/Cengage Learning

NOTE

Before you enter any data, you should remember that the "parent" table entries always precede the "child" data entries. In other words, in a 1:M relationship, the "1" side's data entry precedes that of the "M" side data entry. For example, if you try to make a data entry into the INVOICE table for a customer who does not yet exist, you will commit a data integrity violation. You will learn how to create relationships between tables in Section 1.3—and you will then discover that you will have the option to have MS Access enforce referential integrity.

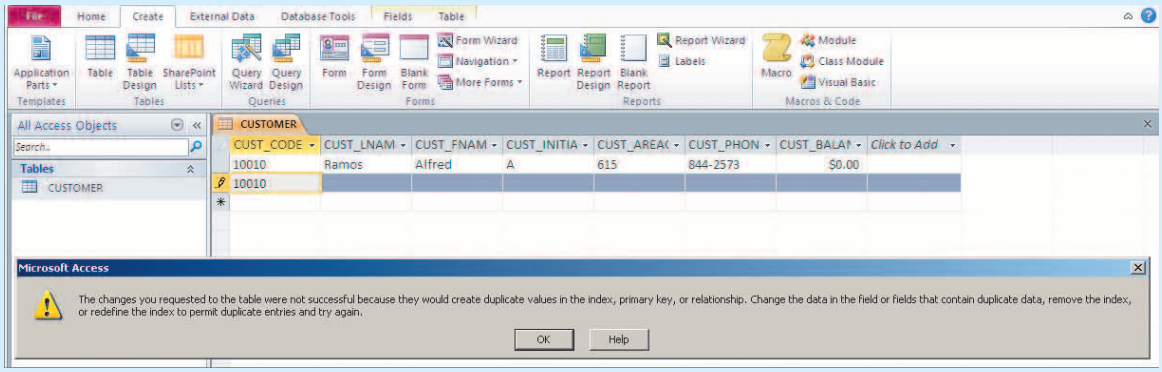
FIGURE M.20 Entering the first CUSTOMER record



SOURCE: Course Technology/Cengage Learning

Suppose you now try to enter the second record. After accepting the 0.00 CUST_BALANCE value, you tap the **Enter** key to move the cursor to the second record. Now type in the same 10010 CUST_CODE value you used for the first record … and then try to close the CUSTOMER table. Your reward will be the error message shown in Figure M.21, because you violated entity integrity requirements.

FIGURE M.21 **Enforcement of entity integrity**



SOURCE: Course Technology/Cengage Learning

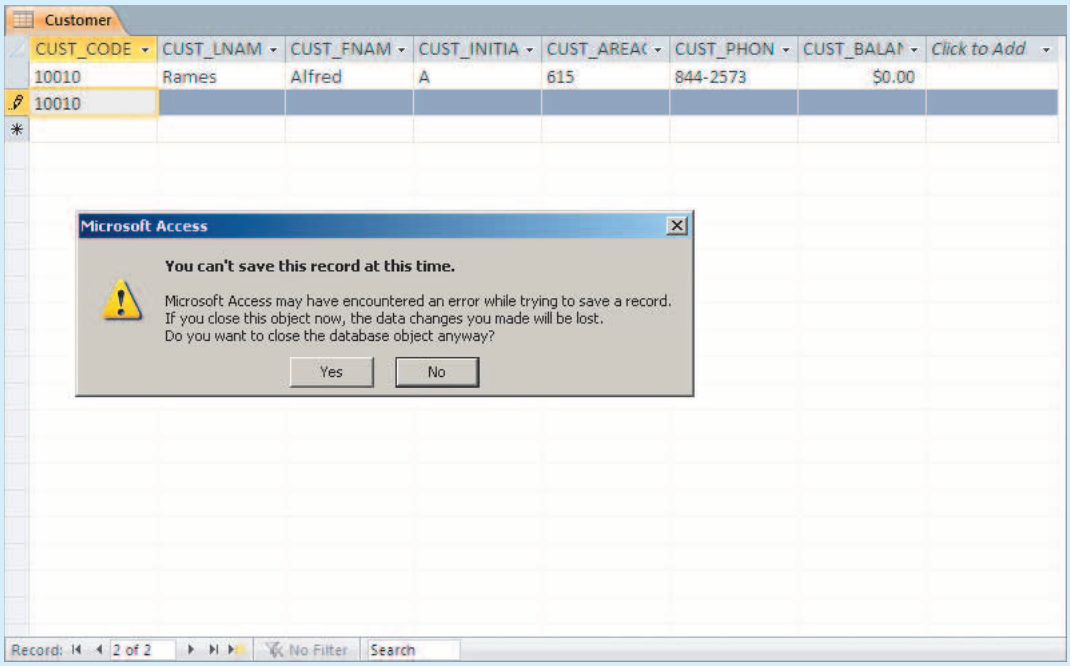
Click **OK** to acknowledge the error message. This action will generate the result you see in Figure M.22. (In this figure, the MS Access window selection is "Restore down" to ensure that all the components you see in Figure M.22 can be shown. If you had selected the "Maximize window" option, you would be limited to seeing the table and the dialog box.)

If you click the **Yes** button shown in Figure M.22, Access will return you to the table option screen you saw in Figure M.18. In this case, the second record will not be saved and the table simply retains the first record "as is." If you click the **No** button, the cursor returns to the data entry screen. This action lets you make the necessary correction by typing in a CUST_CODE value other than 10010.

The remaining records are handled just as the first record was. When you are done, the CUSTOMER table contents should match Figure M.23.

You are now ready to create the remaining tables (INVOICE, LINE, PRODUCT, and VENDOR) and their contents. However, rather than wasting your time on such repetitive tasks, we will show you how to import these items from a database you already have. That database, named **Ch07_SaleCo**, is used in the text's Chapter 7, Introduction to Structured Query Language (SQL)." This database is available online, so go ahead and download it now into a folder of your choice. You will learn how to import database components from the **Ch07_SaleCo** database to your **SaleCo** database in the next section.

FIGURE M.22 Record not saved warning



SOURCE: Course Technology/Cengage Learning

FIGURE M.23 Completed CUSTOMER table entries

Customer

CUST_CODE	CUST_LNAME	CUST_FNAME	CUST_INITIAL	CUST_AREACODE	CUST_PHONE	CUST_BALANCE	Add New Field
10010	Rames	Alfred	A	615	844-2573	0.00	
10011	Dunne	Leona	K	713	894-1238	0.00	
10012	Smith	Kathy	W	615	894-2285	165.52	
10013	Olowski	Paul	F	615	894-2180	536.75	
10014	Orlando	Myron		615	222-1672	136.60	
10015	O'Brian	Amy	B	713	442-3381	0.00	
10016	Brown	James	G	615	297-1228	221.19	
10017	Williams	George		615	290-2556	768.93	
10018	Farriss	Anne	G	713	382-7185	216.55	
10019	Smith	Olette	K	615	297-3809	0.00	

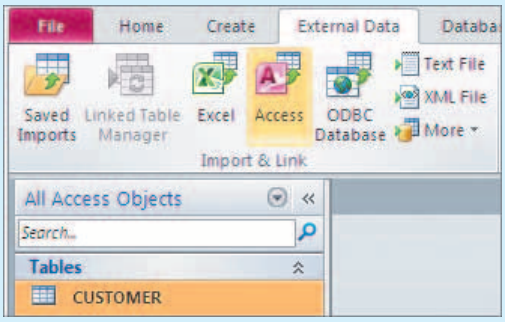
SOURCE: Course Technology/Cengage Learning

1.3 Importing Components from other MS Access Databases

To import a table—or, for that matter, any other Access object—start by clicking the **Access** button found in the **External Data** tab as shown in Figure M.24.

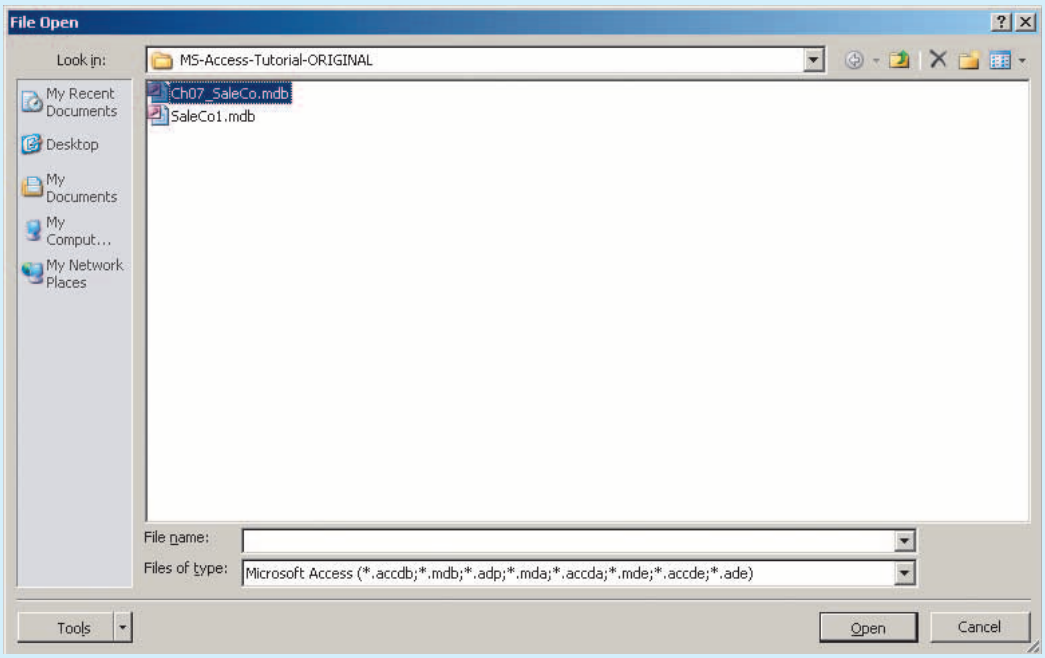
Next, select the data source. In this example, the **Ch07_SaleCo** database stored in the student folder will be used, so move to the folder that contains this database and select the **Ch07_SaleCo** database as shown in Figure M.25.

FIGURE M.24 Starting the Import sequence



SOURCE: Course Technology/Cengage Learning

FIGURE M.25 Selecting the data source

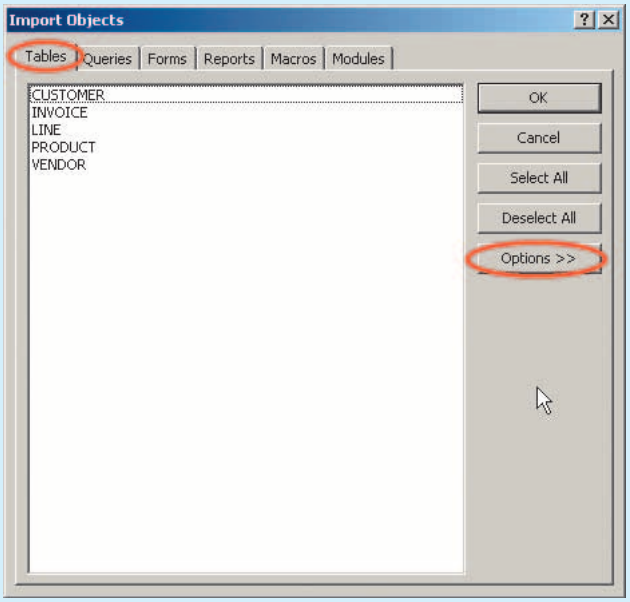


SOURCE: Course Technology/Cengage Learning

After selecting **Ch07_SaleCo.mdb** you can click the **Open** button to open the database.

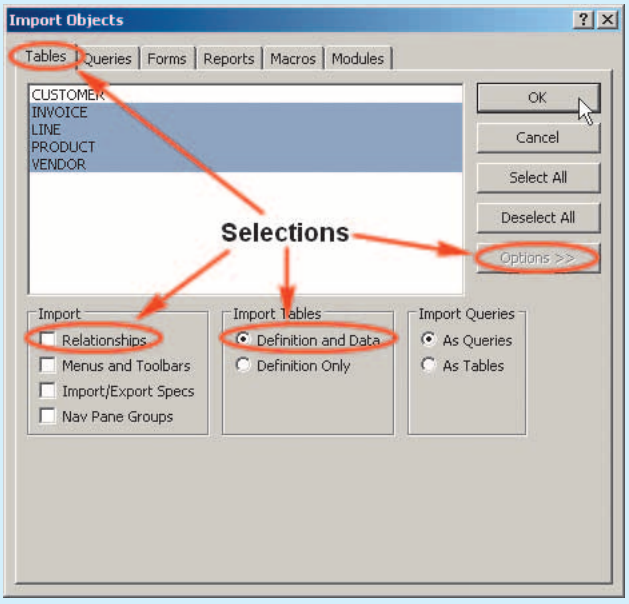
Now select the objects you want to import and specify in what format you want them imported. Figure M.26 shows that **Tables** are to be imported. Note that the **Options >>** have been selected to give you a chance to make the selections shown in Figure M.27.

FIGURE M.26 Selecting the Import options



SOURCE: Course Technology/Cengage Learning

FIGURE M.27 The selected options



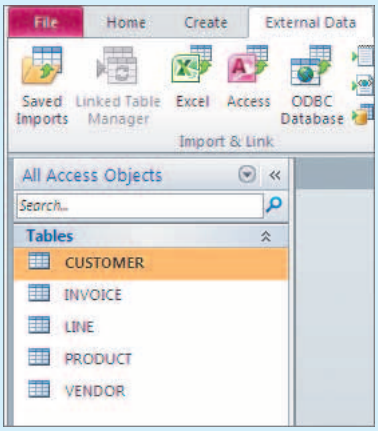
SOURCE: Course Technology/Cengage Learning

As you examine Figure M.27, note the following selections:

- The INVOICE, LINE, PRODUCT and VENDOR tables have been marked.
- The **Relationships** option has been deselected—no check mark is shown in the box. (If the box contains a check mark, click it to remove the check mark. The reason for not including the relationships is that you should learn how to create such relationships later.)
- The **Definition and Data** option has been selected to ensure that the table structure and all the data contained in each selected table will be imported.

After you have made the selections shown in Figure M.27, click **OK** to import the tables. The results of this action are shown in Figure M.28.

FIGURE M.28 The imported tables

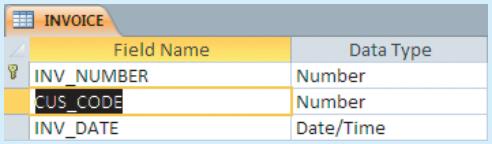


SOURCE: Course Technology/Cengage Learning

1.3.1 Editing the Imported Tables

As you can tell by looking at Figure M.28, the imported tables are all there … but you will discover that their attribute names do not precisely conform to the ones used in Figure M.1's ERD. For example, note that in Figure M.29 the INVOICE table's Foreign Key—the field that will later connect this INVOICE table to the CUSTOMER table—is CUS_CODE, rather than CUST_CODE. Therefore, you should edit the attribute names in each of the tables to be consistent.

FIGURE M.29 Imported INVOICE table structure

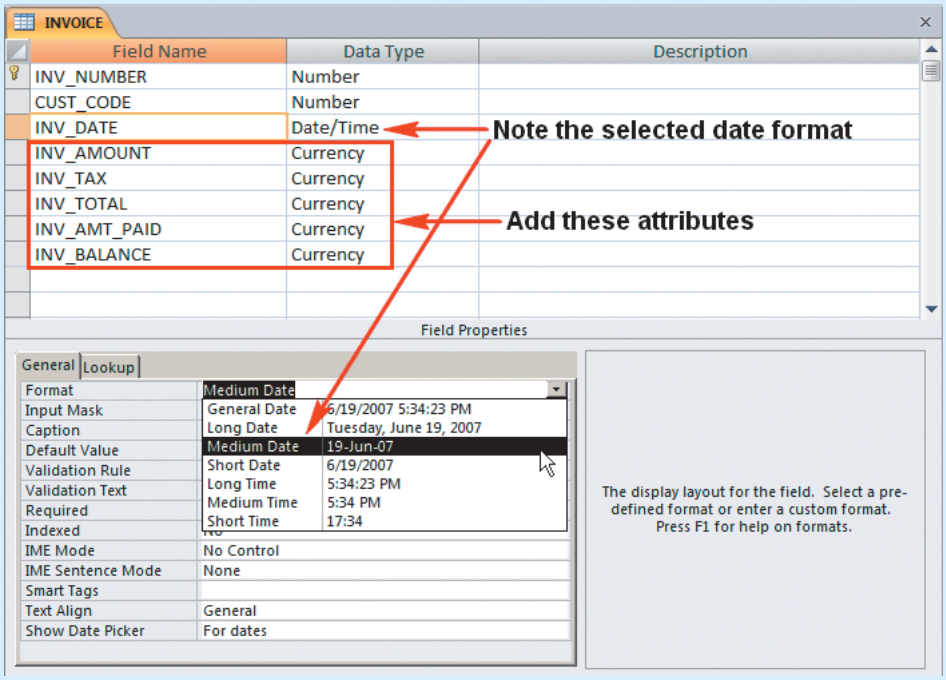


SOURCE: Course Technology/Cengage Learning

Make the changes illustrated in Figure M.30. Note the addition of attributes and their field types. (All changes in the table structures are made via the Design View.)

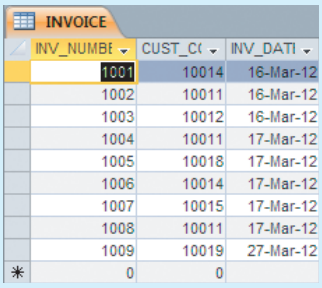
When you open the INVOICE table to see its contents, you will see that the new attributes are there, but they do not (yet) contain values. (See Figure M.31.)

FIGURE M.30 The edited INVOICE table structure



SOURCE: Course Technology/Cengage Learning

FIGURE M.31 The INVOICE table contents



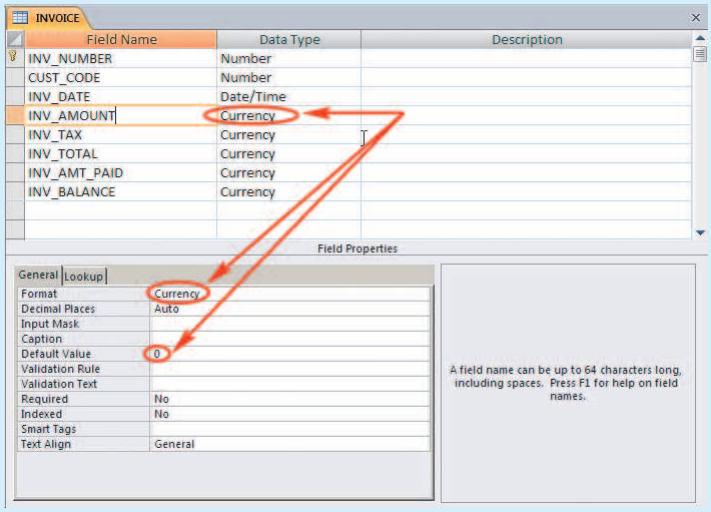
SOURCE: Course Technology/Cengage Learning

If you wonder why the new attributes in Figure M.31's INVOICE table have $0.00 default values, open the INVOICE table in its design format and note the selection of the currency format and default value in the **Field Properties** box shown in Figure M.32.

While you have the INVOICE table open, you can change the spacing between the attributes. When you put the cursor on a boundary between attributes, note that the cursor changes to a two-side arrow as shown in Figure M.33. You can drag the boundary to increase or decrease the column display width, or you can double-click any column boundary to change the column display width to whatever size is required to show the column values.

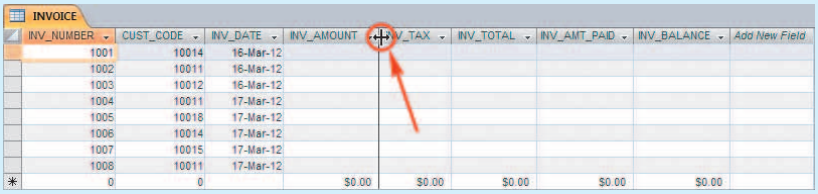
You can also change the presentation font by selecting the **Font** option shown in Figure M.34.

FIGURE M.32 INVOICE table field properties



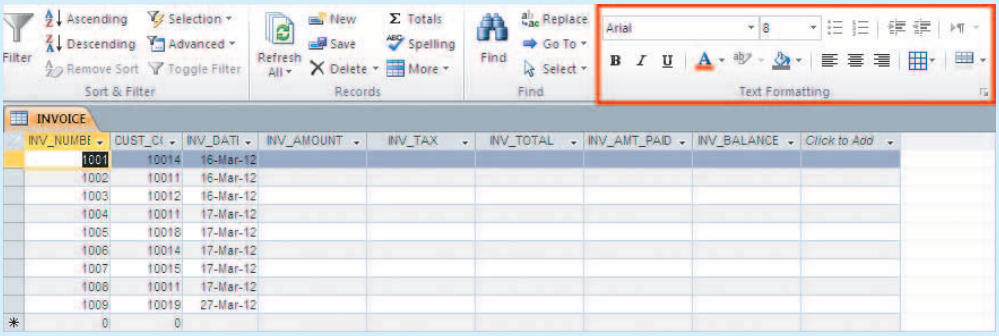
SOURCE: Course Technology/Cengage Learning

FIGURE M.33 Changing the column spacing



SOURCE: Course Technology/Cengage Learning

FIGURE M.34 Selecting the font format

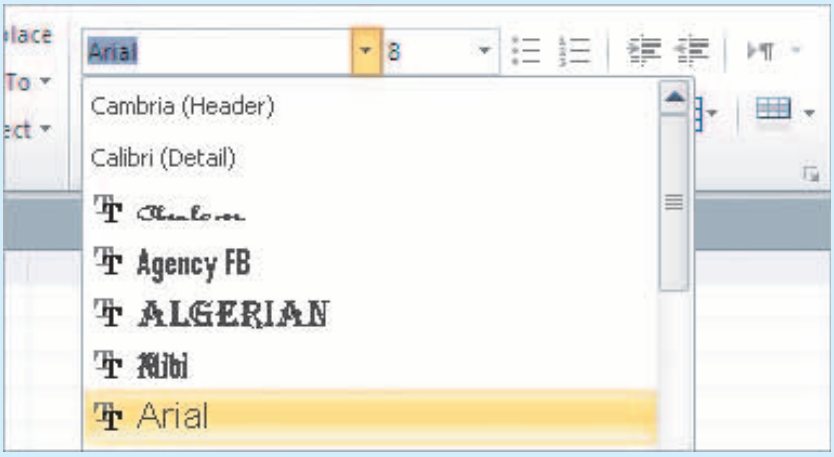


SOURCE: Course Technology/Cengage Learning

Note the selection of the font, style, and size in Figure M.35.

Go ahead and experiment with different formatting options. A good rule to follow is this: If you don't know what something is or what it does, right-click it.

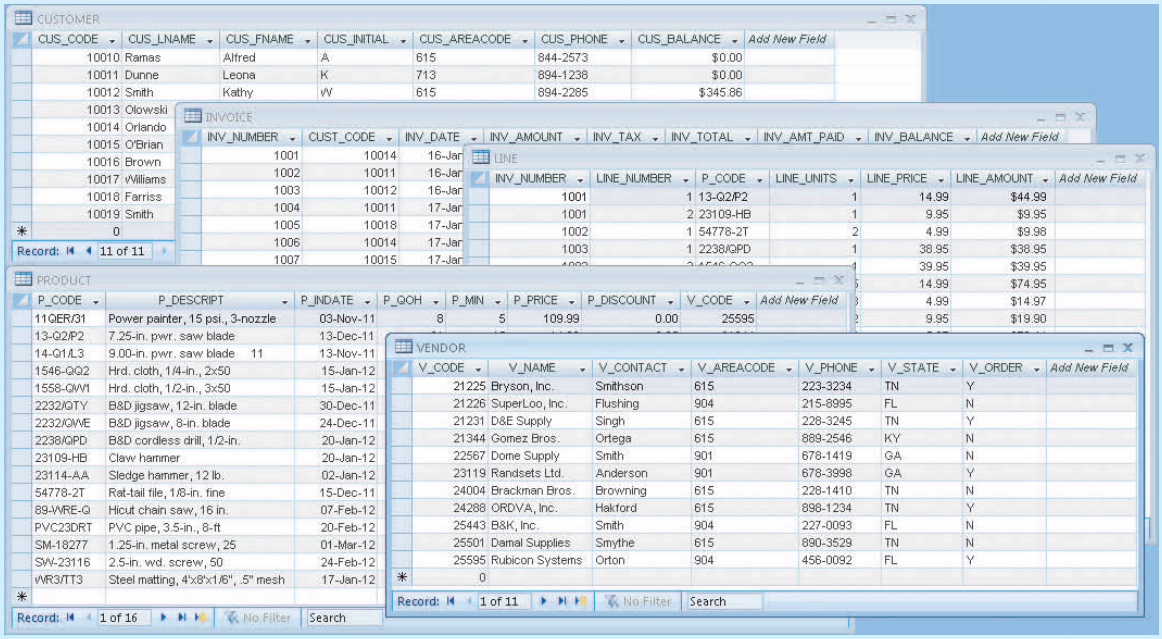
FIGURE M.35 Selecting the font format – Part 2



SOURCE: Course Technology/Cengage Learning

Now go ahead and edit the table structures of the remaining tables. Use the database tables shown in Figure M.36 as your guide. Remember, you imported most of the tables from the **Ch07_SaleCo** database, so the table values match those in that database ***at this point***. However, while the ***values*** match, a careful examination of Figure M.36 shows that we have already made a few changes in the **SaleCo** database.

FIGURE M.36 The SaleCo database tables



SOURCE: Course Technology/Cengage Learning

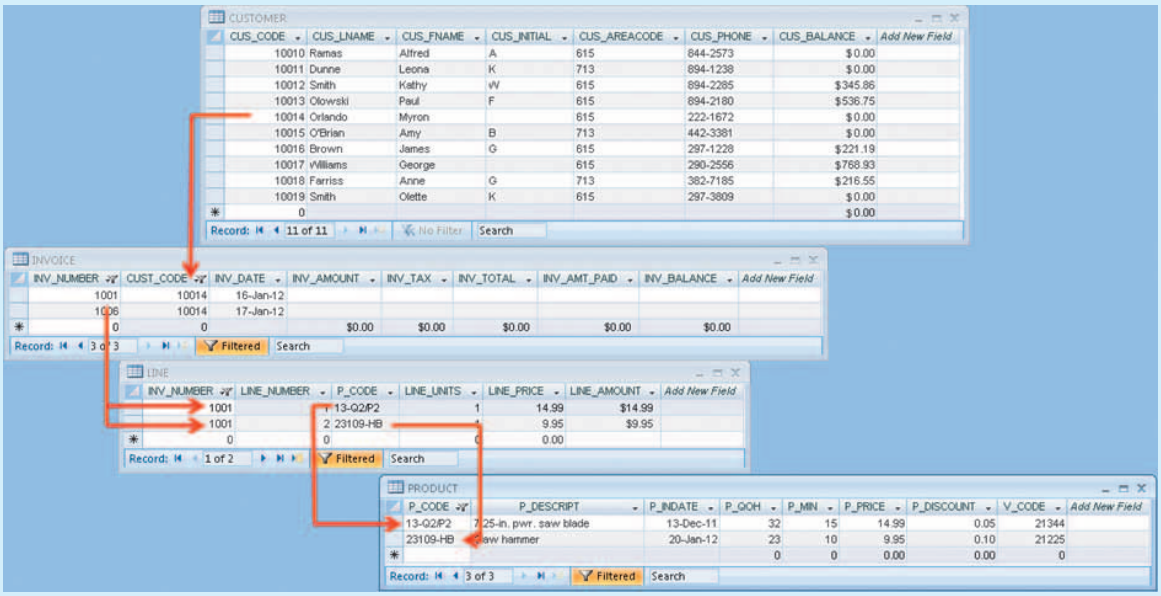
For example, if you look at the LINE table in Figure M.36, note that:

- The LINE table now has an additional currency field, LINE_AMOUNT, which is the product of the LINE_UNITS and LINE_PRICE. (The third record in the LINE table shows that the LINE_AMOUNT = 2 x $4.99 = $9.98.)
- The LINE_AMOUNT data type and field property were both set to **currency**, thus causing the dollar signs to precede the amount. You can change all the other fields that store dollar values to a currency data type at any time.
- There is an entry error in the first record of the LINE table—the LINE_PRICE is $14.99 and the LINE_UNITS value is 1, so the LINE_AMOUNT should be 1 x 14.99 = $14.99. (Go ahead and open the LINE table in its datasheet view and make the correction.)

1.4 Tracing a Transaction

How do you enter the values for the new attributes in the INVOICE table? You can, of course, hand-calculate the invoice amount, tax, total, and then enter the amount paid and calculate the balance. To get that job done, you'll have to do some major work. Let's do just one invoice entry at this point, using the "activity trace" shown in Figure M.37. (Note that the LINE_AMOUNT in the LINE table's first record has been corrected.)

FIGURE M.37 Tracing a purchase



SOURCE: Course Technology/Cengage Learning

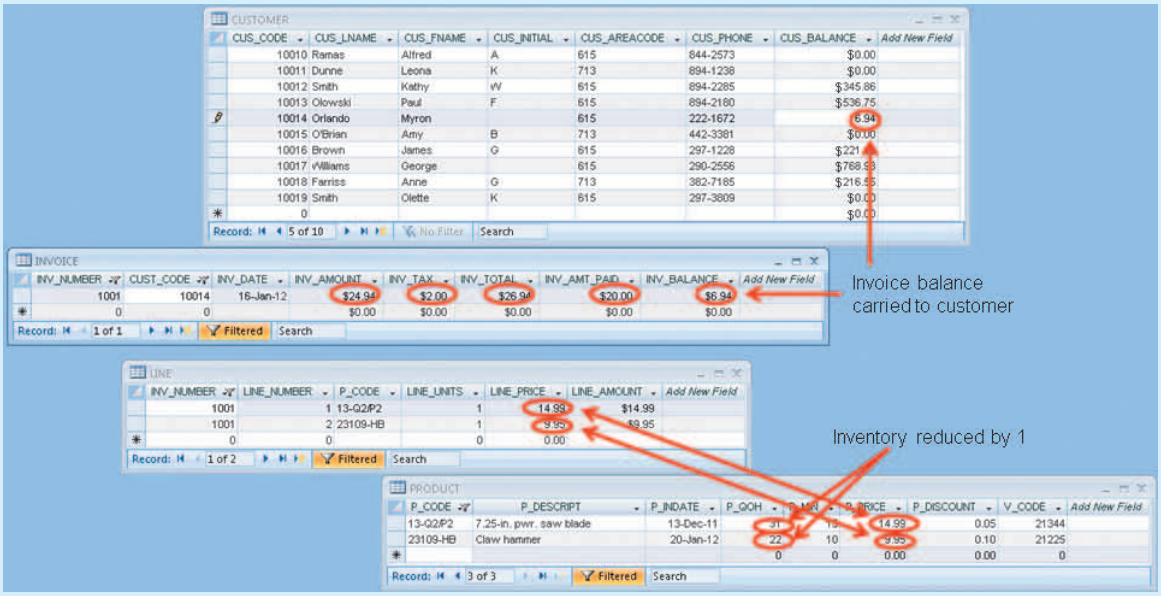
As you can tell by looking at Figure M.37, the following activities take place:

1. The customer 10014 (Myron Orlando) made the purchase(s) that produced invoice number 1001.
2. Invoice number 1001 records two line items, for a purchase of one $14.99 saw blade and one $9.95 claw hammer. Let's assume for simplicity's sake that discounts are not offered unless the purchase is made by a commercial customer … and none of the customers in the CUSTOMER table is a commercial purchaser. Therefore, type the PRODUCT table's PROD_PRICE values into the LINE table as LINE_PRICE values.
3. The total invoice amount is $14.99 + $ 9.95 = $ 24.94.

4. Assuming a tax percentage of 8%, the tax is $24.84 x 0.08 = $1.9952, rounded to $2.00.
5. The invoice total is thus $24.94 + $2.00 = $26.94.
6. Let's assume that the customer pays $20.00, leaving a $6.94 balance.
7. Customer 10014 now has a customer balance of $0.00 + $6.94 = $6.94.
8. The product table must be updated, too. The PROD_QOH values for each of the two products must be decreased by one each to reflect the sale activity.

The completed transaction is shown in Figure M.38. (The table limits were "dragged" to ensure that all the tables would fit in the same window in Figure M.38.)

FIGURE M.38 **The completed transaction**



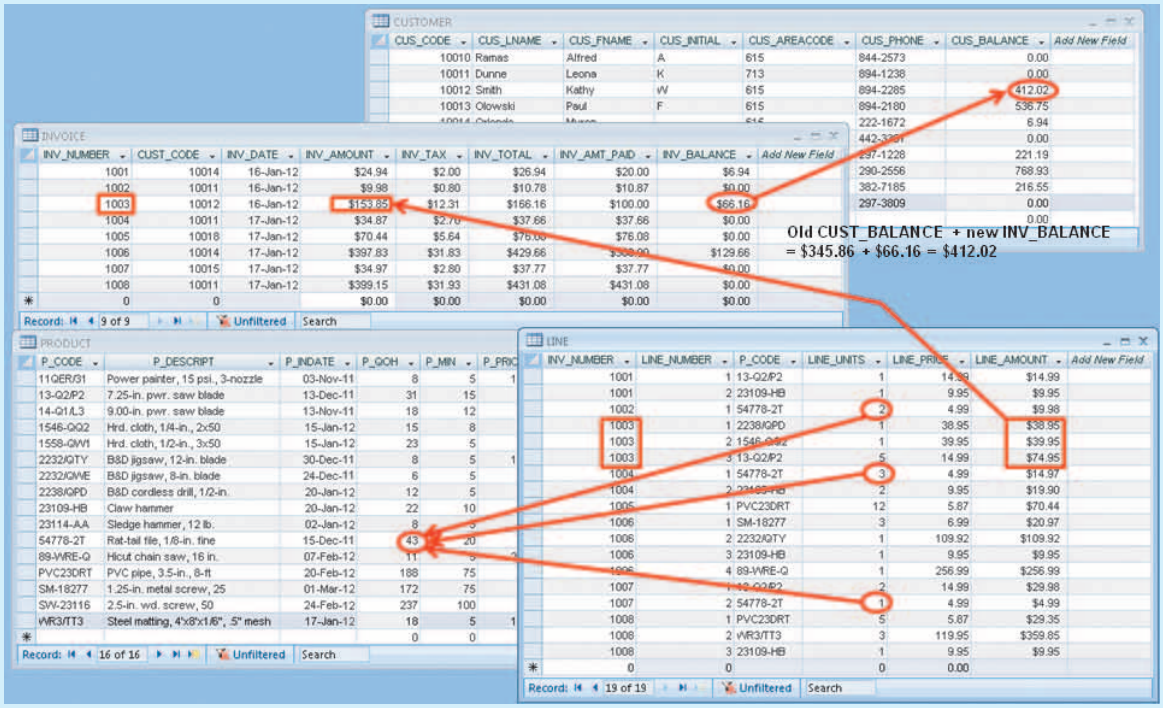
SOURCE: Course Technology/Cengage Learning

NOTE

Remember that the product price is copied to the LINE table to maintain the historical accuracy of the transaction. It is likely that, over time, the product prices will change … but the original product prices will be maintained in the LINE table to reflect the product price that was correct ***at the time of the transaction***. If necessary, review the text's Chapter 3, "The Relational Database Model," Section 3.7, "Data Redundancy Revisited."

Go ahead and complete the remaining transactions to practice the just illustrated tracing process. Note that the data entry is awkward, because you have to actually enter the product price in the LINE table, multiply the product price by the number of items (LINE_UNITS) to get the line total values. After that job is done, you'll have to calculate the subtotals, the sales tax, and the total in the INVOICE table, and then update the inventory—PROD_QOH in the PRODUCT table—and the customer balance—CUST_BALANCE in the CUSTOMER table. When you are done, your completed tables should contain the values shown in Figure M.39.

FIGURE M.39 All completed transactions



SOURCE: Course Technology/Cengage Learning

If you have recorded all the transactions shown in Figure M.39 manually, you realize that such time consuming and exacting tasks are not a good idea in the real world. You will learn in Section 2.1 that MS Access includes a query type known as an **Update Query** that, as its name suggests, can make the necessary transaction updates. In Section 5.1 you will learn that macros can be used to automate the transaction update process.

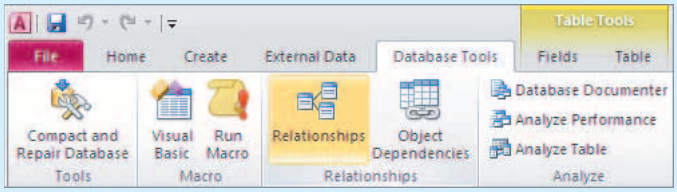
1.5 Setting the Relationships between Tables

Thus far, you have learned to create, populate, and import tables. Actually, if you were to create a real world relational database, you would first create the table structures and then create the relationships among them. The reason for doing so is simple—you want to make sure that referential integrity is enforced. (If you don't quite remember what referential integrity is and why it is important, review the text's Chapter 3, "The Relational Database Model," Section 3.2, "Keys."

To create a relationship, start by selecting the relationships option as shown in Figure M.40.

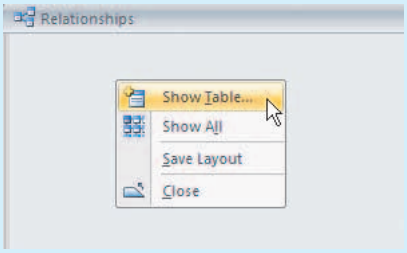
After you click the relationships button shown in Figure M.40, a blank relationships screen is shown. (That is, the relationships screen will be blank if you have not yet established relationships or you have not imported the relationships with the tables when you learned how to import tables.) Close the Show Table dialog box and right-click anywhere on the blank relationships screen to pop up the options—**Show Table...**, **Show All**, **Save Layout**—you see in Figure M.41.

FIGURE M.40 Select the Relationships option



SOURCE: Course Technology/Cengage Learning

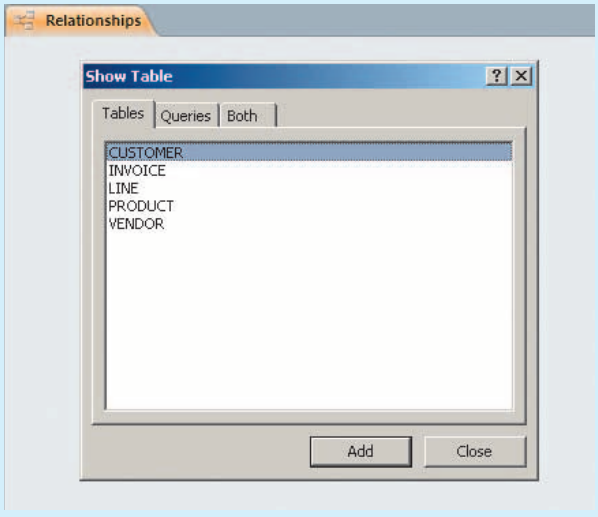
FIGURE M.41 Show Table



SOURCE: Course Technology/Cengage Learning

Click the **Show Table ...** option shown in Figure M.41 to generate the **Show Table** window you see in Figure M.42. Now select the two tables as shown in Figure M.42 and the click the **Add** button. (You can also double-click each table to move it from the **Show Table** window to the relationships screen.) This action will generate the screen shown in Figure M.43. Note: you will have to close the Show Table window to interact with the tables.

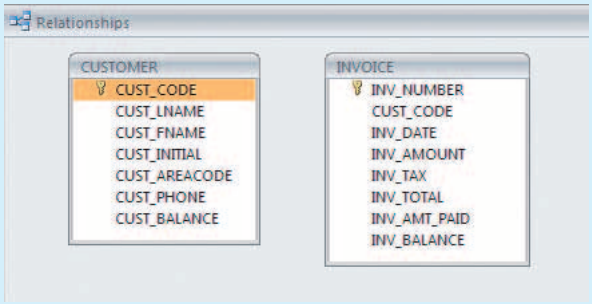
FIGURE M.42 Select tables



SOURCE: Course Technology/Cengage Learning

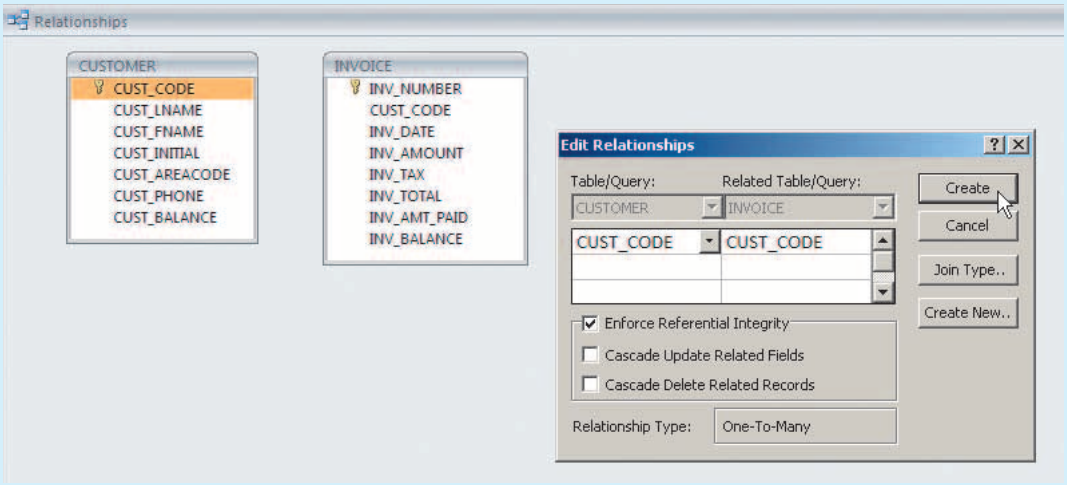
To create the relationship between the CUSTOMER and INVOICE tables, select the CUST_CODE in the CUSTOMER table and drag it to the CUST_CODE in the INVOICE table, and then drop it on the CUST_CODE in the INVOICE table. This action will produce the **Edit Relationships** dialog box you see in Figure M.44.

FIGURE M.43 Selected tables



SOURCE: Course Technology/Cengage Learning

FIGURE M.44 Create relationship

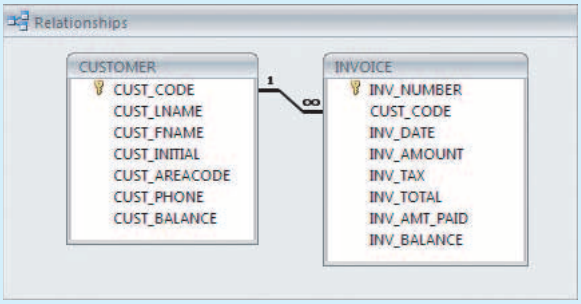


SOURCE: Course Technology/Cengage Learning

Next, click the **Enforce Referential Integrity** option you see in Figure M.44's **Edit Relationships** window to place a check mark in the square shown on the left side of that option and then click the **Create** button to create the relationship. This action will produce the results shown in Figure M.45. Note that the "1" side of the relationship is marked by the boldfaced **1**, while the many (M) side of the relationship is marked by the infinity symbol **∞**.

Now go ahead and create the relationships between all the tables. When you are done, your results will resemble Figure M.46. (We have dragged and sized the tables to enhance the presentation.)

FIGURE M.45 The relationship between CUSTOMER and INVOICE

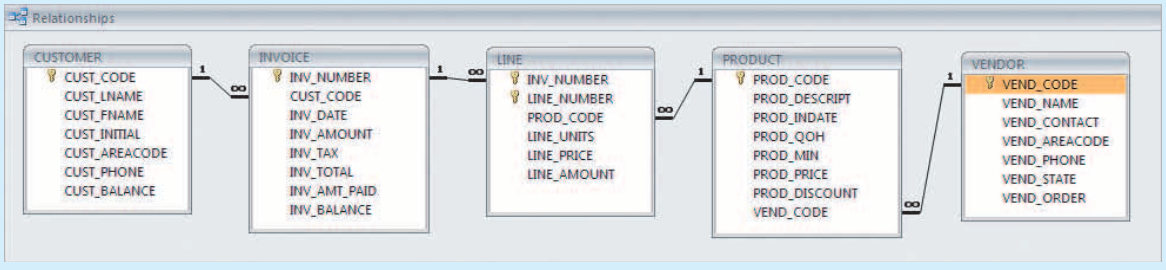


SOURCE: Course Technology/Cengage Learning

NOTE

Always drag and drop from the 1 side to the M side in a one-to-many (1:M) relationship. If you are creating a relationship between tables in a 1:1 relationship, drag from the parent entity to the dependent entity.

FIGURE M.46 The relationships for all the SaleCo database tables



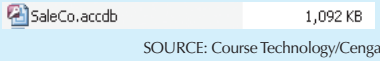
SOURCE: Course Technology/Cengage Learning

1.6 Repairing and Compacting the Database

A word of caution is in order. When you do a lot of work on your database, it tends to grow rapidly and its data dictionary fills up with all sorts of objects that you have edited or objects that do not exist. Sooner or later, this situation will degrade the performance of your database. Therefore, clean up the debris frequently.

In the following sequence, note that the **SaleCo** database size is 1,092 KB at this point. (See Figure M.47. Depending on the number of changes you made, your database size is likely to differ from the 1,092 KB you see here.)

FIGURE M.47 The SaleCo database size before the cleanup

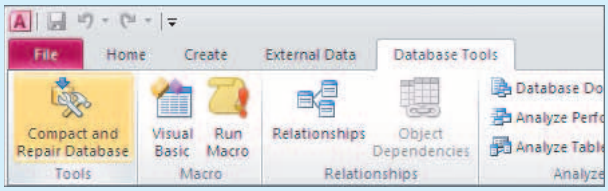


SOURCE: Course Technology/Cengage Learning

The cleanup is accomplished by selecting the sequence shown in Figure M.48—**Database Tools/Compact and Repair Database ...**

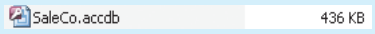
When the procedure is completed, look how much smaller the **SaleCo** database has become. Figure M.49 shows that the **SaleCo** database now uses only 436 KB.

FIGURE M.48 Compact and repair the database



SOURCE: Course Technology/Cengage Learning

FIGURE M.49 The SaleCo database size after the cleanup

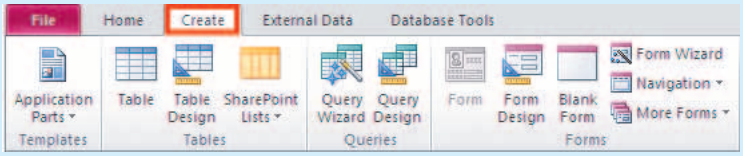


SOURCE: Course Technology/Cengage Learning

2.1 QUERIES

Queries are used to extract data from the database and to help transform data into information. To create a query, follow the procedure shown in Figure M.50. That is, click the **Create** tab in the Ribbon.

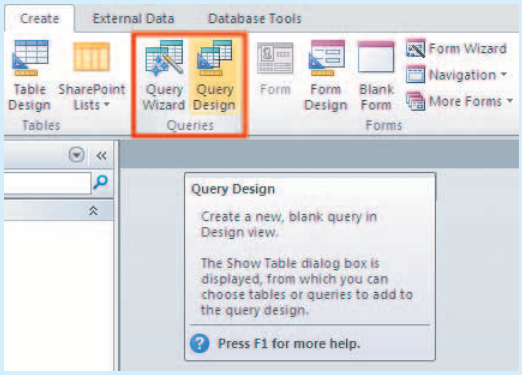
FIGURE M.50 Creating the first query



SOURCE: Course Technology/Cengage Learning

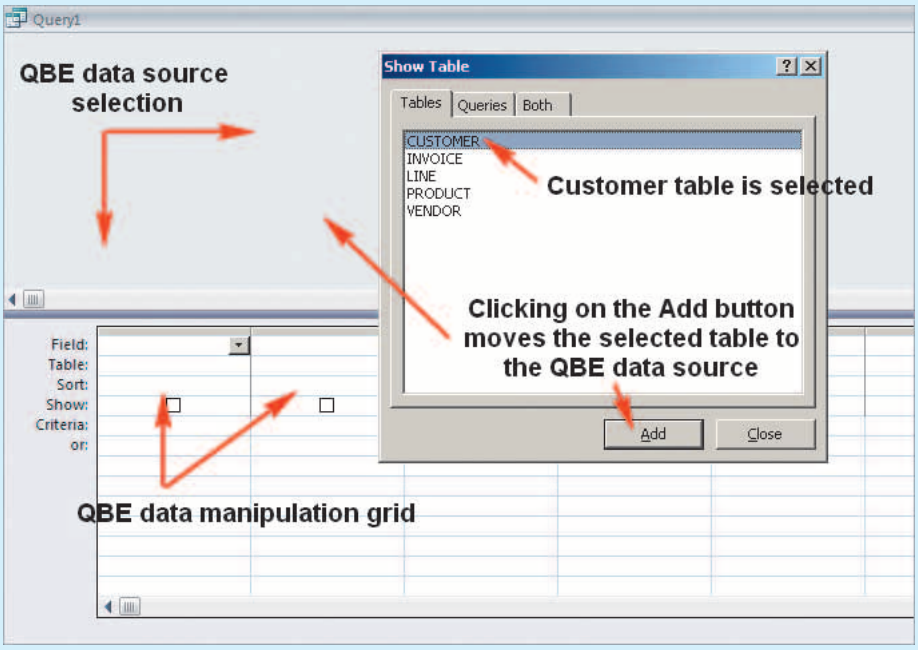
Clicking the **Create** tab gives you access to the Queries tool group that is shown in Figure M.50 and highlighted in Figure M.51. Select the **Query Design** option to generate the screen you see in Figure M.51. (Many of the wizards are useful, but you are better served by using the **Design view** option if you want to learn how to create and manipulate queries. All the queries in this tutorial will be created with the help of the MS Access Graphical User Interface, or GUI.)

FIGURE M.51 Selecting the Design view



SOURCE: Course Technology/Cengage Learning

FIGURE M.52 Selecting the CUSTOMER table



SOURCE: Course Technology/Cengage Learning

Figure M.52 shows three QBE (query by example) window components.

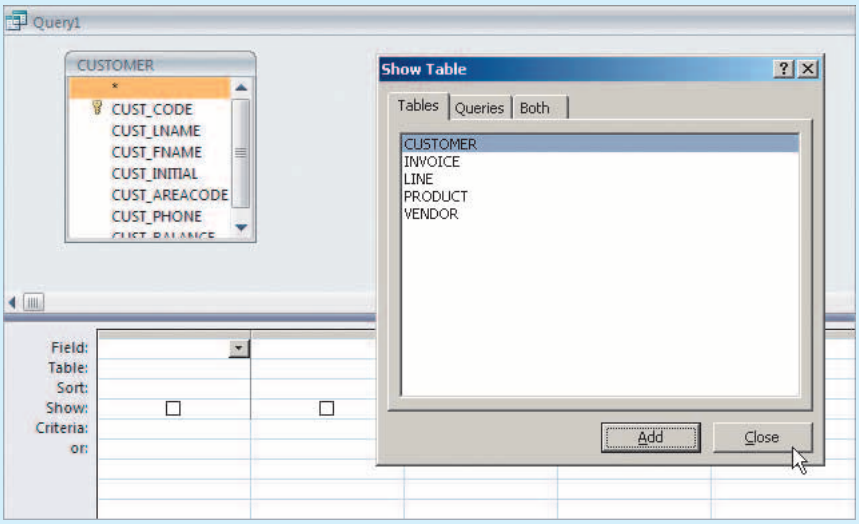
1. The top—blank—portion of the window is the QBE window's *data source* display, which may be a table or another query.
2. The bottom—lined—portion shows the available options in the QBE *grid*. The grid represents the *data manipulation* portion of the QBE window. As its name suggests, the **Field:** option lets you place a table's—or

query's—field on the line. The **Table:** shows the field's origin. (Clicking on the **Queries** tab will show the list of queries that are available as data sources. You can query a query.) The **Sort:** option lets you sort selected field values in either ascending or descending order. The **Show:** option lets you select whether or not to display a selected field value set. (Note that the box on the **Show:** line is not marked at this point, indicating that a selected field's values will not be shown.) The **Criteria:** option lets you select various display options and constraints. And the **Or:** option lets you modify the **Criteria:** constraint selection. (For example, you might place a restriction of the query output for a CUSTOMER table to show only customers named "Smith" … ***or*** to also show customers whose current balance is over $200.)

3. The **Show Table** dialog box lets you select tables or queries to be placed as data sources in the first portion of the screen. (If the **Show Table** dialog box is not shown, right-click the blank portion of the screen.)

To place the CUSTOMER table on the data source portion of the screen, either select the CUSTOMER as shown in Figure M.52 and then click the **Add** button or double-click the CUSTOMER table. When you have placed all the tables you need on the top (data source) portion of the screen, close the **Show Table** dialog box by clicking on its **Close** button. (See Figure M.53.) In this example, only the CUSTOMER table has been selected as the data source.

FIGURE M.53 **Closing the Show Table window**

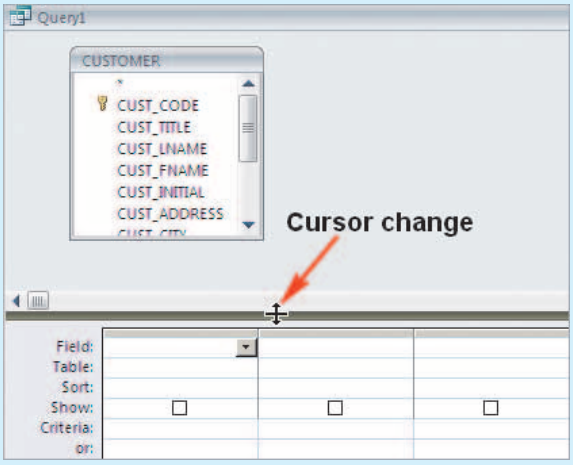


SOURCE: Course Technology/Cengage Learning

After you have selected the data source—the CUSTOMER table—and placed it on the data source portion of the QBE screen, size the QBE window's components by dragging their limits. Figure M.54 shows that the cursor changes when you place it on the boundary between the data source component that now contains the CUSTOMER table and the data manipulation grid component. When the cursor changes its shape to the two-sided arrow seen here, you can drag the boundary up or down.

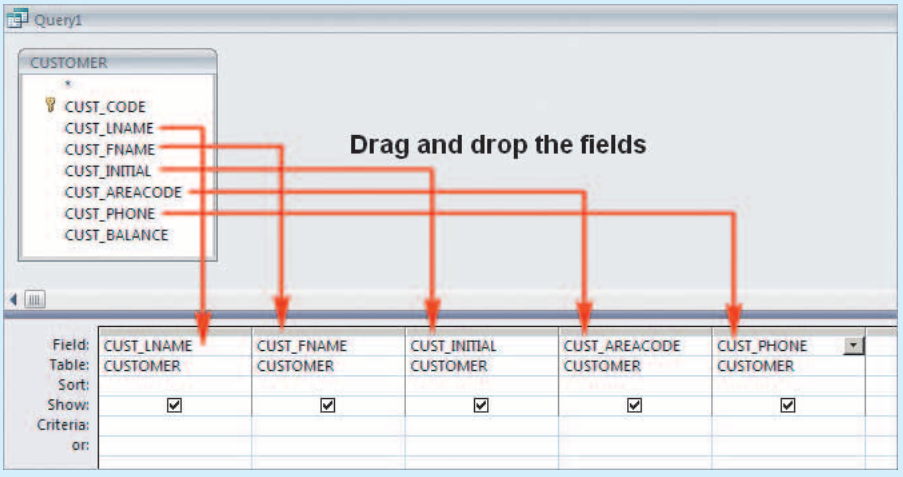
After enlarging the data component portion of the QBE window, drag the CUSTOMER limits to show all the fields and their names. Then select the CUSTOMER table's fields and drag and drop them in the **Field:** spaces as shown in Figure M.55. You can drag and drop the individual fields from the CUSTOMER table to the field spaces in the data manipulation section or you can select multiple fields and drop them on a single field space. (Access will automatically space them across the field spaces.) Note that the field origin is automatically displayed in the **Table:** portion of the grid.

FIGURE M.54 Size the table window



SOURCE: Course Technology/Cengage Learning

FIGURE M.55 Drag and drop the fields

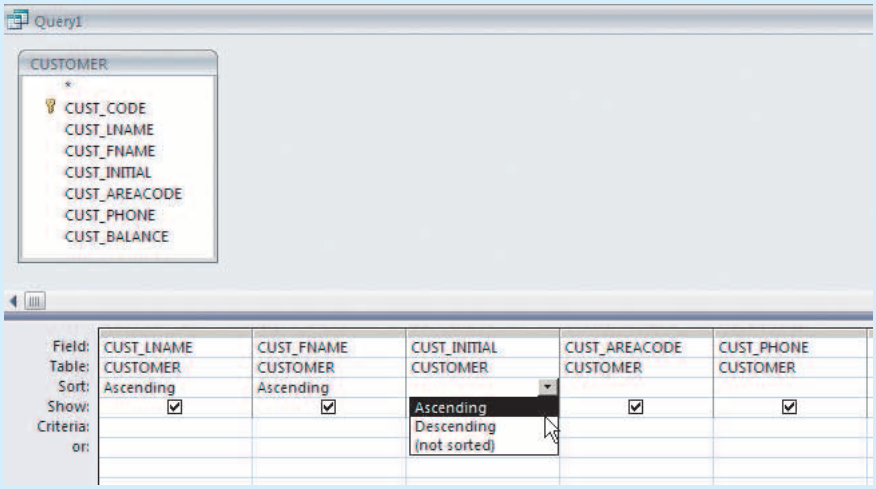


SOURCE: Course Technology/Cengage Learning

After you have selected the fields you want to use in the query, you can choose whether or not to sort the field values by clicking on the grid's **Sort:** option. Clicking on this option yields a drop-down list of option you see in Figure M.56. Click the option you want to use to move the option to the grid. (As you can tell, the CUST_LNAME, CUST_FNAME, and CUST_INITIAL show the selection of the ascending option.)

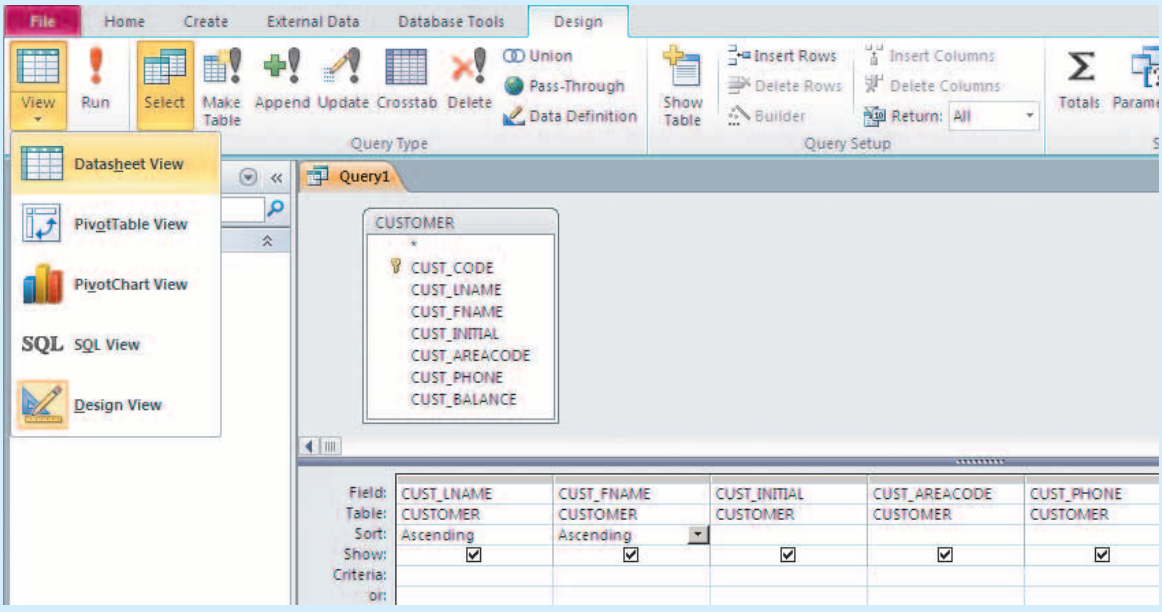
If you want to see the effect of the query actions you have taken thus far, change the **View** from the current **Design View** to the **Datasheet View**, as shown in Figure M.57.

FIGURE M.56 Select Ascending sort



SOURCE: Course Technology/Cengage Learning

FIGURE M.57 Select Datasheet View



SOURCE: Course Technology/Cengage Learning

The selected **Datasheet View** yields the query output shown in Figure M.58.

FIGURE M.58 The Datasheet view

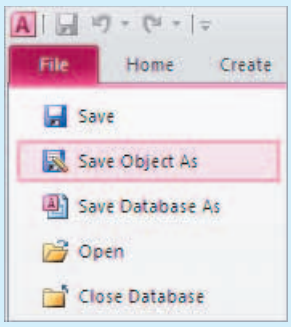
CUST_LNAME	CUST_FNAME	CUST_INITIAL	CUST_AREACODE	CUST_PHONE
Brown	James	G	615	297-1228
Dunne	Leona	K	713	894-1238
Farriss	Anne	G	713	382-7185
O'Brian	Amy	B	713	442-3381
Olowski	Paul	F	615	894-2180
Orlando	Myron		615	222-1672
Rames	Alfred	A	615	844-2573
Smith	Kathy	W	615	894-2285
Smith	Olette	K	615	297-3809
Williams	George		615	290-2556

SOURCE: Course Technology/Cengage Learning

Incidentally, you can easily control the query display characteristics through the **Font** option you see in the button bar. For example, you can change the font size from the (default) 10 value to 8, you can select the font type, and so on.

Before you do any additional work on the query, save it. Figure M.59 shows the selection of the **File/Save Object As** option.

FIGURE M.59 Select the Save As file option

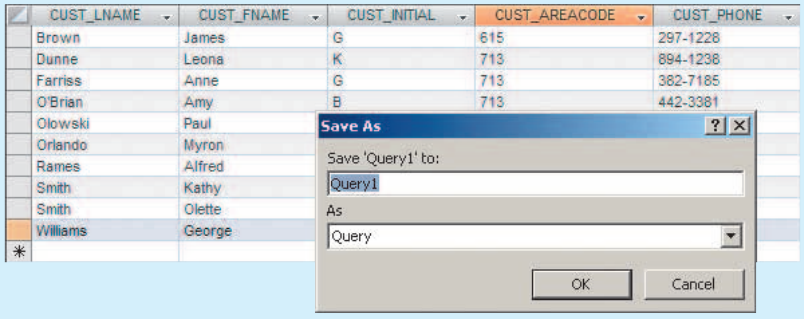


SOURCE: Course Technology/Cengage Learning

The selection of the **File/Save Object As** option shown in Figure M.59 will produce the **Save As** dialog box you see in Figure M.60. The dialog box shows a default query name, in this case, **Query1**.

Always use a query name that is self-documenting. In this example, the query will be used as the basis for a phone list. Therefore, **PhoneList** is an appropriate name. However, you want to also show that this object is a query, so use the

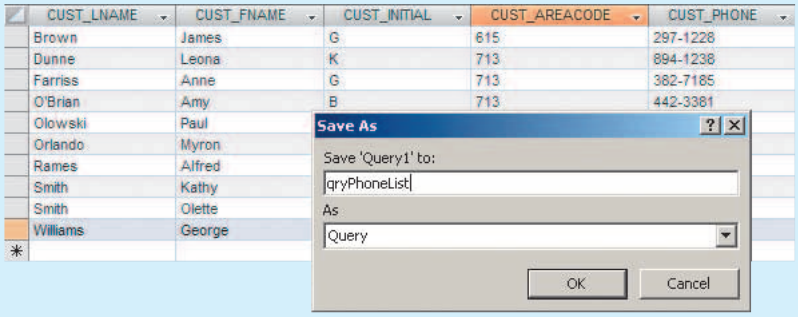
FIGURE M.60 The Save As dialog box



SOURCE: Course Technology/Cengage Learning

prefix **qry** to indicate that fact. Therefore, the appropriate name will be **qryPhoneList**. And you see that query name used in Figure M.61. (You will discover that this self-documenting naming convention is very desirable, because it enables you to easily keep track of multiple components in an application set. For example, you will learn how to create forms in Section 3 … and if you see two objects, **frmPhoneList** and **qryPhoneList**, you will know which object you are looking at, and it will be easy to see that the data source for the form is the query that has the same name. Using naming conventions that are ***not*** self-documenting is ***not*** a sign of professionalism—sowing confusion is not an ideal worth pursuing.)

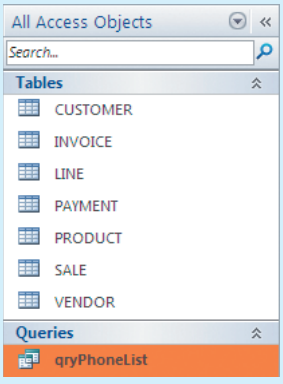
FIGURE M.61 The query name



SOURCE: Course Technology/Cengage Learning

After you type the query name as shown in Figure M.61, click **OK** to save the query. Note that the saved query now shows up in the **Queries** list shown in Figure M.62.

FIGURE M.62 The first listed query



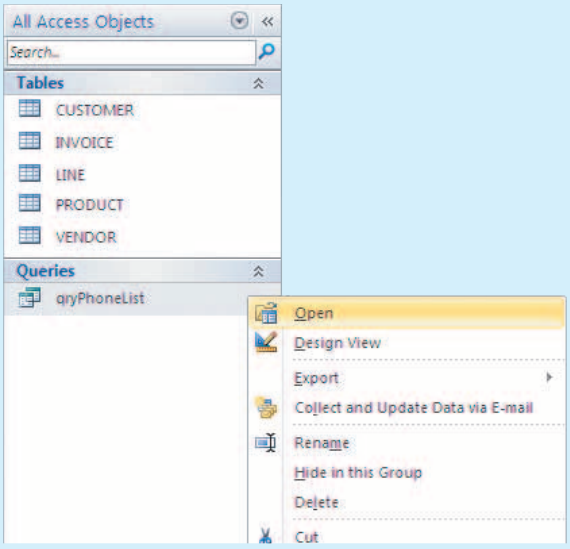
SOURCE: Course Technology/Cengage Learning

After saving the query, you can take a quick look at its output by selecting the query **Open** option shown in Figure M.63. (You can also open the query by double-clicking on the query name. Go ahead and do so, then close the query.)

2.1.1 Editing the Query Output

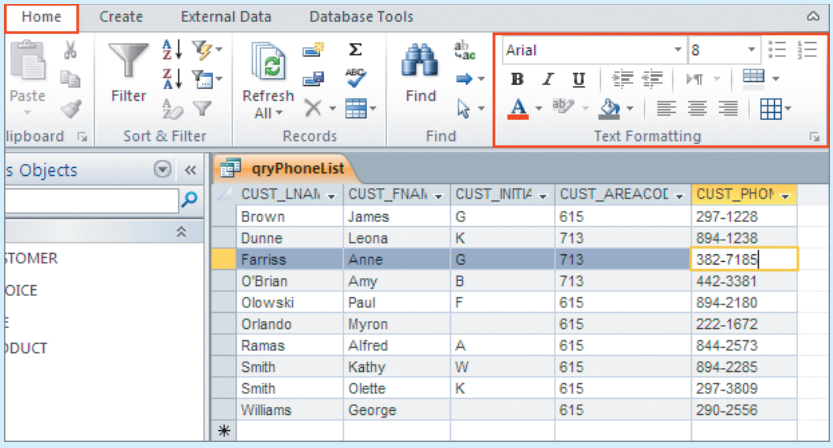
If you are not satisfied with the default font selection when you open the query in its datasheet view, you can easily modify those fonts. Just click the **Home** tab at the top of the screen to view the **Font** selection box shown in Figure M.64. Note that you can select a **Font:**, a **Font Style:**, and a **Size:**. Figure M.64 shows the selection of an **Arial** font, a **Regular** font style, and a font size of **8**.

FIGURE M.63 Open the query



SOURCE: Course Technology/Cengage Learning

FIGURE M.64 Font selection

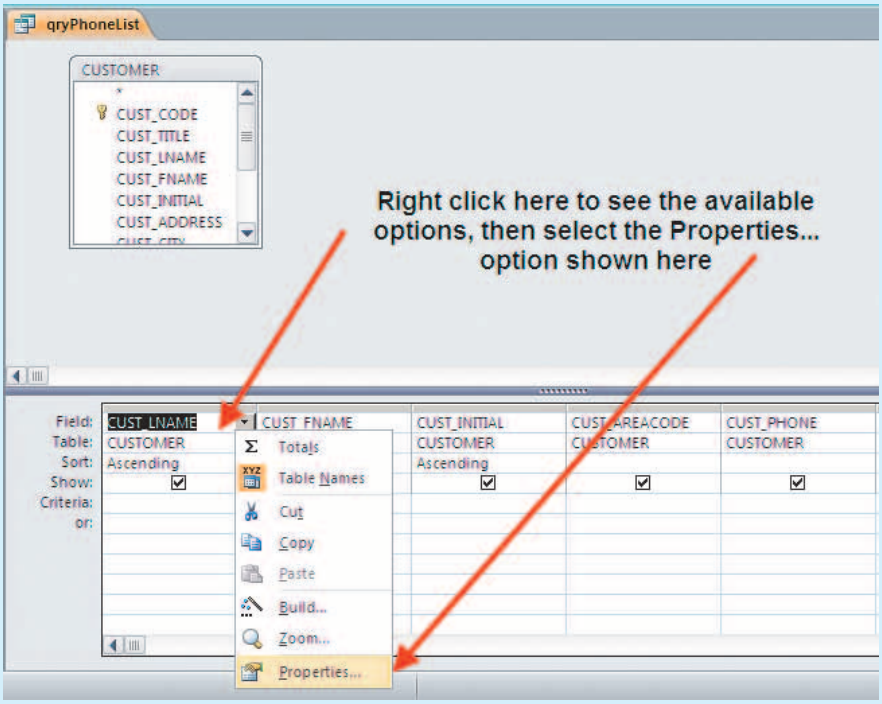


SOURCE: Course Technology/Cengage Learning

Because the default query output uses the field headers—such as CUST_LNAME—used by the query data source table, you might want to make the presentation more "finished" looking by changing the query field headers. You can get that job done while you are in the query design mode. Note the procedure illustrated in Figure M.65. That is, right-click

a selected field name grid cell to see the list of available options, and then click the **Properties...** option to generate the **Property Sheet** you see in Figure M.66.

FIGURE M.65 Selecting the field Properties... option

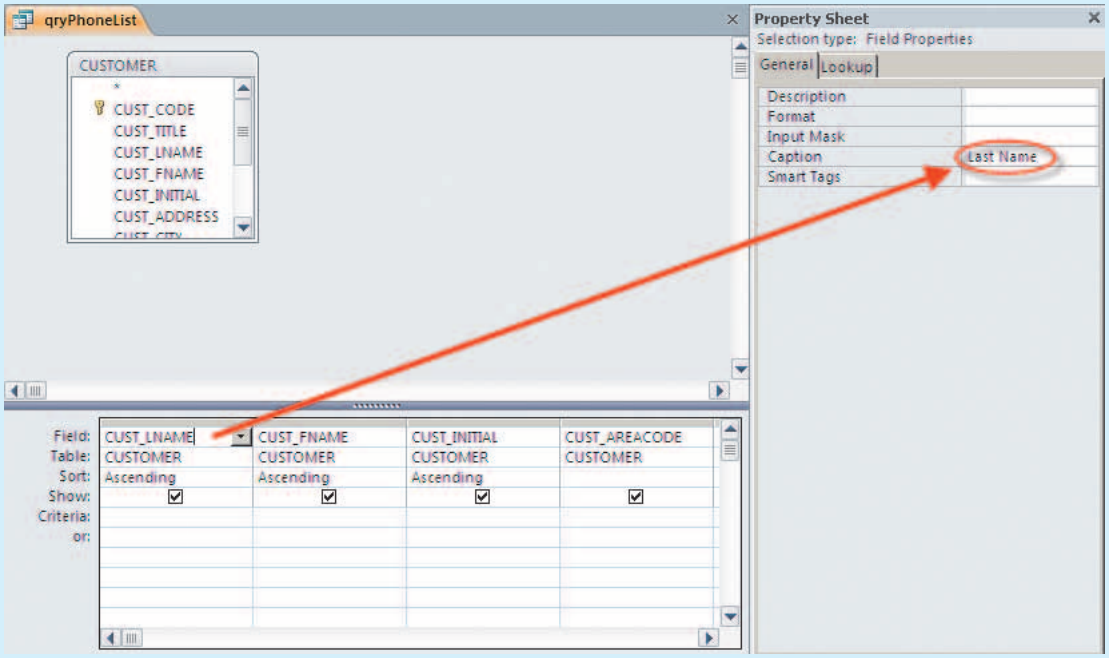


SOURCE: Course Technology/Cengage Learning

You can change the field header by selecting the **Caption** option in the **Property Sheet** you see in Figure M.66, then type in the field header you want to use. In this case, the field header will be **Last Name**. Changing the query header does ***not*** affect the attribute name (CUST_LNAME) in the query data source—the CUSTOMER table.

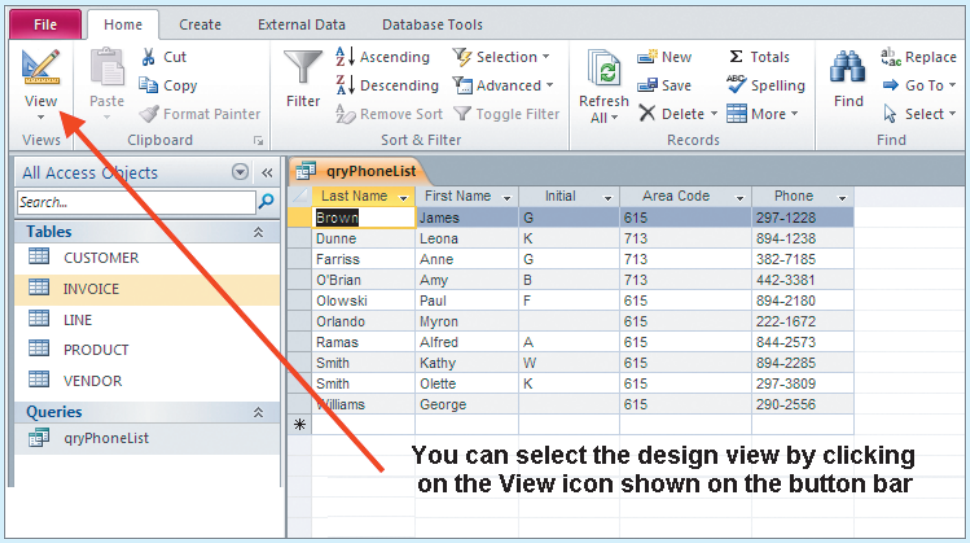
Repeat the editing for the remaining query fields and then open the query in its **Datasheet View** to see the editing results shown in Figure M.67.

FIGURE M.66 Modifying the field caption



SOURCE: Course Technology/Cengage Learning

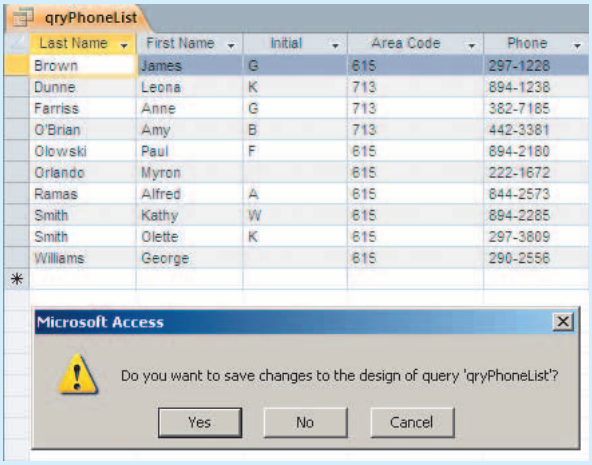
FIGURE M.67 The modified query field headers



SOURCE: Course Technology/Cengage Learning

If you are satisfied with the results, remember to save the query again. (If you forget to save, Access will remind you as shown in Figure M.68.)

FIGURE M.68 Reminder to save

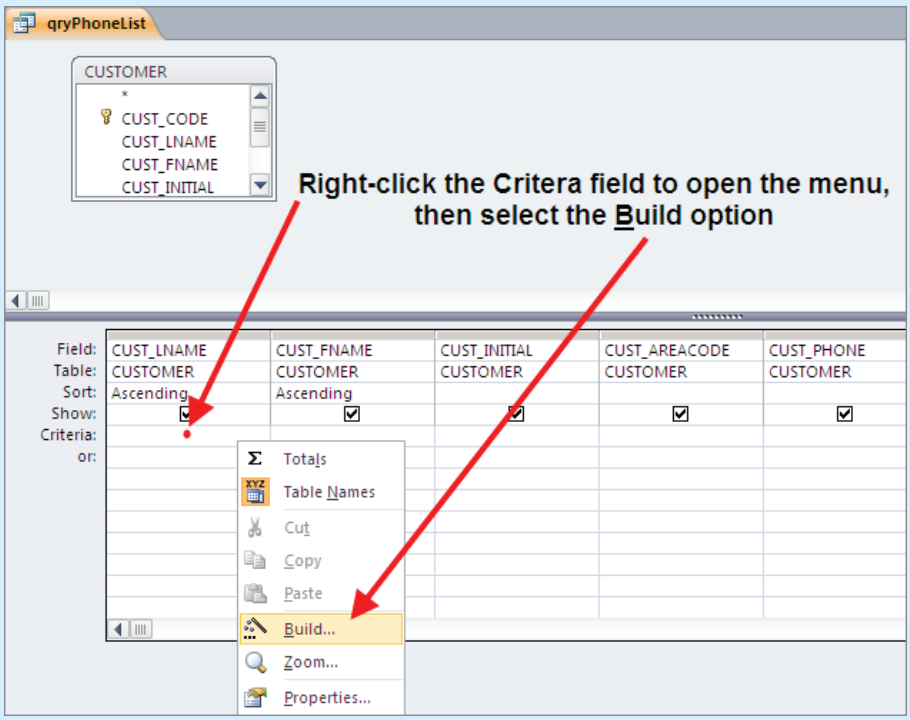


SOURCE: Course Technology/Cengage Learning

2.1.2 PARAMETER QUERIES

You can easily create a query in which you specify the ***criteria*** governing the query output. Note the procedure summarized in Figure M.69. In the following example, the objective will be to limit the phone list output to a specified customer's last name. (A query whose output is limited through specified criteria that restrict output for one or more parameters is also known as a ***parameter query***.)

FIGURE M.69 Setting criteria



SOURCE: Course Technology/Cengage Learning

If you want to limit the query phone list output to customers whose last name is "Smith", you can simply type "Smith" in the CUST_LNAME criteria grid space—marked by the red dot in Figure M.69. Unfortunately, that procedure means that the query must be changed each time a different last name limitation is required. You will have a much more flexible query if you let the end user specify the last name restriction through a dialog box. Such a dialog box is created automatically if you type

Like "*" & [Enter the last name] & "*"

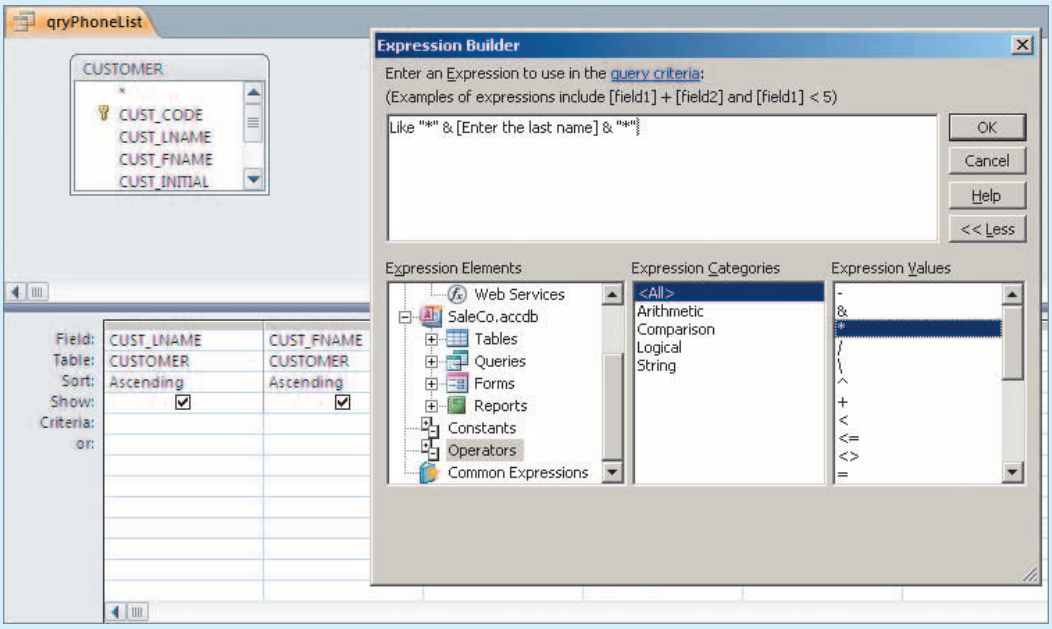
in the CUST_LNAME criteria grid space. (Review Chapter 7, "Introduction to Structured Query Language (SQL)", Section 7.4.4, to review the LIKE operator and wildcard characters such as "*".)

NOTE

Although you can type the simple criteria restriction directly into the grid, you should become used to the **Expression Builder**. This tool is especially useful when you later try to enter more complex criteria or even simple criteria with multiple components. In fact, you will discover the Section 5, "Macros," that you will have a difficult time typing the sometimes long character strings without making errors … it will be a lot easier to select items from a list than to do the typing. Therefore, we will use the expression builder in most examples.

If you click the **Build...** option shown in Figure M.69, you will see the **Expression Builder** dialog box shown in Figure M.70. You can either type the entire expression or you can type **Like "**, click the *, type **"**, click the **&** … and so on. (You do not, of course, type the "**|**" symbol you see at the end of the expression line … that' just the shape of the cursor in the **Expression Builder**.) Although you can type the * and **&** symbols easily enough, go ahead and practice using the expression builder's features while the expressions are very simple. Familiarity breeds, well, competence!

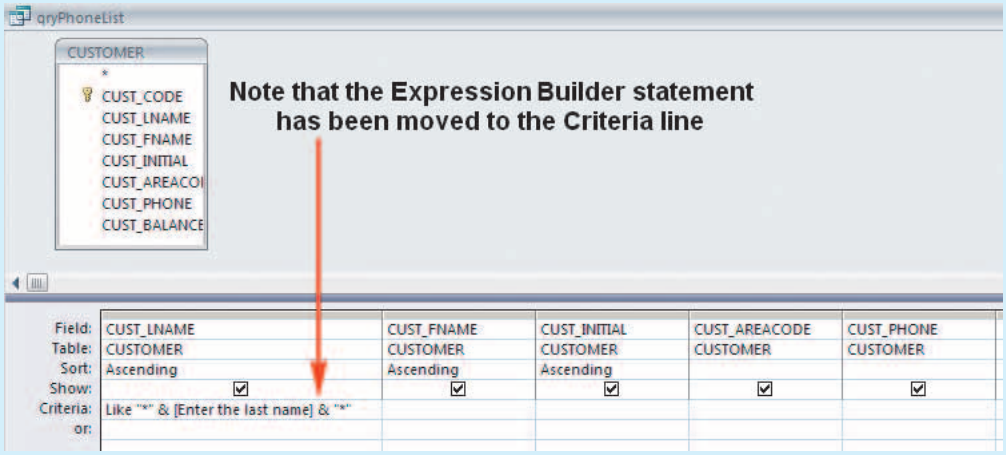
FIGURE M.70 The Expression Builder



SOURCE: Course Technology/Cengage Learning

When you have completed the expression shown in Figure M.70, click **OK** to close the **Expression Builder** and to transfer the expression to the QBE grid as shown in Figure M.71. (If you want to see the entire expression in the grid space, drag the grid space limit to widen it, as was done in Figure M.71.)

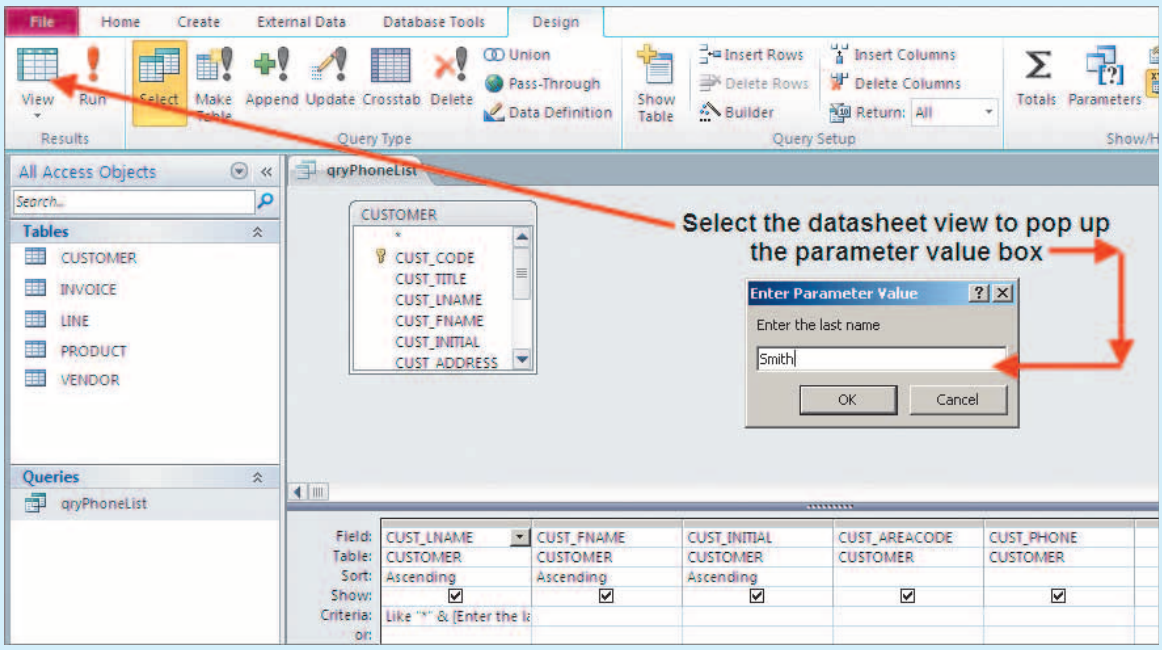
FIGURE M.71 The completed Criteria line



SOURCE: Course Technology/Cengage Learning

Next, open the query in its datasheet view. The expression you see in Figure M.71 will trigger the input request you see in Figure M.72. Type the last name **Smith** to generate the output shown in Figure M.73. Incidentally, the parameter search is not case-sensitive. Therefore, it goes not matter whether you type **SMITH**, **smith**, **Smith**, or any other combination of lower- and upper-case letters. Also, keep in mind that the use of the * wildcard character in combination with **Like** will yield these results:

FIGURE M.72 Name selection



SOURCE: Course Technology/Cengage Learning

INPUT	OUTPUT
None. (You just press the **Enter** key, instead of typing a character and then pressing the **Enter** key.)	All records.
The letter **s**.	All records corresponding to a customer whose last name includes the letter **s**. For example, Ramas, Williams, Olowski, Farriss, and Smith would all be included.
The letters **br**.	All records corresponding to a customer whose last name includes the letters **br**. For example, customer O'Brian and Brown would be included.

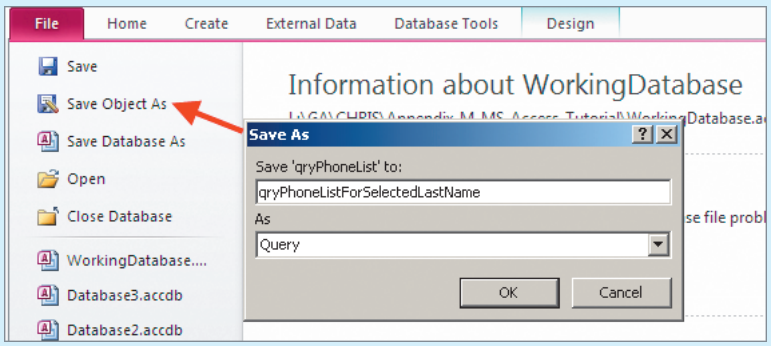
FIGURE M.73 Phone numbers for selected name

Last Name	First Nan	Initial	Area Code	Phone
Smith	Kathy	W	615	894-2285
Smith	Olette	K	615	297-3809
*				

SOURCE: Course Technology/Cengage Learning

Using the **Save As** option to save a file, save the **qryPhoneList** query you have just modified as **qryPhoneListForSelectedLastName**. (See Figure M.74.If you have maximized the screen, you will only see the **Save As** dialog box.)

FIGURE M.74 Saving the parameter query



SOURCE: Course Technology/Cengage Learning

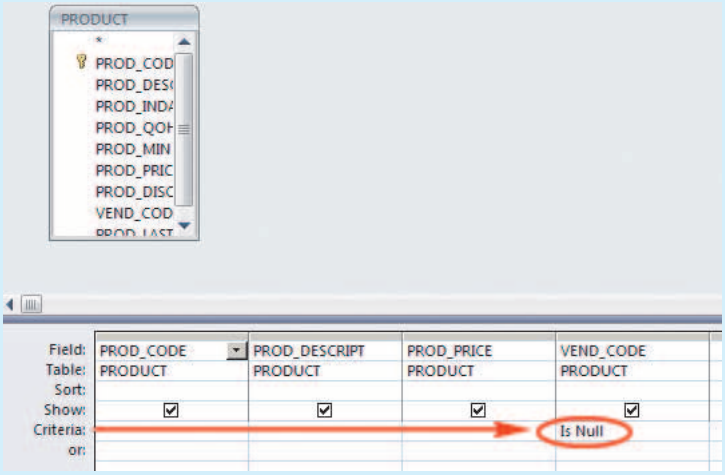
You can use the same technique to limit output by any selected criteria for any field. For example, Figure M.75 shows a query that limits its output by VEND_CODE values that are null. (In short, the output will show all products that do not have a known vendor.) The output is shown in Figure M.76.

Note that the query was saved as **qryProductsWithoutVendors**.

2.1.3 Multiple Table Queries

You can include any number of tables in your queries. For example, note that the query in Figure M.77 includes two tables, CUSTOMER and INVOICE. You do not have to relate the tables, because that was done earlier via the relationships window. (See Section 1.3.)

FIGURE M.75 Products without vendors



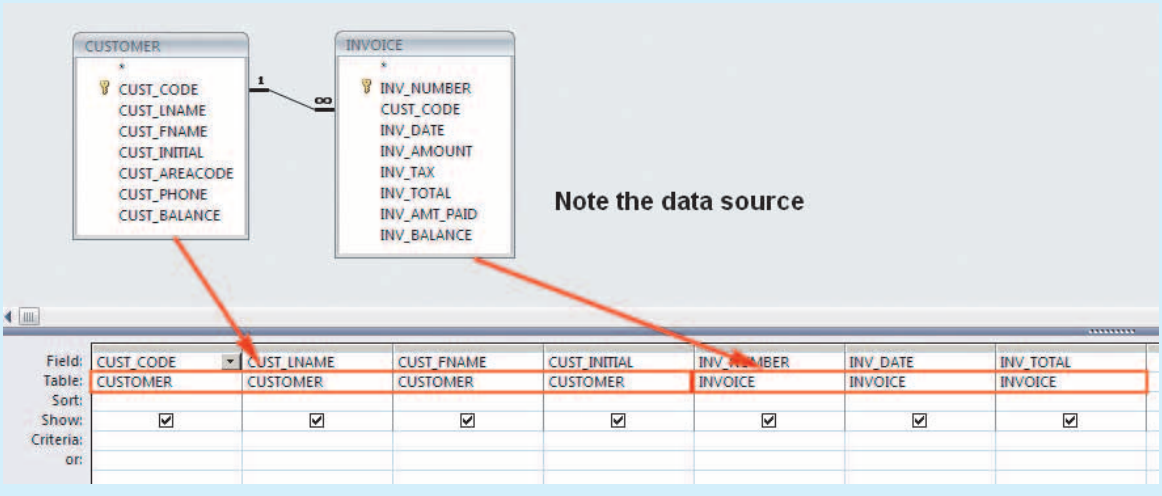
SOURCE: Course Technology/Cengage Learning

FIGURE M.76 Products without vendors output

PROD_CODE	PROD_DESCRIPT	PROD_PRICE	VEND_CODE
23114-AA	Sledge hammer, 12 lb.	$14.40	
PVC23DRT	PVC pipe, 3.5-in., 8-ft	$5.87	
		$0.00	0

SOURCE: Course Technology/Cengage Learning

FIGURE M.77 Multiple table query



SOURCE: Course Technology/Cengage Learning

Save this query as **qryCustomerInvoices** and then open this query to see its output. (See Figure M.77a.)

FIGURE M.77a The qryCustomerInvoices output

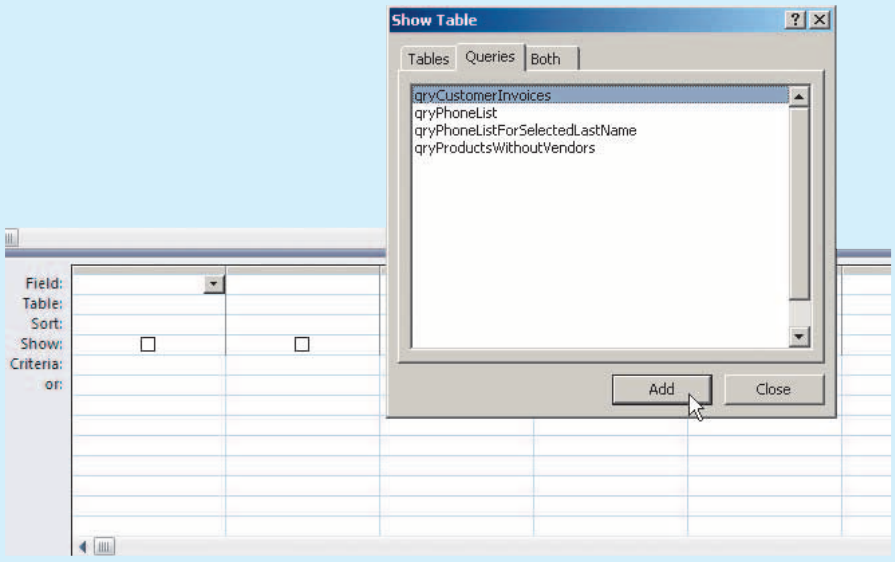
CUST_CODE	CUST_LNAME	CUST_FNAME	CUST_INITIAL	INV_NUMBER	INV_DATE	INV_TOTAL
10011	Dunne	Leona	K	1002	16-Mar-12	$10.78
10011	Dunne	Leona	K	1004	17-Mar-12	$37.66
10011	Dunne	Leona	K	1008	17-Mar-12	$431.08
10012	Smith	Kathy	W	1003	16-Mar-12	$166.16
10014	Orlando	Myron		1001	16-Mar-12	$26.94
10014	Orlando	Myron		1006	17-Mar-12	$429.66
10015	O'Brian	Amy	B	1007	17-Mar-12	$37.77
10018	Farriss	Anne	G	1005	17-Mar-12	$76.08
10019	Smith	Olette	K	1009	27-Mar-12	$195.20

SOURCE: Course Technology/Cengage Learning

2.1.4 QUERYING A QUERY

Suppose you want to know the total sales for each of the customers. If you run the **qryCustomerInvoices** query, you will see all the invoices for each of the customers. For example, if you look at Figure M.77a, you see that customer 10011, Leona Dunne, has three invoices for $10.78, $37.66, and $431.08, respectively. The sum of these invoice totals will be $479.52. How do write a query that will generate the sum of all the invoice totals for each of the customers? The answer turns out to be simple: As you can see in Figure M.78, you can write a query that uses the **qryCustomerInvoices** query output as its data source. (Note that the **Show Table** dialog box shows that the **Queries** tab was selected.)

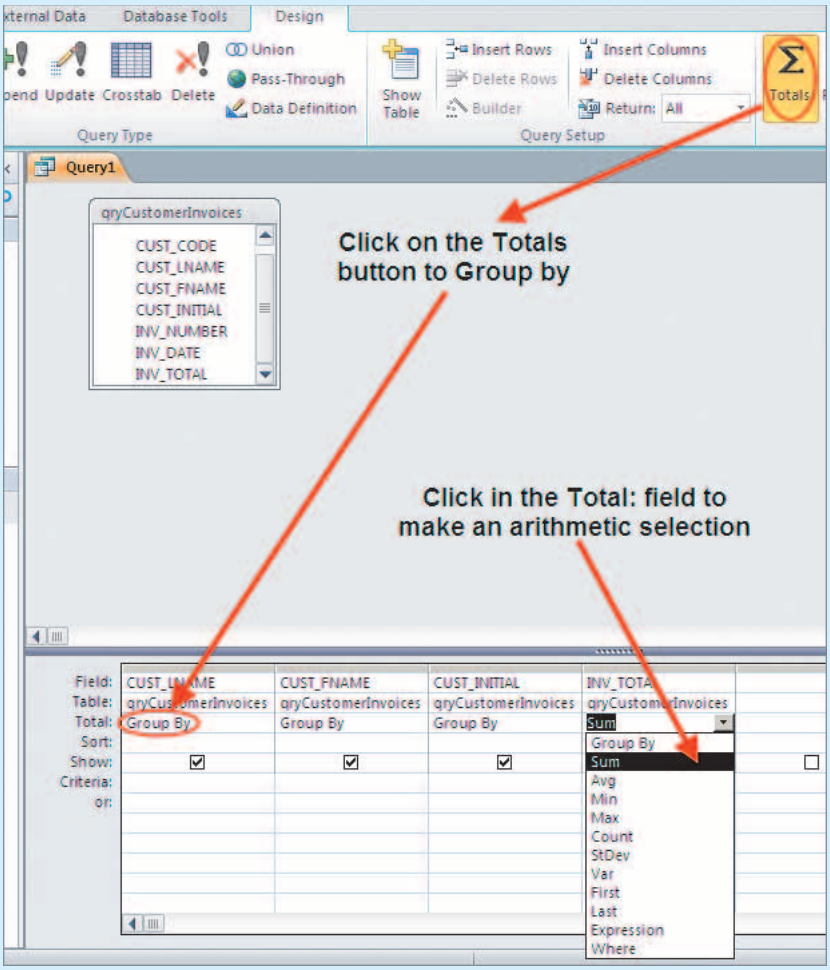
FIGURE M.78 Querying a query



SOURCE: Course Technology/Cengage Learning

Figure M.79 shows the selected fields. It also shows that, after the fields have been dragged and dropped to their **Field:** spaces. Finally, it shows the selection of the **Totals** button at the top of the screen. (The button, circled in red, shows the summation symbol.)

FIGURE M.79 Invoice totals by customer



SOURCE: Course Technology/Cengage Learning

Figure M.79 also shows the remaining actions required to complete the query design:

- Click the **Totals** button to insert the **Group By** entry into all the **Total:** spaces.
- Click the INV_TOTAL field space to produce the drop-down list.
- Select the **Sum** option from the list.

Now check the completed query in its **Datasheet View** to generate the results shown in Figure M.80. Note that the $479.52 sum for customer Leona Dunne is correct. (Customer Leona Dunne's customer number is 10011. If you take a look at the INVOICE table contents displayed in Figure M.80A, you will see that customer 10011 has invoice totals of $10.78, $37.66, and $411.08. The sum of these three invoice totals is $479.52.)

FIGURE M.80 Customer invoice totals

CUST_LNAME	CUST_FNAME	CUST_INITIAL	SumOfINV_TOTAL
Dunne	Leona	K	$479.52
Farriss	Anne	G	$76.08
O'Brian	Amy	B	$37.77
Orlando	Myron		$456.60
Smith	Kathy	W	$166.16
Smith	Olette	K	$195.20

SOURCE: Course Technology/Cengage Learning

Save the query as **qryCustomerInvoiceTotals** to complete the query design process.

FIGURE M.80a INVOICE table entries for customer 10011

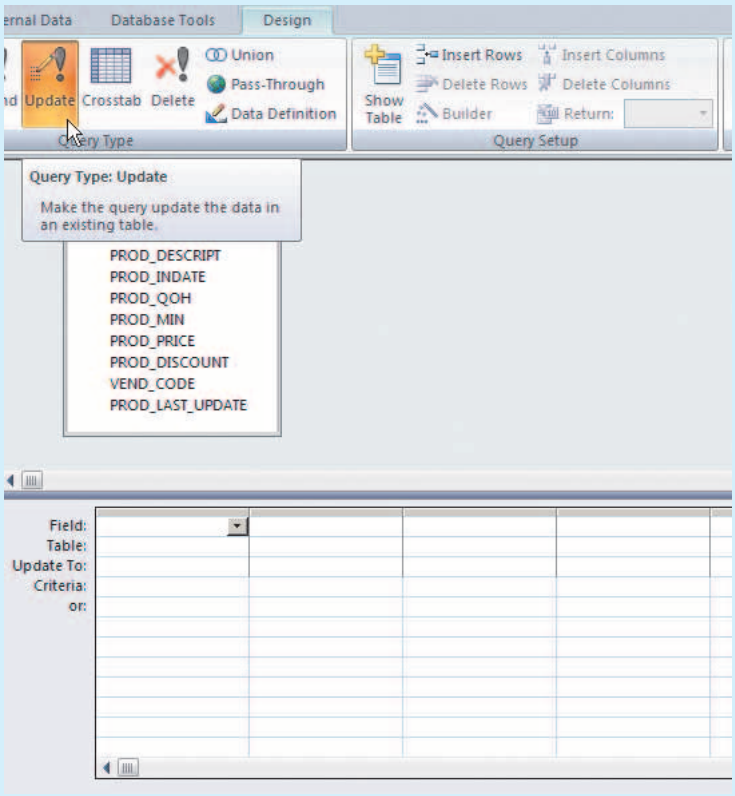
INV_NUMBER	CUST_CODE	INV_DATE	INV_AMOUNT	INV_TAX	INV_TOTAL	INV_AMT_PAID	INV_BALANCE	Add New Field
1001	10014	16-Mar-06	$24.94	$2.00	$26.94	$20.00	$6.94	
1002	10011	16-Mar-06	$9.98	$0.80	$10.78	$10.87	$0.00	
1003	10012	16-Mar-06	$153.85	$12.31	$166.16	$100.00	$66.16	
1004	10011	17-Mar-06	$34.87	$2.79	$37.66	$37.66	$0.00	
1005	10018	17-Mar-06	$70.44	$5.64	$76.08	$76.08	$0.00	
1006	10014	17-Mar-06	$397.83	$31.83	$429.66	$300.00	$129.66	
1007	10015	17-Mar-06	$34.97	$2.80	$37.77	$37.77	$0.00	
1008	10011	17-Mar-06	$399.15	$31.93	$431.08	$431.08	$0.00	
1009	10019	27-Mar-06	$180.74	$14.46	$195.20	$0.00	$0.00	
0	0		$0.00	$0.00	$0.00	$0.00	$0.00	

SOURCE: Course Technology/Cengage Learning

2.1.5 UPDATE QUERIES

An update query, as its name suggests, is used to update one or more attributes (fields) in a table. To illustrate the development and use of a simple update query, let's use one to alter the PRODUCT table's product price for a selected vendor. Because it is common to record the date of the update, first add a new PRODUCT table attribute named PROD_LAST_UPDATE. Use a time/date data type and a short date format. Note that this attribute shows up in Figure M.81. Next, start a new query, select the PRODUCT table, and then click the **Design** button to access the query option list shown in Figure M.81. Note that the **Update Query...** has been marked for selection.

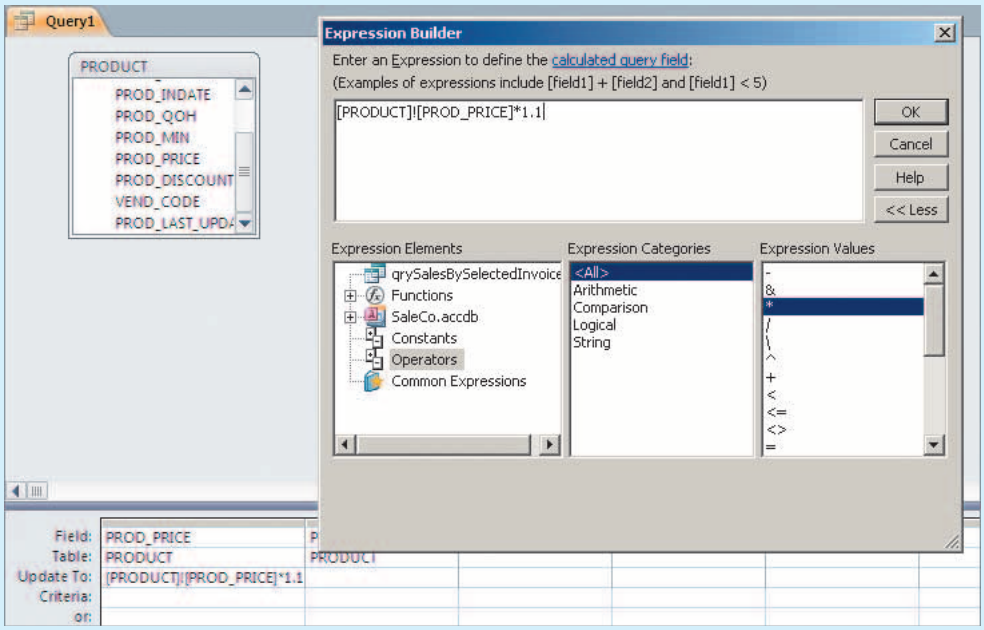
FIGURE M.81 Selecting the Update query type



SOURCE: Course Technology/Cengage Learning

Go ahead and add the PRODUCT table to the query (remember: Right-click, Show Table). Next, drag and drop the PROD_PRICE and PROD_LATE_UPDATE to the field locations as shown in Figure M.82. There will be two updates: The PROD_PRICE will be increased by 10% for a selected vendor and the current date of the update will be recorded. As you can tell by looking at Figure M.82, the **Expression Builder** was selected to produce the price change. (Although these updates are simple enough to type into the **Update To:** space directly, use the Expression Builder on all of the updates to make sure that the Expression Builder becomes a thoroughly familiar tool. (You will also have to update the table to enter the current date into the PROD_LATE_UPDATE field and to create a parameter entry for the selected vendor.)

FIGURE M.82 Using the Expression Builder for the update



SOURCE: Course Technology/Cengage Learning

As you examine the PROD_PRICE update statement in Figure M.82, note that the asterisk indicates a multiplication. You can either type the asterisk or you can click the asterisk button shown in Figure M.82. Type the value 1.10 to indicate the 10% increase.

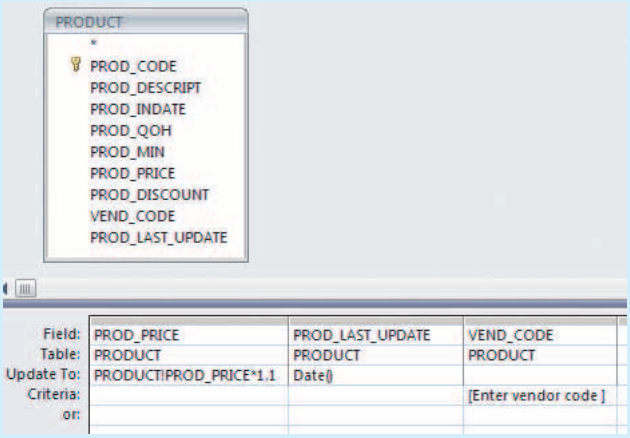
Because the update is probably too broad—it includes all vendors—modify the update query by adding a parameter entry that will generate a dialog box. Figure M.83 shows the modification that will yield a dialog box that will request the vendor code for the update. Note also that the date of the update is included. (The system date is provided by the **Date()** function.)

Save the query as **qryUpdateProductPriceAndDateOfUpdate-SelectedVendor** and then close the query. On the right side of the screen, under the Queries section, you will see the new query listed as shown in Figure M.84. (Note that the update query is identified by a special marker.)

Before you run the update query you have just created and saved, take a look at the current prices for vendor 21344, shown in Figure M.85.

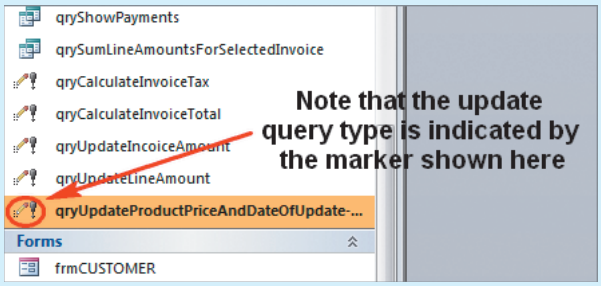
Next, run the **qryUpdateProductPriceAndDateOfUpdate-SelectedVendor** and note the effect of doing so. Because the update query modifies the table contents, Access will give you several opportunities to change your mind. Figure M.86 shows the first of the warnings.

FIGURE M.83 Completed update requirements



SOURCE: Course Technology/Cengage Learning

FIGURE M.84 Update query marker



SOURCE: Course Technology/Cengage Learning

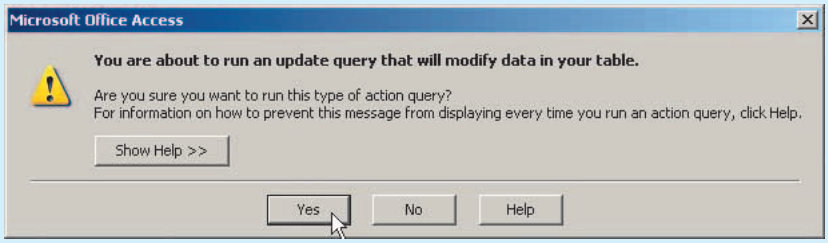
FIGURE M.85 Prices for vendor 21344 before update

PROD_CODE	PROD_DESCRIPT	PROD_INDATE	PROD_QOH	PROD_MIN	PROD_PRICE	PROD_DISCOUNT	VEND_CODE	PROD_LAST_UPDATE
11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-11	8	5	$109.99	0.00	25595	
13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-11	24	15	$14.99	0.05	21344	
14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-11	18	12	$17.49	0.00	21344	
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-12	14	8	$39.95	0.00	23119	
1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-12	23	5	$43.99	0.00	23119	
2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-11	7	5	$109.92	0.05	24288	
2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-11	6	5	$99.87	0.05	24288	
2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-12	11	5	$38.95	0.05	25595	
23109-HB	Claw hammer	20-Jan-12	18	10	$9.95	0.10	21225	
23114-AA	Sledge hammer, 12 lb.	02-Jan-12	8	5	$14.40	0.05		
54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-11	37	20	$4.99	0.00	21344	
89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-12	11	5	$256.99	0.05	24288	
PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-12	171	75	$5.87	0.00		
SM-18277	1.25-in. metal screw, 25	01-Mar-12	169	75	$6.99	0.00	21225	
SW-23116	2.5-in. wd. screw, 50	24-Feb-12	237	100	$8.45	0.00	21231	
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-12	15	5	$119.95	0.10	25595	
			0	0	$0.00	0.00	0	

SOURCE: Course Technology/Cengage Learning

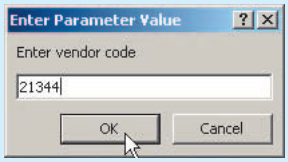
If you click the **Yes** button shown in Figure M.86, your update query will generate the parameter request through the dialog box you see in Figure M.87. Note that the vendor code **21344** has been entered to match the display in Figure M.85.

FIGURE M.86 Update query warning



SOURCE: Course Technology/Cengage Learning

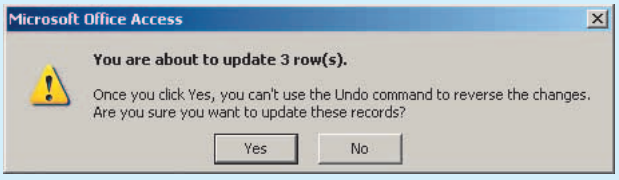
FIGURE M.87 Parameter value request



SOURCE: Course Technology/Cengage Learning

As soon as you click the **OK** button shown in Figure M.87's parameter request, the update query will be launched. But before it actually updates the table, Access gives you one final warning and a way to back out of the update. Note that the warning shown in Figure M.88 shows that you will be updating three rows. (If you check Figure M.85 again, you'll see that this is the correct number of rows.)

FIGURE M.88 Final update warning



SOURCE: Course Technology/Cengage Learning

If you now click the **Yes** button shown in Figure M.88, the update query will make the requested updates. Open the PRODUCT table after running the update query and check the results. Figure M.89 shows that the changes were made as requested.

Compare the original prices shown in Figure M.85 to those shown in Figure M.89. Note that the initial price of PROD_CODE = 13-Q2P2—supplied by vendor 21344—was $14.99. The updated price is $14.99 * 1.10 = $16.489, which is properly rounded to $16.49 because the PROD_PRICE data type *and* format were both recorded as "currency" when the table was created. Also note that the update was made on February 21, 2012. Incidentally, you can record the time of the update, using the MS Access **Time()** function for a new field named PROD_UPDATE_TIME. You can even record the identity of the person who made the updates through an authorization code that may have been recorded on a scanning device of some sort. By recording the update date, time, and person you can track all changes and assign proper responsibility.

Using Update Queries to Manage Transactions

Using the update query techniques you have just learned, you can manage a sales transaction. For example, suppose that customer 10019 had bought 20 units of product SW-23116 @ $8.45 each and 2 units of product PVC23DRT @ $5.87 each on March 27, 2012. Let's suppose you have already entered the following INVOICE table values:

- INV_NUMBER = 1009
- CUST_CODE = 10019
- INV_DATE = 27 March 2012. (Actually, you could use an update query to enter the current date of the transaction, but go ahead and enter the date manually to keep the process simple.)

FIGURE M.89 **Prices for vendor 21344 after the update**

PROD_CODE	PROD_DESCRIPT	PROD_INDATE	PROD_QOH	PROD_MIN	PROD_PRICE	PROD_DISCOUNT	VEND_CODE	PROD_LAST_UPDATE	Add
11QER/31	Power painter, 15 psi., 3-nozzle	03-Nov-11	8	5	$109.99	0.00	25595		
13-Q2/P2	7.25-in. pwr. saw blade	13-Dec-11	24	15	$16.49	0.05	21344	21-Feb-12	
14-Q1/L3	9.00-in. pwr. saw blade	13-Nov-11	18	12	$19.24	0.00	21344	21-Feb-12	
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	15-Jan-12	14	8	$39.95	0.00	23119		
1558-QW1	Hrd. cloth, 1/2-in., 3x50	15-Jan-12	23	5	$43.99	0.00	23119		
2232/QTY	B&D jigsaw, 12-in. blade	30-Dec-11	7	5	$109.92	0.05	24288		
2232/QWE	B&D jigsaw, 8-in. blade	24-Dec-11	6	5	$99.87	0.05	24288		
2238/QPD	B&D cordless drill, 1/2-in.	20-Jan-12	11	5	$38.95	0.05	25595		
23109-HB	Claw hammer	20-Jan-12	18	10	$9.95	0.10	21225		
23114-AA	Sledge hammer, 12 lb.	02-Jan-12	8	5	$14.40	0.05			
54778-2T	Rat-tail file, 1/8-in. fine	15-Dec-11	37	20	$5.49	0.00	21344	21-Feb-12	
89-WRE-Q	Hicut chain saw, 16 in.	07-Feb-12	11	5	$256.99	0.05	24288		
PVC23DRT	PVC pipe, 3.5-in., 8-ft	20-Feb-12	171	75	$5.87	0.00			
SM-18277	1.25-in. metal screw, 25	01-Mar-12	169	75	$6.99	0.00	21225		
SW-23116	2.5-in. wd. screw, 50	24-Feb-12	237	100	$8.45	0.00	21231		
WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	17-Jan-12	15	5	$119.95	0.10	25595		
			0	0	$0.00	0.00	0		

SOURCE: Course Technology/Cengage Learning

Next, open the LINE table to record the following two sets of LINE table entries. (Note that you are not entering the LINE_AMOUNT values.)

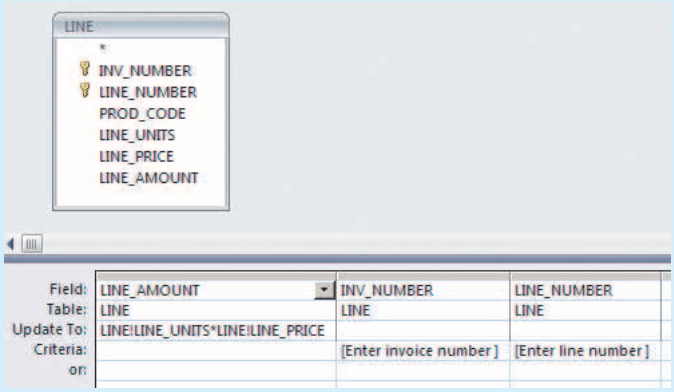
- INV_NUMBER = 1009
- LINE_NUMBER = 1
- PROD_CODE = SW-23116
- LINE_UNITS = 20
- LINE_PRICE = $8.45. (Actually, you could have used an update query to copy the PROD_PRICE from the PRODUCT table to the LINE table, but go ahead and enter the price manually to keep the process simple.)
- INV_NUMBER = 1009
- LINE_NUMBER = 2
- PROD_CODE = PVC23DRT
- LINE_UNITS = 2
- LINE_PRICE = $5.87

After completing the just-described tasks, create an update query to calculate and enter the LINE_AMOUNT value for invoice 1009 and line numbers 1 and 2. The correct update query is shown in Figure M.90. (You did remember to use the **Expression Builder**, didn't you?)

When you run the query shown in Figure M.90, you will be prompted to enter the invoice number 1009 and the line number 1 and the query will properly execute to produce the first line total. Next, run the query again, this time entering the invoice number 1009 and the line number 2. When you are done, open the LINE table to see if the amounts were calculated properly. Figure M.91 indicates that they were.

You can use additional update queries to calculate the invoice subtotal, the tax, and the total. Then enter the amount paid, $100.00, and then use an update query to calculate the balance due and the customer balance. At this point, the INVOICE table's INV_NUMBER 1009 shows no entry for any of the values that are yet to be calculated. (See Figure M.92.)

FIGURE M.90 Update query to calculate LINE_AMOUNT



SOURCE: Course Technology/Cengage Learning

FIGURE M.91 The calculated LINE_AMOUNT values

INV_NUMBI	LINE_NUMBE	PROD_COI	LINE_UNI	LINE_PRI	LINE_AMOUI
1001	1	13-Q2/P2	1	$14.99	$14.99
1001	2	23109-HB	1	$9.95	$9.95
1002	1	54778-2T	2	$4.99	$9.98
1003	1	2238/QPD	1	$38.95	$38.95
1003	2	1546-QQ2	1	$39.95	$39.95
1003	3	13-Q2/P2	5	$14.99	$74.95
1004	1	54778-2T	3	$4.99	$14.97
1004	2	23109-HB	2	$9.95	$19.90
1005	1	PVC23DRT	12	$5.87	$70.44
1006	1	SM-18277	3	$6.99	$20.97
1006	2	2232/QTY	1	$109.92	$109.92
1006	3	23109-HB	1	$9.95	$9.95
1006	4	89-WRE-Q	1	$256.99	$256.99
1007	1	13-Q2/P2	2	$14.99	$29.98
1007	2	54778-2T	1	$4.99	$4.99
1008	1	PVC23DRT	5	$5.87	$29.35
1008	2	WR3/TT3	3	$119.95	$359.85
1008	3	23109-HB	1	$9.95	$9.95
1009	1	SW-23116	20	$8.45	$169.00
1009	2	PVC23DRT	2	$5.87	$11.74
0	0		0	$0.00	$0.00

SOURCE: Course Technology/Cengage Learning

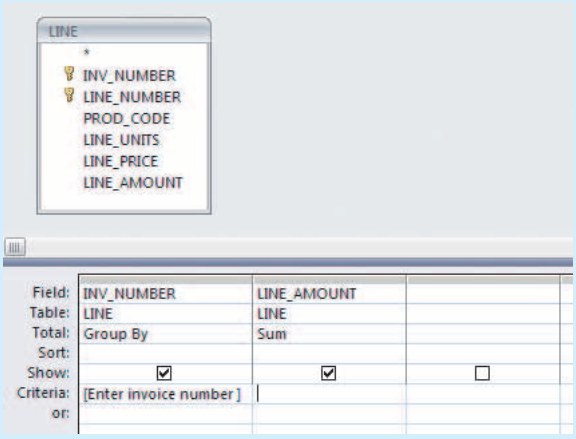
FIGURE M.92 The INVOICE table values before the updates

INV_NUMBER	CUST_CODE	INV_DATE	INV_AMOUNT	INV_TAX	INV_TOTAL	INV_AMT_PAID	INV_BALANCE	Add New Field
1001	10014	16-Mar-12	$24.94	$2.00	$26.94	$20.00	$6.94	
1002	10011	16-Mar-12	$9.98	$0.80	$10.78	$10.87	$0.00	
1003	10012	16-Mar-12	$153.85	$12.31	$166.16	$100.00	$66.16	
1004	10011	17-Mar-12	$34.87	$2.79	$37.66	$37.66	$0.00	
1005	10018	17-Mar-12	$70.44	$5.64	$76.08	$76.08	$0.00	
1006	10014	17-Mar-12	$397.83	$31.83	$429.66	$300.00	$129.66	
1007	10015	17-Mar-12	$34.97	$2.80	$37.77	$37.77	$0.00	
1008	10011	17-Mar-12	$399.15	$31.93	$431.08	$431.08	$0.00	
1009	10019	27-Mar-12	$0.00	$0.00	$0.00	$0.00	$0.00	
0	0		$0.00	$0.00	$0.00	$0.00	$0.00	

SOURCE: Course Technology/Cengage Learning

Keep in mind that you cannot get a total for a set of values in a query and then use those values during the same query run. For example, to get the sales for invoice 1009, you will have to add $169.00 and $11.74 and then use that sum to update the INVOICE table's INV_AMOUNT. Figure M.93 shows a simple parameter query to generate the sum of the line amounts for the selected invoice number. (Note that this query used the sum function to get its job done. *This is not an update query*.)

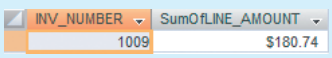
FIGURE M.93 Sum the line amounts



SOURCE: Course Technology/Cengage Learning

Save the query shown in Figure M.93 as **qrySumLineAmountsForSelectedInvoice** and run it to check its execution. Figure M.94 indicates that the query performed its job properly. (Note that the sum of the last two LINE_AMOUNT values in Figure M.91 should be $169.00 + $11.74 = $180.74.)

FIGURE M.94 The correct line amount sum



SOURCE: Course Technology/Cengage Learning

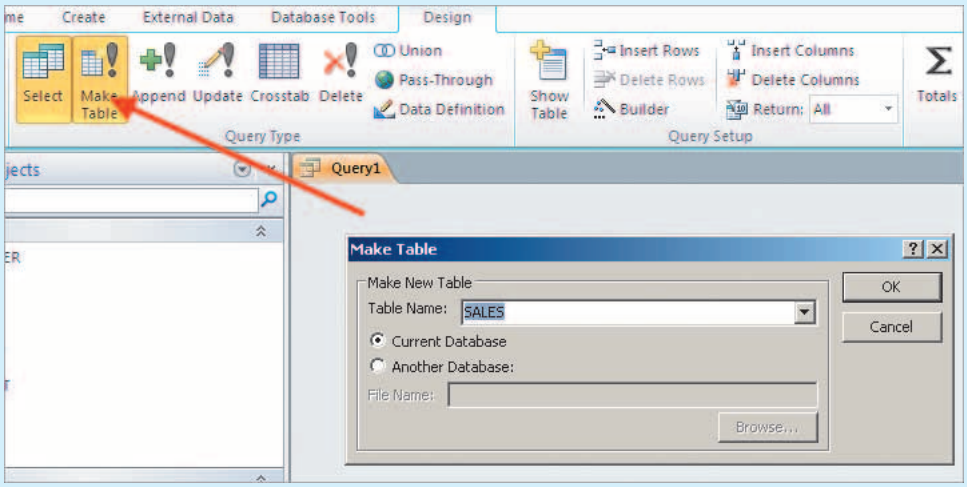
There are many ways to enter the $180.74 value into the INVOICE table. (Although you can create an update query that uses the results of the parameter query in Figure M.93 to update the INVOICE table, you will learn in Section 5.1 how to use some simple, but more effective ways to get that job done.) However, since this section deals with various query types, let's take a look at some other queries that will turn out to be useful.

2.1.6 MAKE TABLE QUERIES

One of the very useful MS Access features is its ability to create new tables based on other tables or even other queries. You should now be familiar with the basic query development routine. Therefore, to create a **Make Table** query, start as you did before. That is, first select the **Query Design** option under the Create tab. If the **Show Table** dialog box is shown on the screen, close it and then use the **Design** button at the top of the screen to produce the query option list. Select the **Make Table Query...** option to generate the **Make Table** dialog box you see in Figure M.95. You will create a SALES table to store the results of running the **qrySumLineAmountsForSelectedInvoice** you saw in Figures M.93 and M.94, so name the new table SALES, as shown in Figure M.95.

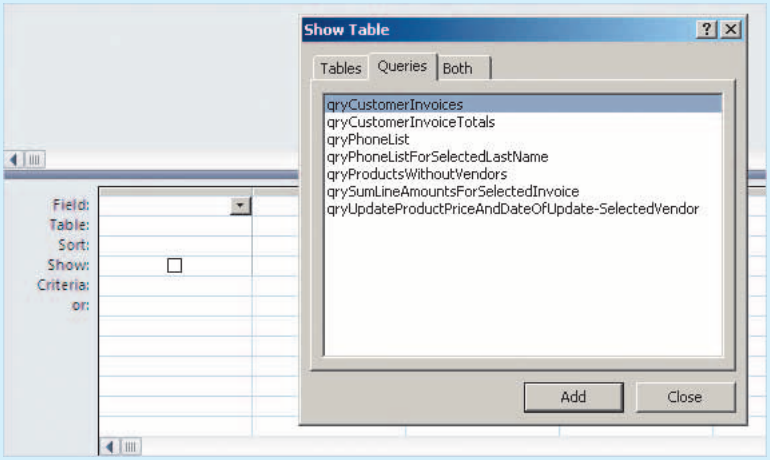
Next, select the data source for the new table as shown in Figure M.96. In this case, the data source will be the **qrySumLineAmountsForSelectedInvoice** query, so select the **Queries** tab and then select the query from the list.

FIGURE M.95 Make Table query



SOURCE: Course Technology/Cengage Learning

FIGURE M.96 Selecting the data source



SOURCE: Course Technology/Cengage Learning

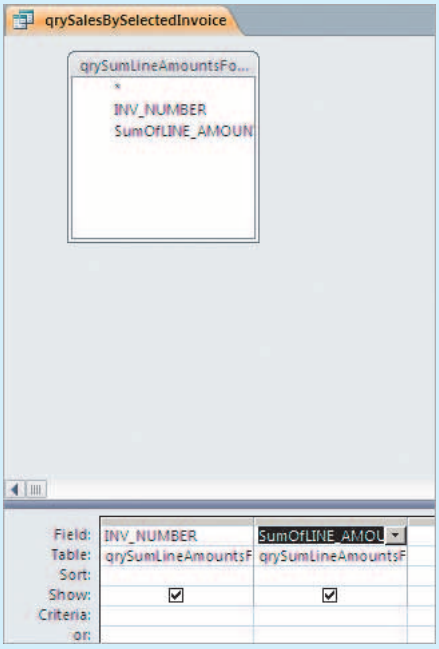
Adding the query will place it on the design screen. Next, drag and drop the query fields on the Field: lines as shown in Figure M.97. Make sure that you save the just-completed **Make Table** query. (As you can tell by looking at Figure M.97, the query has been saved as **qrySalesBySelectedInvoice**.

Note that the updated query list in Figure M.98 includes the new query.

When you run the new **qrySalesBySelectedInvoice** query, Access will alert you to the fact that this query will modify data in a table. (See Figure M.99. The new SALES table does not yet contain any data, but that is about to change.)

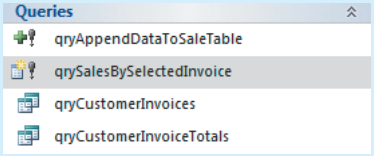
If you accept the fact that the query will modify the data in the new SALES table, click the **Yes** button shown in Figure M.99 to generate the invoice number entry request in Figure M.100.

FIGURE M.97 Completed Make Table query



SOURCE: Course Technology/Cengage Learning

FIGURE M.98 Current query list

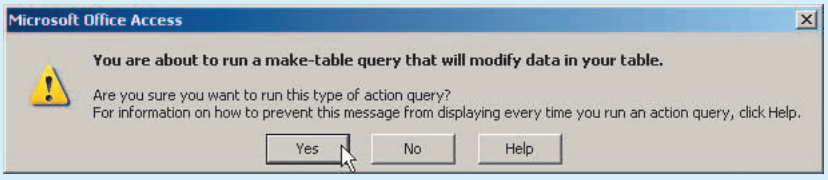


SOURCE: Course Technology/Cengage Learning

Enter the invoice number 1009 as shown in Figure M.100 to produce the warning shown in Figure M.101. Click the **Yes** button to complete the **Make Table** action.

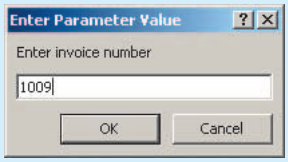
The results are shown in Figure M.102. Note that the SALES table has been added to the list of tables and that the table contents show the invoice number 1009 and the sales (invoice) amount $180.74.

FIGURE M.99 First Make Table warning



SOURCE: Course Technology/Cengage Learning

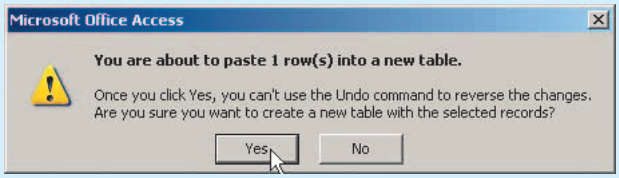
FIGURE M.100 Invoice number selection



SOURCE: Course Technology/Cengage Learning

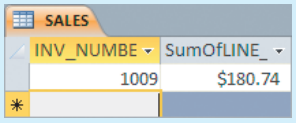
Now that you have a good (SALES table) data source, which is the sum of the invoice lines for INV_NUMBER 1009 in the INVOICE table, you can easily create an update query to update the INVOICE table's INV_AMOUNT. Note that Figures M.103 and M.104 show the required update query structure and the results of running that query.

FIGURE M.101 Make Table warning



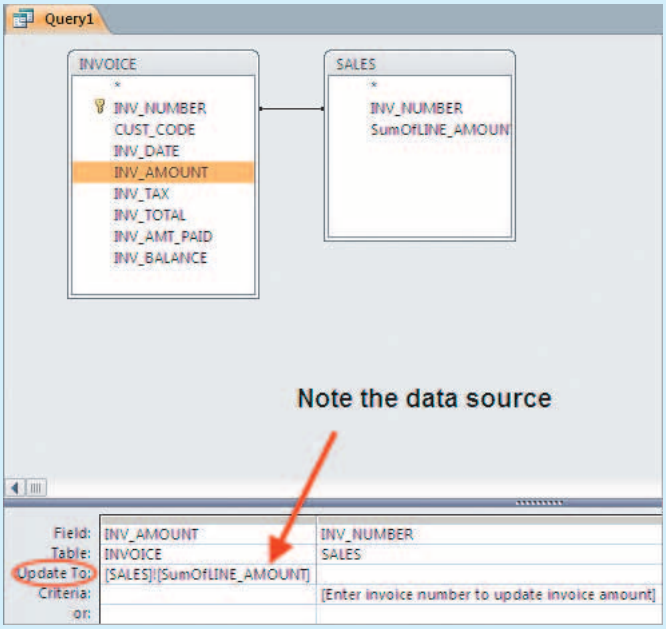
SOURCE: Course Technology/Cengage Learning

FIGURE M.102 The new SALES table



SOURCE: Course Technology/Cengage Learning

FIGURE M.103 Update invoice amount query



SOURCE: Course Technology/Cengage Learning

2.1.7 Append Queries

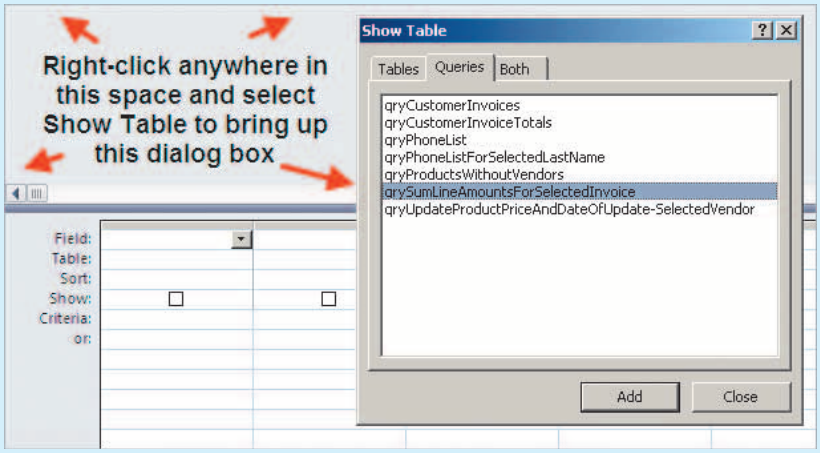
Append queries do what their name suggests: they append data to an existing table. This feature is very useful, because—as you saw in the preceding example—the new SALES table contents are likely to be used to store the invoice amounts that can be used to update the INVOICE table's INV_AMOUNT values via an update query.

FIGURE M.104 Updated invoice amount

	INV_NUMBER	CUST_CODE	INV_DATE	INV_AMOUNT	INV_TAX	INV_TOTAL	INV_AMT_PAID	INV_BALANCE	Add New Field
⊞	1001	10014	16-Mar-12	$24.94	$2.00	$26.94	$20.00	$6.94	
⊞	1002	10011	16-Mar-12	$9.98	$0.80	$10.78	$10.87	$0.00	
⊞	1003	10012	16-Mar-12	$153.85	$12.31	$166.16	$100.00	$66.16	
⊞	1004	10011	17-Mar-12	$34.87	$2.79	$37.66	$37.66	$0.00	
⊞	1005	10018	17-Mar-12	$70.44	$5.64	$76.08	$76.08	$0.00	
⊞	1006	10014	17-Mar-12	$397.83	$31.83	$429.66	$300.00	$129.66	
⊞	1007	10015	17-Mar-12	$34.97	$2.80	$37.77	$37.77	$0.00	
⊞	1008	10011	17-Mar-12	$399.15	$31.93	$431.08	$431.08	$0.00	
⊞	1009	10019	27-Mar-12	$180.74	$0.00	$0.00	$0.00	$0.00	
*	0	0		$0.00	$0.00	$0.00	$0.00	$0.00	

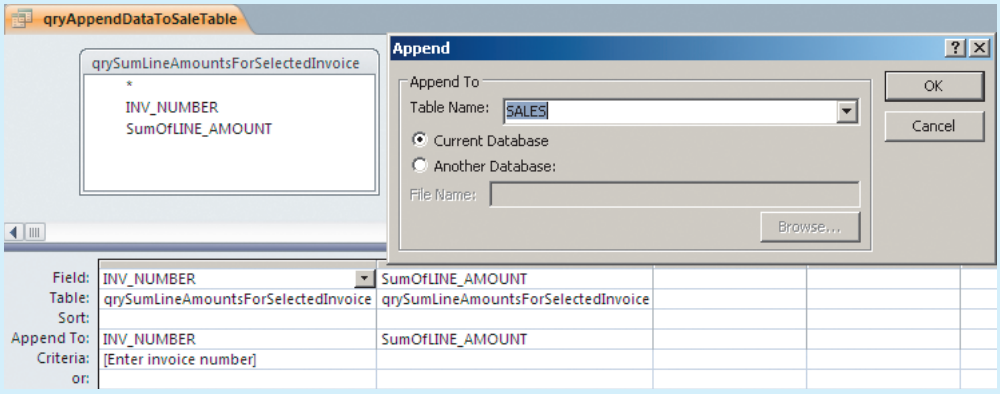
SOURCE: Course Technology/Cengage Learning

FIGURE M.105 Append query design



SOURCE: Course Technology/Cengage Learning

FIGURE M.106 Design view of completed append query



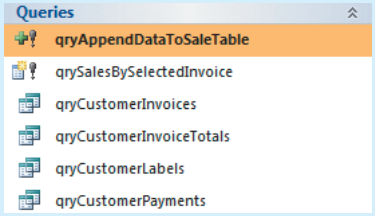
SOURCE: Course Technology/Cengage Learning

To create an **Append Query**, follow the now familiar new query design steps. However, this time, select the **Append Query** from the query type list. In the following sequence—shown in Figures M.105 and M.106—you will see how an append query can be used to add data to an existing table. There are two important limitations you should know about when you append data to an existing table:

- For each field, the data type of the data to be appended must be identical to the existing data type of the data in the receiving table.
- For each field, the field name of the data to be appended must be identical to the existing data field name of the data in the receiving table.

Save the new append query as **qryAppendDateToSaleTable** and then close the query. You will now see the new query as shown in Figure M.107.

FIGURE M.107 The saved append query



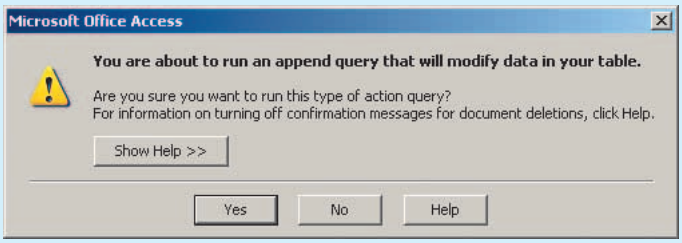
SOURCE: Course Technology/Cengage Learning

Running the new append query will cause Access to provide the appropriate warnings and the parameter request as shown in Figures M.108, M.109, and M.110.

You can check the results of running the append query by checking the SALES table. Figure M.111 shows that the append query performed its work as intended.

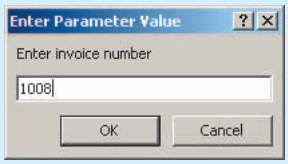
You can now run the **qryUpdateInvoiceAmountQuery** again—see Figures M.103 and M.104—to enter the INV_AMOUNT into the INVOICE table for invoice 1008. (Actually, that entry was already made manually before as you can see in Figure M.39's INVOICE table, so this $399.15 value from the SALE table simply replaces that value.)

FIGURE M.108 First append query action warning



SOURCE: Course Technology/Cengage Learning

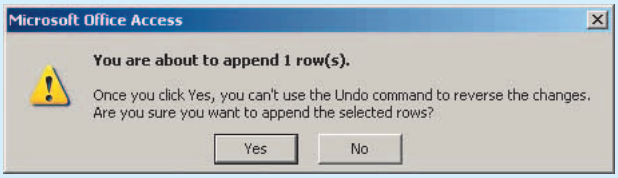
FIGURE M.109 Parameter entry



SOURCE: Course Technology/Cengage Learning

At this point, you have learned how to use various query types to store, update, and transfer data. Therefore, you have the basic tools at your disposal to help you automate the sales transaction process. For example, you can now create update queries to update the remaining fields in the INVOICE table, to reduce the QOH values in the PRODUCT table, to update the CUST_BALANCE in the CUSTOMER table, and so on. Macros or Visual Basic code can then be used to complete the automation process by executing the queries "behind the scenes." Figures M.112 through M.115 show several update queries and their effects.

FIGURE M.110 Final append action warning



SOURCE: Course Technology/Cengage Learning

FIGURE M.111 SALE table append result

INV_NUMBER	SumOfLINE_AMOUNT
1009	$180.74
1008	$399.15

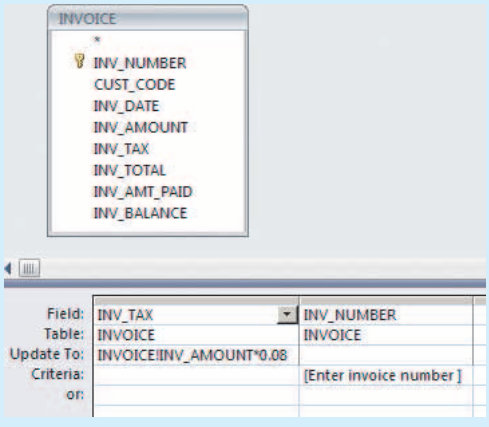
SOURCE: Course Technology/Cengage Learning

FIGURE M.112 INVOICE tax and total before update

INV_NUMBER	CUST_CODE	INV_DATE	INV_AMOUNT	INV_TAX	INV_TOTAL	INV_AMT_PAID	INV_BALANCE
1001	10014	16-Mar-12	$24.94	$2.00	$26.94	$20.00	$6.94
1002	10011	16-Mar-12	$9.98	$0.80	$10.78	$10.87	$0.00
1003	10012	16-Mar-12	$153.85	$12.31	$166.16	$100.00	$66.16
1004	10011	17-Mar-12	$34.87	$2.79	$37.66	$37.66	$0.00
1005	10018	17-Mar-12	$70.44	$5.64	$76.08	$76.08	$0.00
1006	10014	17-Mar-12	$397.83	$31.83	$429.66	$300.00	$129.66
1007	10015	17-Mar-12	$34.97	$2.80	$37.77	$37.77	$0.00
1008	10011	17-Mar-12	$399.15	$31.93	$431.08	$431.08	$0.00
1009	10019	27-Mar-12	$180.74	$0.00	$0.00	$0.00	$0.00
0	0		$0.00	$0.00	$0.00	$0.00	$0.00

SOURCE: Course Technology/Cengage Learning

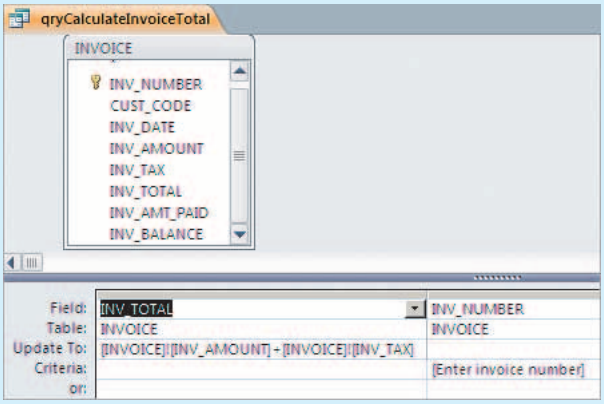
FIGURE M.113 Calculate invoice tax update query



SOURCE: Course Technology/Cengage Learning

Keep in mind that all the queries must be run in the proper order. For example, you cannot calculate the sales tax before you have the updated invoice total. And the invoice total cannot be calculated unless all the invoice line totals have been calculated and summed. The table values will become meaningless if the queries were executed in the wrong order. Therefore, it would not be wise to trust humans to remember the proper sequence. Fortunately, you can use programming techniques or macros to automate the process. (You will learn how to create and use macros in Section 5.)

FIGURE M.114 Calculate invoice total update query



SOURCE: Course Technology/Cengage Learning

FIGURE M.115 Updated invoice tax and total

	INV_NUMBER	CUST_CODE	INV_DATE	INV_AMOUNT	INV_TAX	INV_TOTAL	INV_AMT_PAID	INV_BALANCE
⊞	1001	10014	16-Mar-12	$24.94	$2.00	$26.94	$20.00	$6.94
⊞	1002	10011	16-Mar-12	$9.98	$0.80	$10.78	$10.87	$0.00
⊞	1003	10012	16-Mar-12	$153.85	$12.31	$166.16	$100.00	$66.16
⊞	1004	10011	17-Mar-12	$34.87	$2.79	$37.66	$37.66	$0.00
⊞	1005	10018	17-Mar-12	$70.44	$5.64	$76.08	$76.08	$0.00
⊞	1006	10014	17-Mar-12	$397.83	$31.83	$429.66	$300.00	$129.66
⊞	1007	10015	17-Mar-12	$34.97	$2.80	$37.77	$37.77	$0.00
⊞	1008	10011	17-Mar-12	$399.15	$31.93	$431.08	$431.08	$0.00
⊞	1009	10019	27-Mar-12	$180.74	$14.46	$195.20	$0.00	$0.00
*	0	0		$0.00	$0.00	$0.00	$0.00	$0.00

SOURCE: Course Technology/Cengage Learning

NOTE

Although the update and append queries can be very useful in managing transactions such as invoicing and letting customers make payments on their accounts, there are other techniques available that will perform the transaction management job more efficiently. You will learn how to use a **Set Value** technique to manage customer payments on account and to automate the process through macros in Section 5.

3.1 FORMS

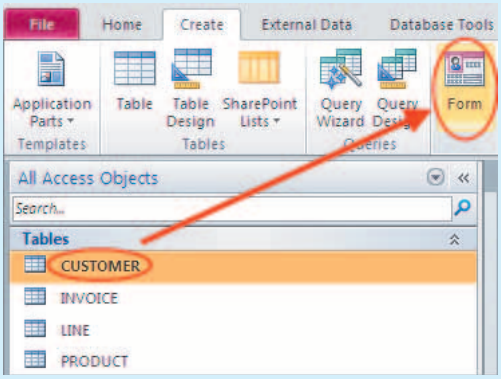
While queries let you get data and/or information from the database, forms let you control the presentation format much better. In addition, forms will enable you to control data input and to present the results from multiple queries and/or tables. Forms can also be used to tie the application components together through menus and other devices. In short, forms are the way in which the end user is best connected to the database … and they provide that "professional" look to your data management efforts.

Forms may be based on tables and/or queries. The simplest and most efficient way to create a form is through a ***form wizard***, which lets you automatically generate the code that produces the form. Figure M.116 illustrates the

use of one of the many form wizards. Note that the form production process in Figure M.116 is set in motion through the following sequence:

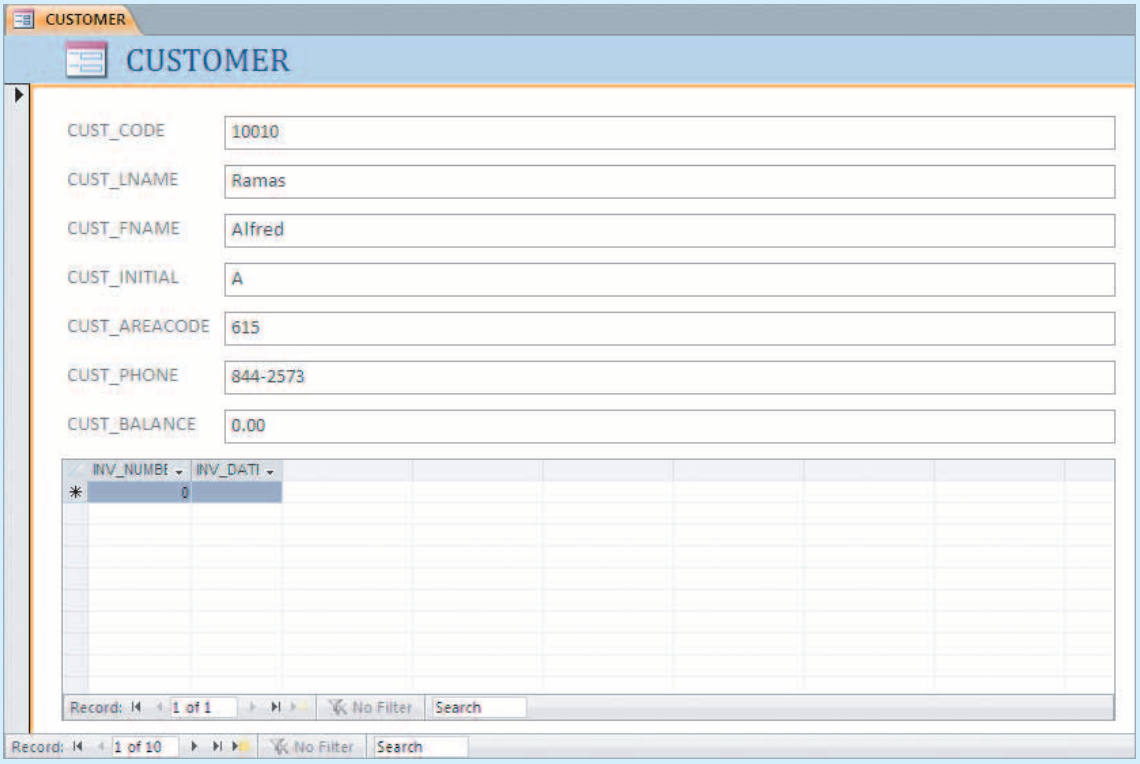
1. Select the **Create** tab on the ribbon.
2. Select the **Form** option to create the form shown in Figure M.117.

FIGURE M.116 Starting a new form



SOURCE: Course Technology/Cengage Learning

FIGURE M.117 The Access-generated CUSTOMER form

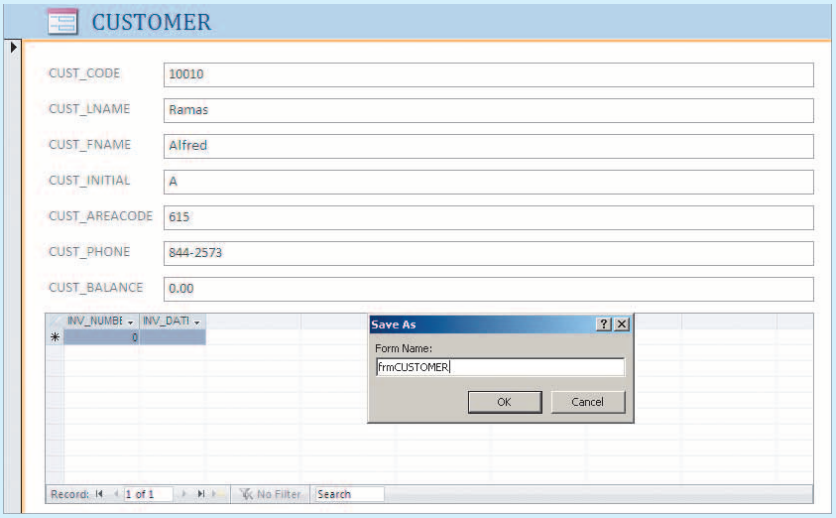


SOURCE: Course Technology/Cengage Learning

As you can see by looking at Figure M.117, the form opens up in its **Layout View** format. (Given the work you have done with the query development, you should be familiar with the various views.)

Save the form before continuing the form design process. Figure M.118 shows that the new form was named **frmCUSTOMER**.

FIGURE M.118 Save the new CUSTOMER form



SOURCE: Course Technology/Cengage Learning

NOTE

Here is another example of self-documentation: the **frm** prefix indicates that the object is a form and the capital letters used in the **CUSTOMER** portion indicate that the query data source was a ***table*** named CUSTOMER. If the form's data source had been a ***query*** named **qryCustomer**, the form name would have indicated the data source through the use of lowercase letters, using "caps" only to separate the components. Therefore, if the form had been based on the **qryCustomer** query, the form name would have been **frmCustomer**. In either case, looking at the object name would immediately tell you that the object is a form and that it displays customer information.

Making it easy to keep track of the many components in a set of database applications is a mark of professionalism. Nobody wins if nobody can figure out what the database objects are or what they do. ***Documentation conventions should be made clear in the formal application documentation.***

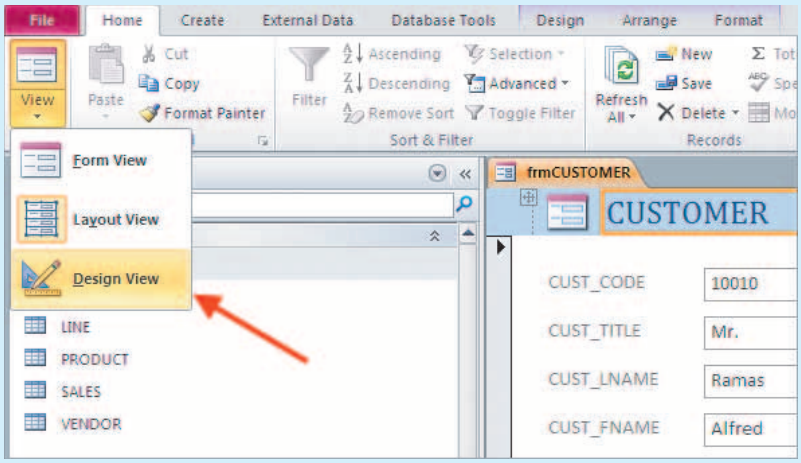
3.1.1 EDITING THE FORM

While the **frmCUSTOMER** form presents the data properly, it is a good idea to modify the form's format for eye-appeal reasons. Your expertise is often graded on the basis of how "professional" your forms look. Therefore, put the form in its **Design View** format, as shown in Figure M.119.

The first time you open the form in its Design format, you will see the attribute list shown in Figure M.120. All the CUSTOMER table's attributes have already been included in the form, so go ahead and close this field list. (If you later want to open this field list box again, click the **Add Existing Fields** button shown in Figure M.120.)

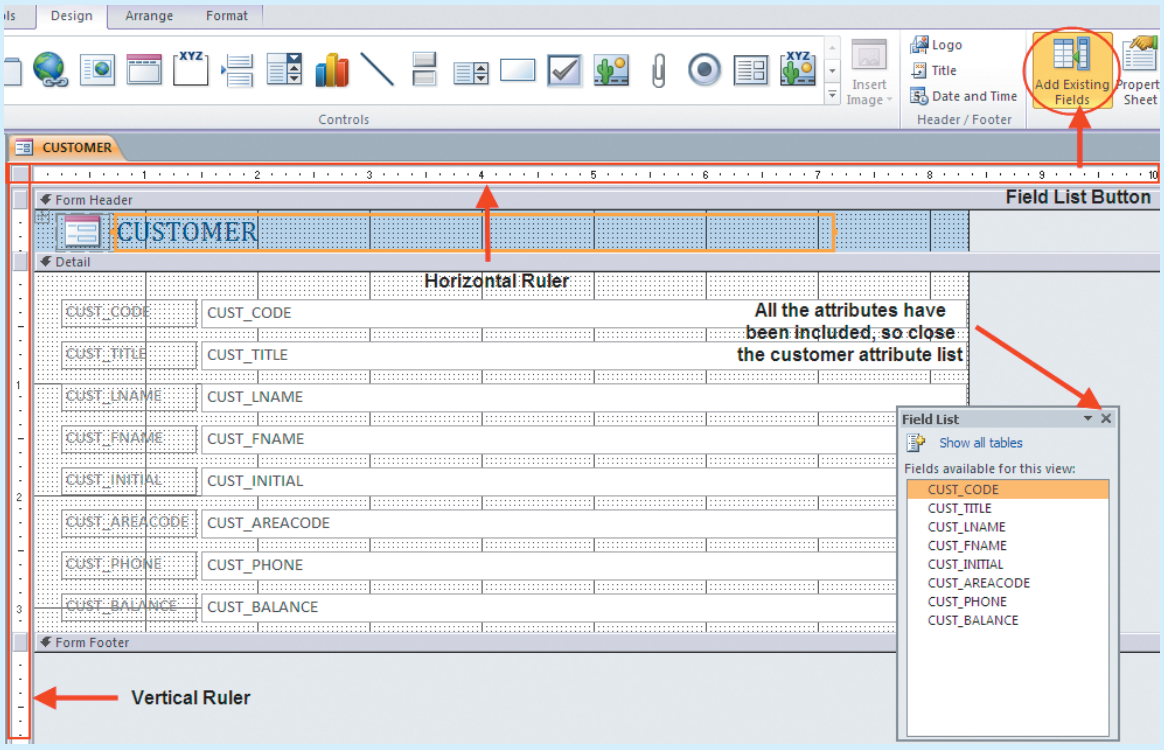
If you want to widen the form, put the mouse pointer on the form's edge, and then drag the form limit to wherever you want it to be. You can control the vertical size the same way. Just put the cursor on the ***top edge*** of the **Form Footer** and drag up or down to suit your needs. If you put the cursor on the ***bottom edge*** of the **Form Footer**, you will

FIGURE M.119 Change the CUSTOMER form to Design View



SOURCE: Course Technology/Cengage Learning

FIGURE M.120 Control the form width

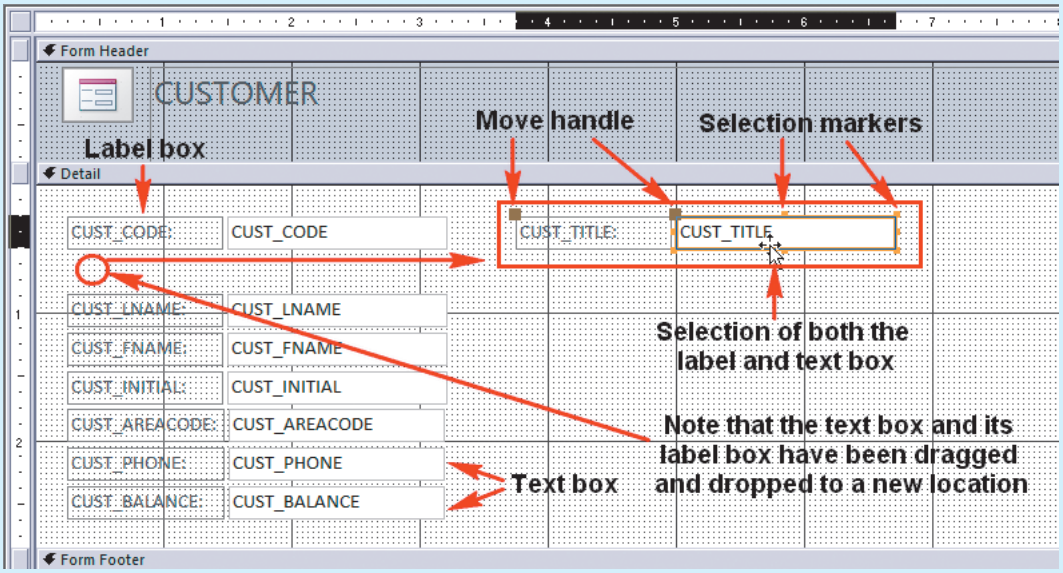


SOURCE: Course Technology/Cengage Learning

be able to create a footer and control its width. (Incidentally, the horizontal and vertical rulers can be used later to help you line up selected output components or to mark multiple selection on the form.)

Figure M.121 shows that the form limits have been dragged down and to the right to produce a larger form surface on which to place the form contents. Also note the various design components that have been marked on the figure.

FIGURE M.121 **Design components**



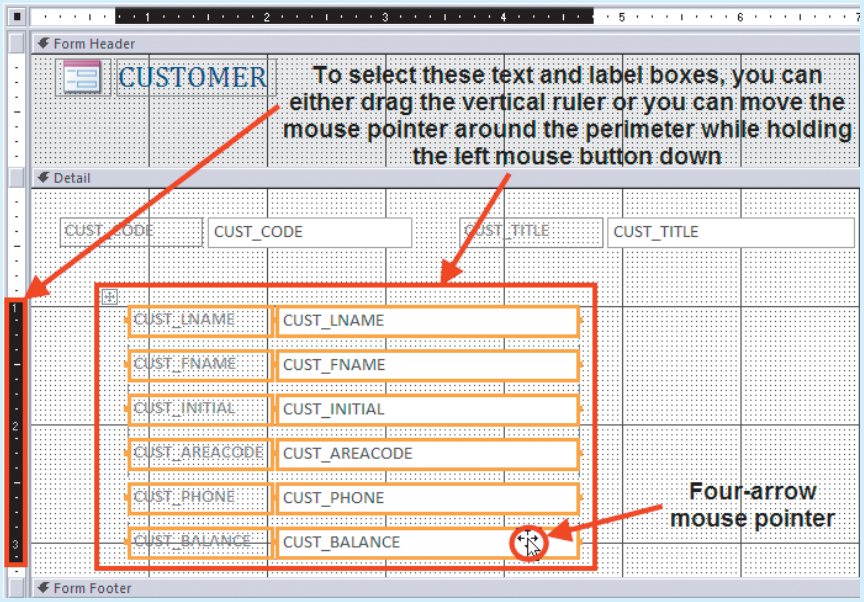
SOURCE: Course Technology/Cengage Learning

As you examine Figure M.121, keep the following points in mind:

- Clicking the text box portion selects both the text box and its label box as a unit.
- Clicking the label box selects only the label box.
- To create the "open hand" cursor, put the mouse pointer on the text box and then move the pointer to the lower or upper limit of the text box to change its shape. (The pointer is rather sensitive, so move it slowly until it changes.)
- The open hand indicates that both the text box and its associated label box have been selected as a unit. If you hold the left mouse button down, you can drag the text box ***and its associated label box*** to any location.
- If you put the mouse pointer on a move handle, it will change to a pointing finger. Note that there are two move handles, one for the text box and one for the label box. (This mouse pointer is not shown in Figure M.121, because you can only show one pointer position at a time. However, go ahead and place your mouse pointer over a move handle and watch the pointer change to pointing finger.)
- If you drag with the move handle, only the component marked by that move handle will move. Therefore, you can move the text and label boxes to different locations. For example, you can place the label box above the text box.
- If you place the mouse pointer over a selection marker, its shape will change to a two-headed arrow. You can drag on the selection markers to change the text and label box sizes.

If you want to select all the text and label boxes at once, you can drag the mouse pointer along the rulers. You can also select multiple text/label boxes by moving the pointer while holding down the left mouse button. Once the block of multiple text/label boxes has been selected, you can place the mouse pointer on any marked text box to change its shape to an open hand—you can then drag the block to any position on your form. (See Figure M.122 to see a multiple selection.)

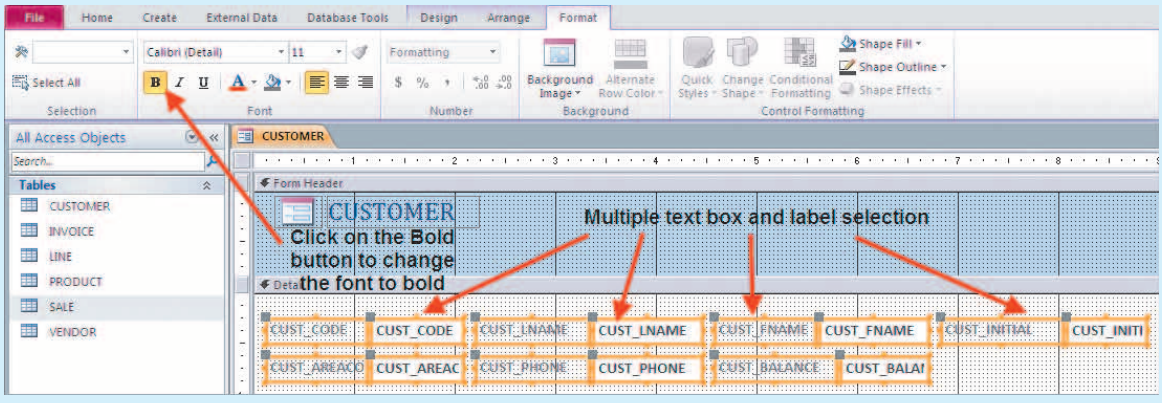
FIGURE M.122 Multiple component selection



SOURCE: Course Technology/Cengage Learning

Practice moving the form components around until they approximately match Figure M.123. Then change the fonts to Bold as shown in Figure M.123.

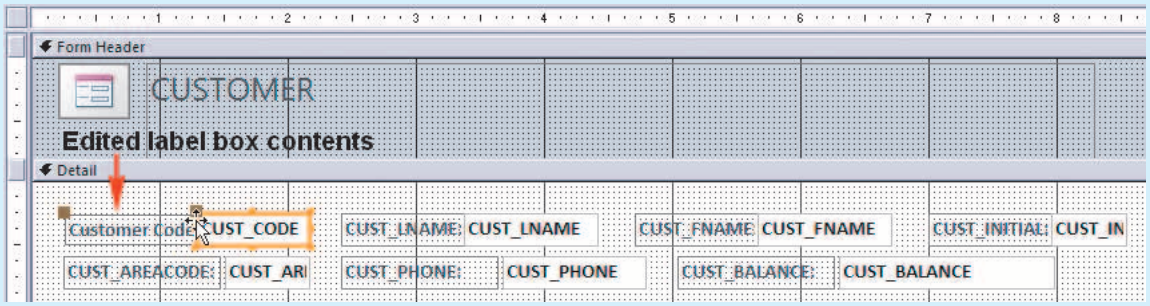
FIGURE M.123 Font set to bold



SOURCE: Course Technology/Cengage Learning

You can also change the label box contents by selecting the label box and then clicking that selected label box to put it in Edit mode. You can then edit label text. Note that Figure M.124 shows that the CUST_CODE was changed to **Customer code**. (Remember that the label box width can be changed by dragging its limits.) Note also that the mouse pointer was placed over the text box move handle to change the pointer shape to a four-headed arrow—you can now drag the text box to line it up with the edited label box.

FIGURE M.124 Editing the label boxes



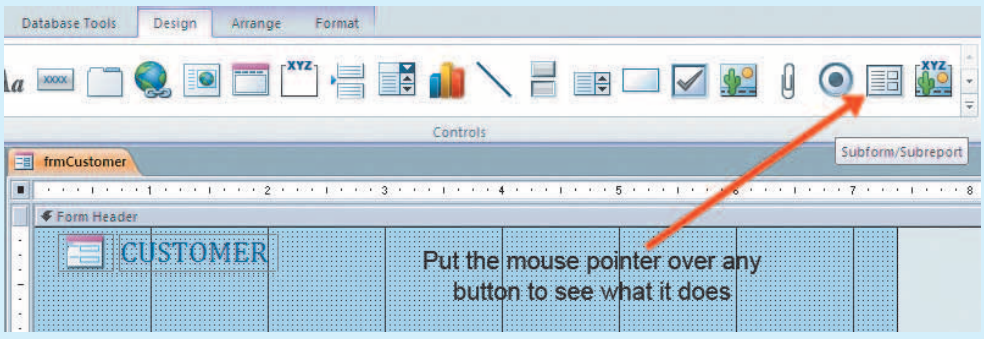
SOURCE: Course Technology/Cengage Learning

You have a large number of form design tools available. For example, you can add additional text, create boxes to delineate form components, and so on. Click **Design** to view the **Controls** group you see in Figure M.125. If you are unsure what any of these design tools does, hover the mouse pointer over a button as shown in Figure M.126.

FIGURE M.125 The Controls group

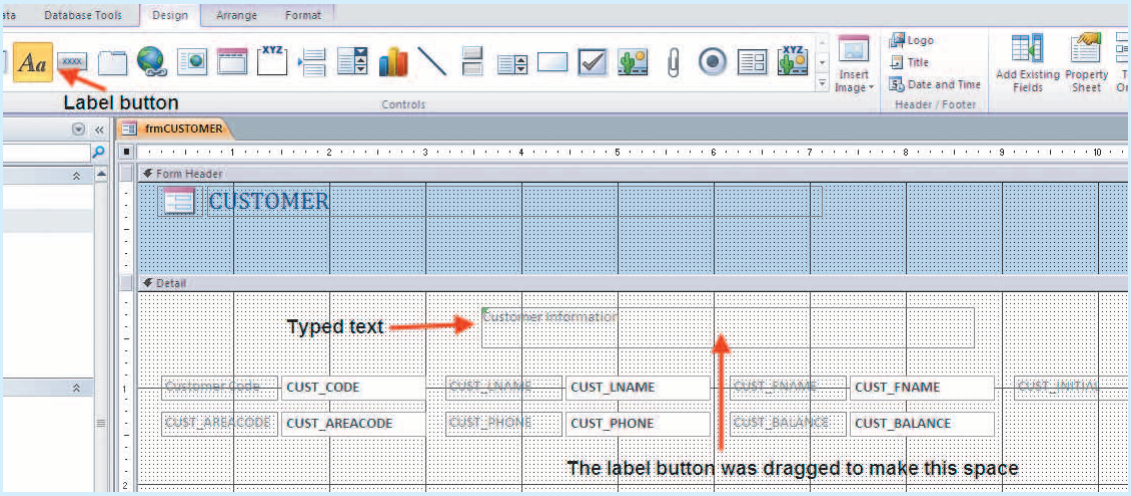
SOURCE: Course Technology/Cengage Learning

FIGURE M.126 Controls buttons



SOURCE: Course Technology/Cengage Learning

FIGURE M.127 Using the Label tool



SOURCE: Course Technology/Cengage Learning

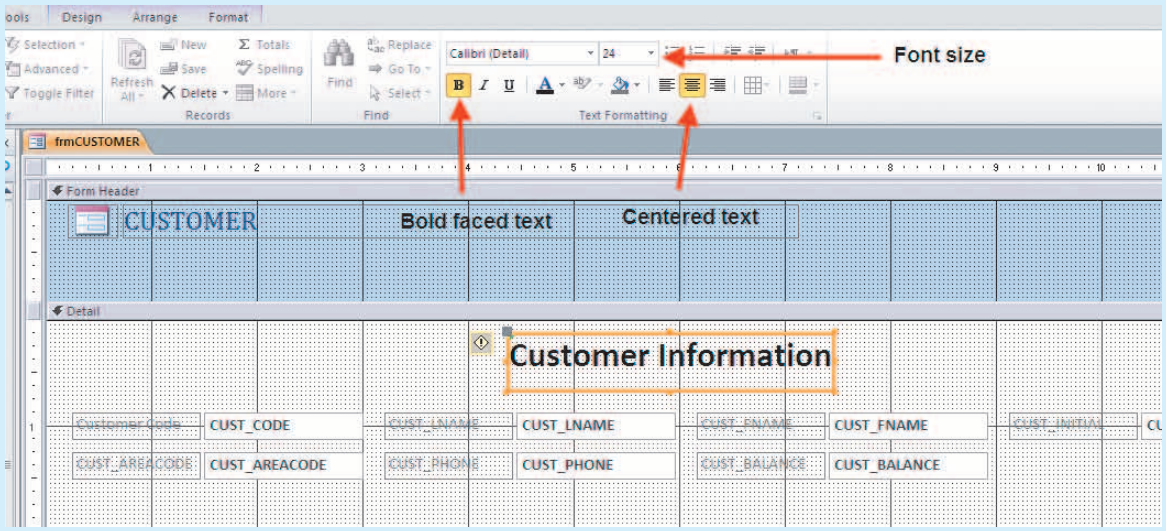
Figure M.127 also shows that the **Label** button was selected. This selection changes the pointer to the **+A** shape. If you drag this pointer anywhere on the form, you will create a text space in which you can type whatever is needed. In this example, the typed text is **Customer Information**. Click anywhere outside the text space to return the mouse pointer to its normal function.

You can edit the text as needed. Note that Figure M.128 shows that the font size was changed, the text was centered, and the label box was resized.

The form's looks can be improved with raised text box limits and color, and you can use the Rectangle tool to create logical groupings of information presented on the form. Figure M.129 shows several enhancements that you can accomplish by doing the following:

- Select the **Customer Information** text box you just edited as shown in Figure M.128. (The selection is confirmed by the selection markers—the small black squares around the text box perimeter.)
- Click the **Special Effect: Shadowed** button to generate the special effect options.
- Select the **Shadowed** option by clicking it. Note the change in the text box edge.

FIGURE M.128 Edited text



SOURCE: Course Technology/Cengage Learning

Let's take a look at just a few more form design options. For example, if you look at Figure M.130, you will see that there are quite a few color options available. To match the figure you see here, you can take the following actions:

- To make the form light blue, click any empty portion of the form to select the ***entire*** form. Next, click the **Fill/Back color** button and click the light blue color square.
- To make the **Customer Information** text box dark blue, click the text box to select it—note that Figure M.130 shows the selection markers around the text box perimeter. Next, click the **Fill/Back color** button and click the dark blue color square.
- To change the font color in the text box to white, make sure that the text box is still selected. Next, click the **Font/Fore color** button and click the white color square.

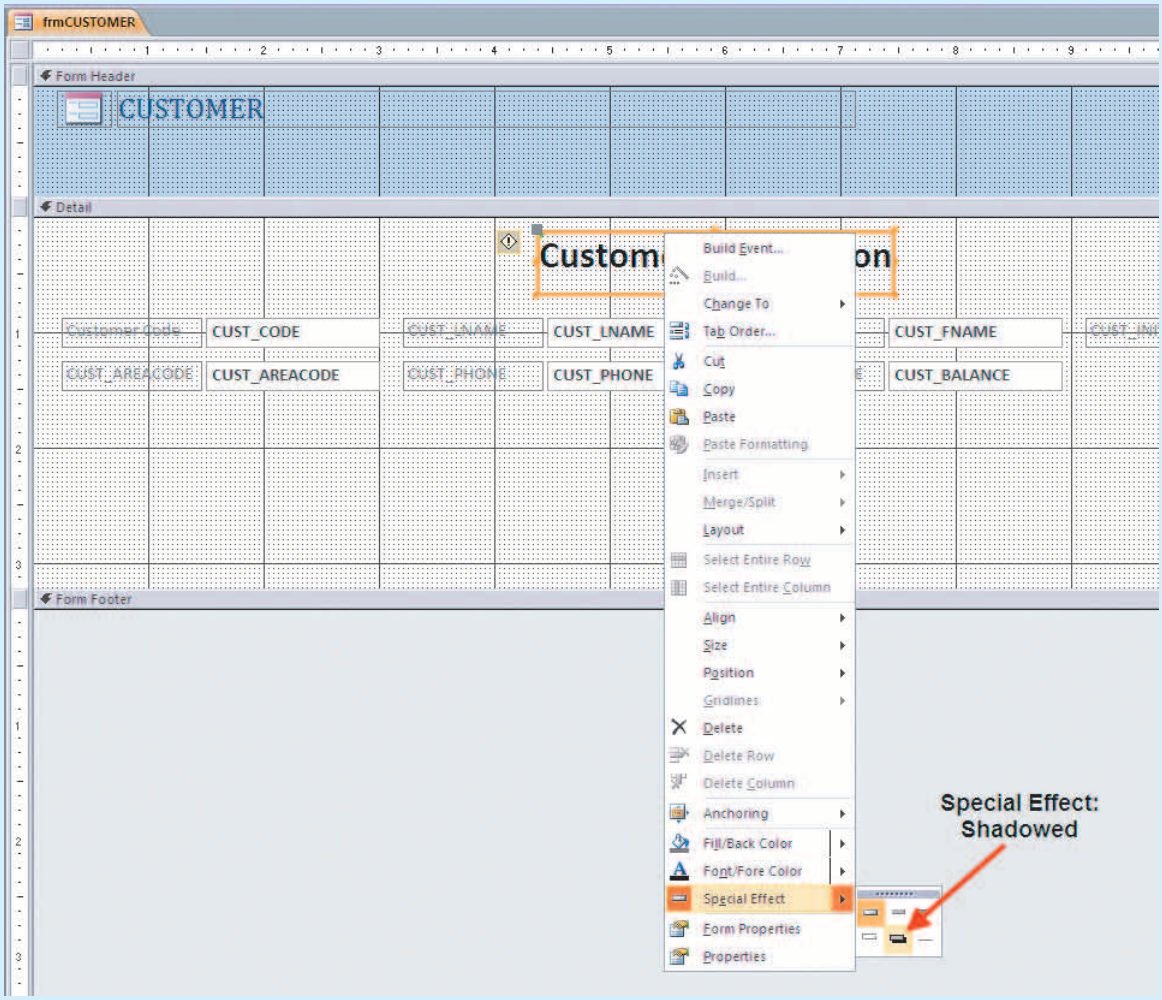
Note that Access keeps track of all the recently selected colors to make it easy to later match color selections of other form components.

It is often useful to group logically similar attributes together as a visual unit. You can use the **Rectangle tool** to draw a rectangle around such a group. To get the job done, go to the tool box and click the Rectangle tool, then drag a rectangle around the attributes you want to group. The rectangle is initially clear and just shows its outline. However, you can use the **Fill/Back color** option to give the rectangle a color. Figure M.131 shows that the selected fill color was light blue. Unfortunately, the default setting on the Rectangle tool places the rectangle in front of the attributes, thus shading them out. However, note that you can use the **Position/Send to Back** option to send the now opaque light blue rectangle to the back, thus making the attributes visible again.

Repeat the process for the CUST_BALANCE attribute, using a light green color. Finally, drag the **Customer Information** text box limits to line up with the two rectangles to balance the form. When you are done, open the form in **Form View** to see Figure M.132.

Go back to the form's Design view. Notice that right-clicking the unoccupied portion of the detail section of the form in Design view will generate the context dialog box for the detail section—see Figure M.133. Once Properties dialog box is open, you can generate the properties for any Access object by simply clicking it. Go ahead and perform this action a few times until it becomes routine.

FIGURE M.129 Additional form enhancements

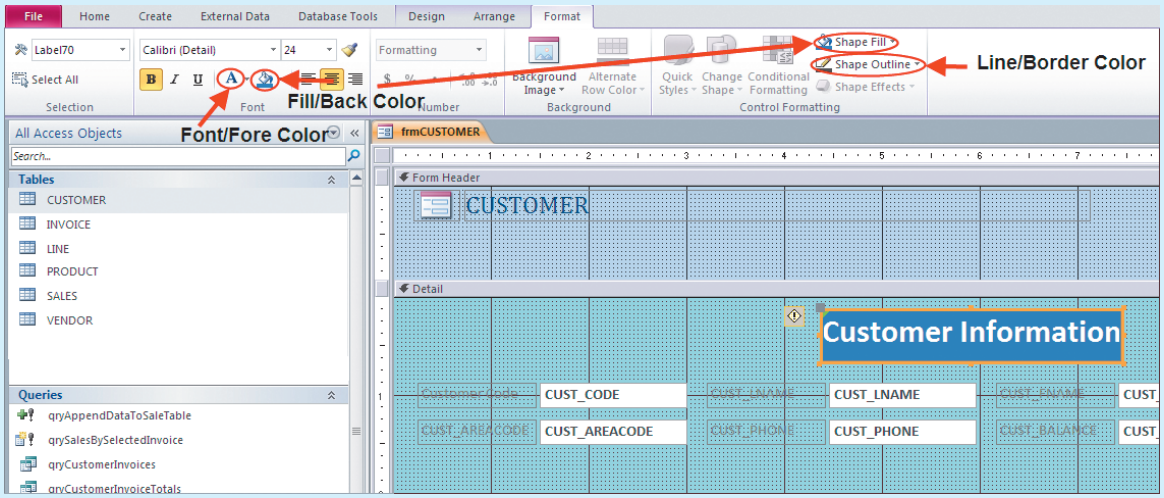


SOURCE: Course Technology/Cengage Learning

You can get the properties box for any Access object by selecting it and then right-clicking your selection. For example, if you want to see the properties of just the CUST_CODE ***text*** box, click the CUST_CODE text box to select it and then right-click. If you want to see the properties of the Customer code: ***label*** box, click the CUST_CODE text box to select it and then right-click. Figure M.134 shows the Property Sheet for the Detail section of the form.

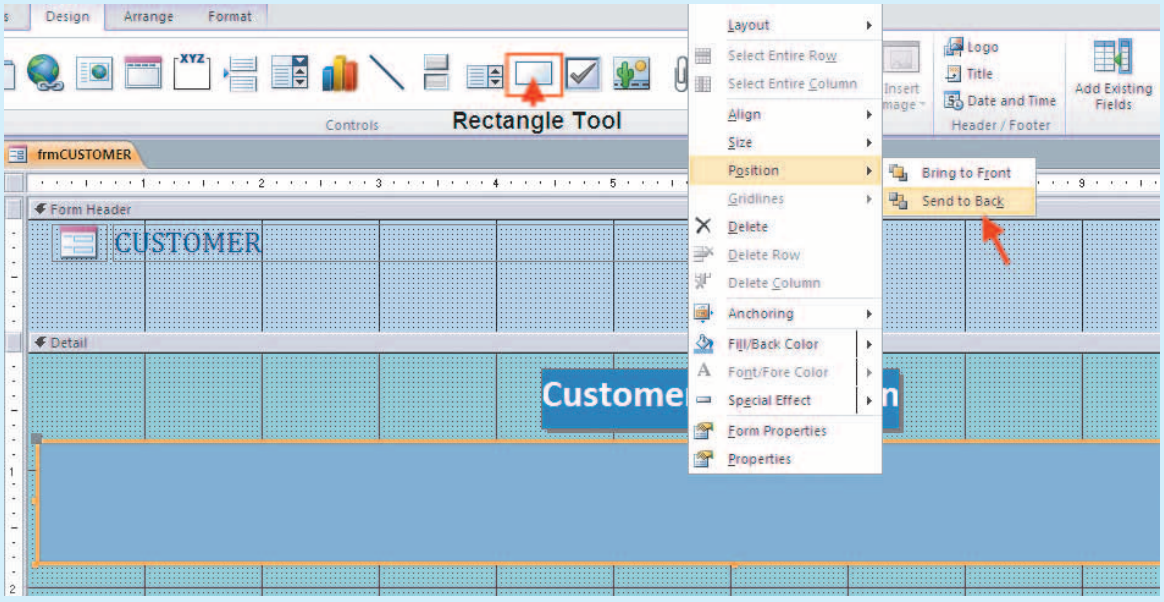
The form is essentially completed. However, you may want to put the **Customer Information** text box in the header section of the form. You can create the header space by simply dragging the form's detail limit down. (You can create the form footer by simply dragging the footer limit down.) Finally, if the white background for the text box contents are annoying to you and you want them to have the same color as the rectangles, you can select them and then use a fill color to match the background. See Figure M.135. Figure M.136 shows the **frmCUSTOMER** form in **Form view**. (If you wish, you can resize the window and the form within that window.)

FIGURE M.130 Color selections



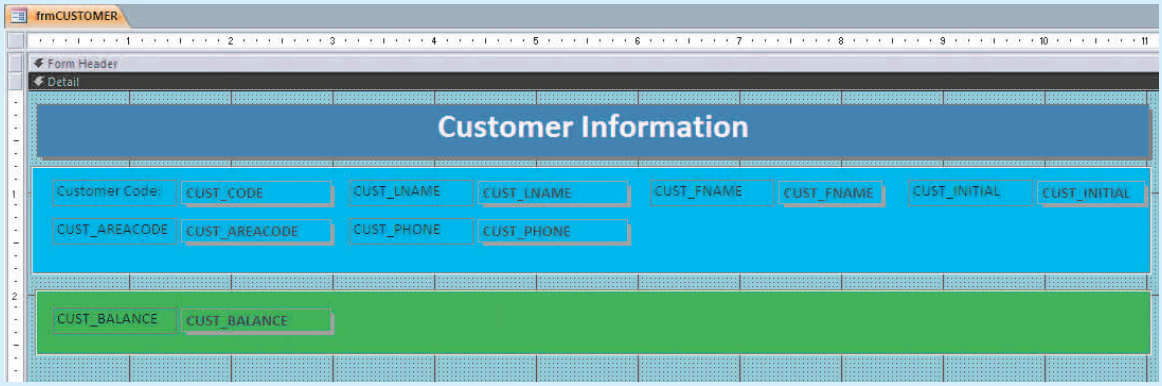
SOURCE: Course Technology/Cengage Learning

FIGURE M.131 Using the Rectangle tool



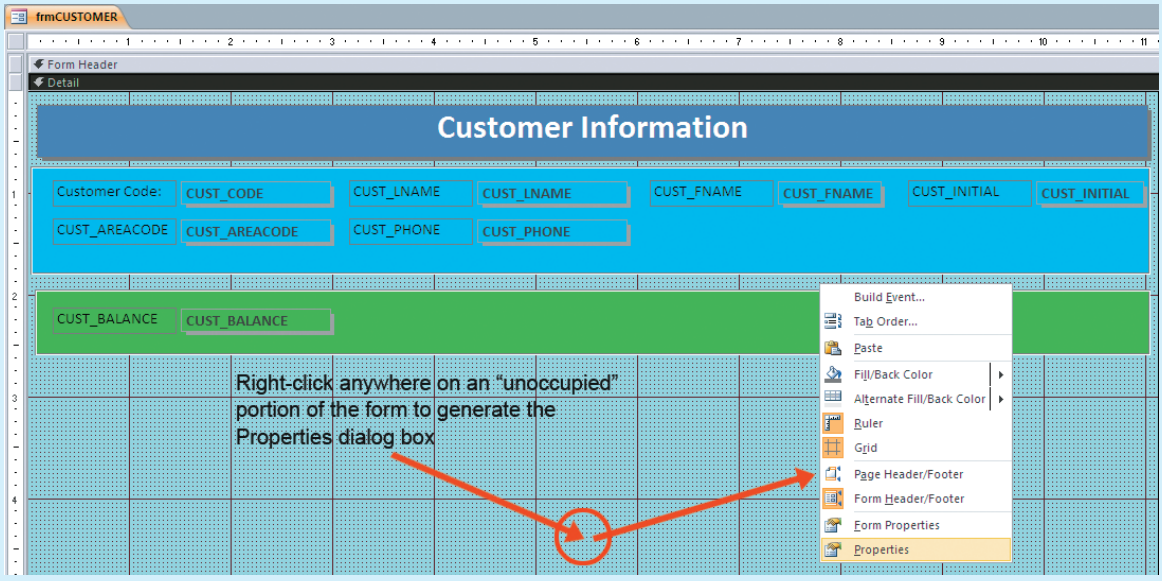
SOURCE: Course Technology/Cengage Learning

FIGURE M.132 The form in Form View



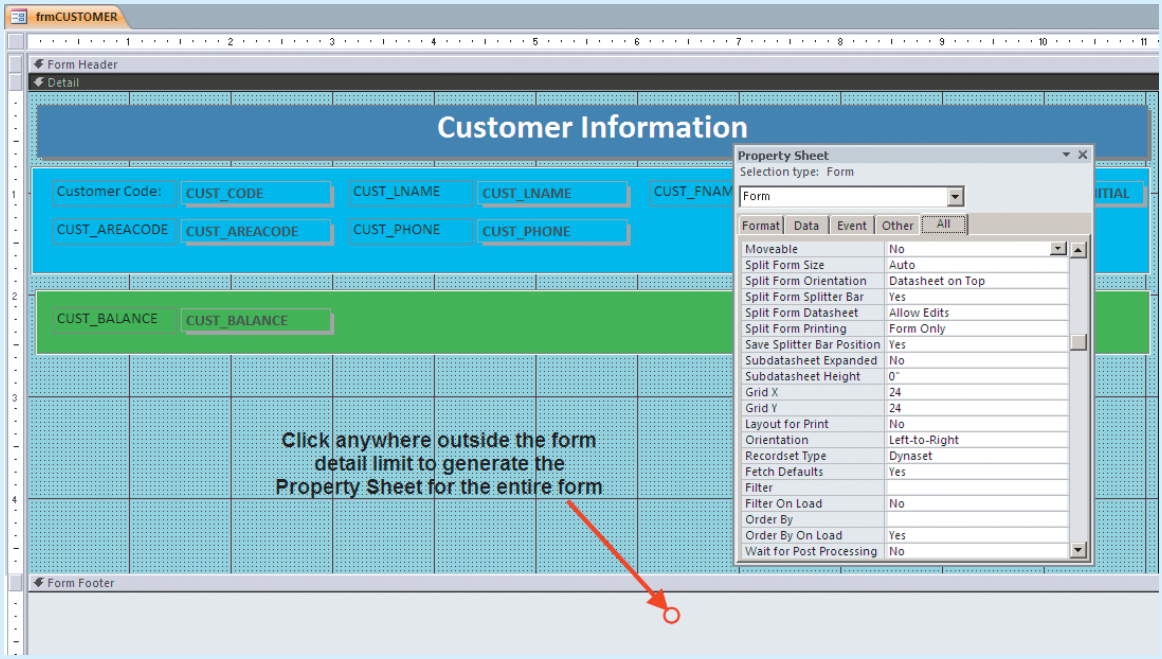
SOURCE: Course Technology/Cengage Learning

FIGURE M.133 Generating the Properties dialog box



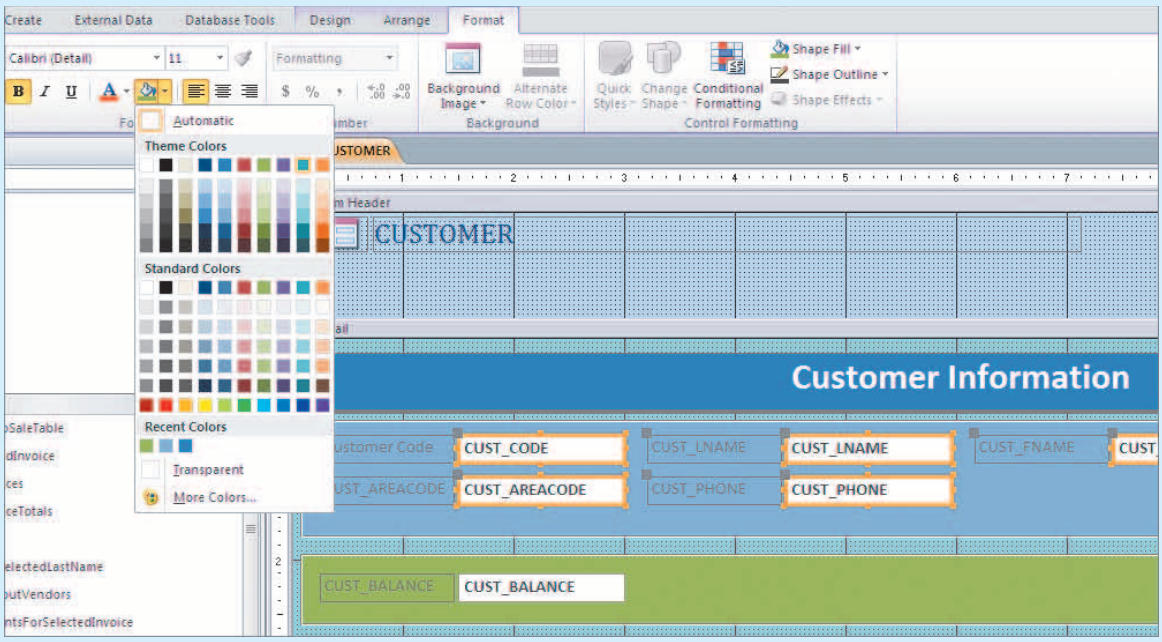
SOURCE: Course Technology/Cengage Learning

FIGURE M.134 Properties for the entire form



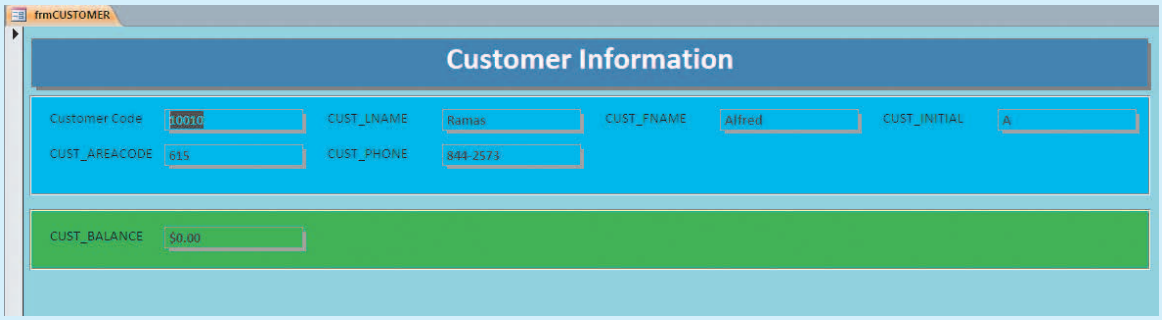
SOURCE: Course Technology/Cengage Learning

FIGURE M.135 Final touches



SOURCE: Course Technology/Cengage Learning

FIGURE M.136 The completed CUSTOMER form



SOURCE: Course Technology/Cengage Learning

3.1.2 Forms Based on Queries

Forms can be used to present data and/or information from multiple sources. Queries are particularly good as a basis for form design, because they can access multiple tables directly.

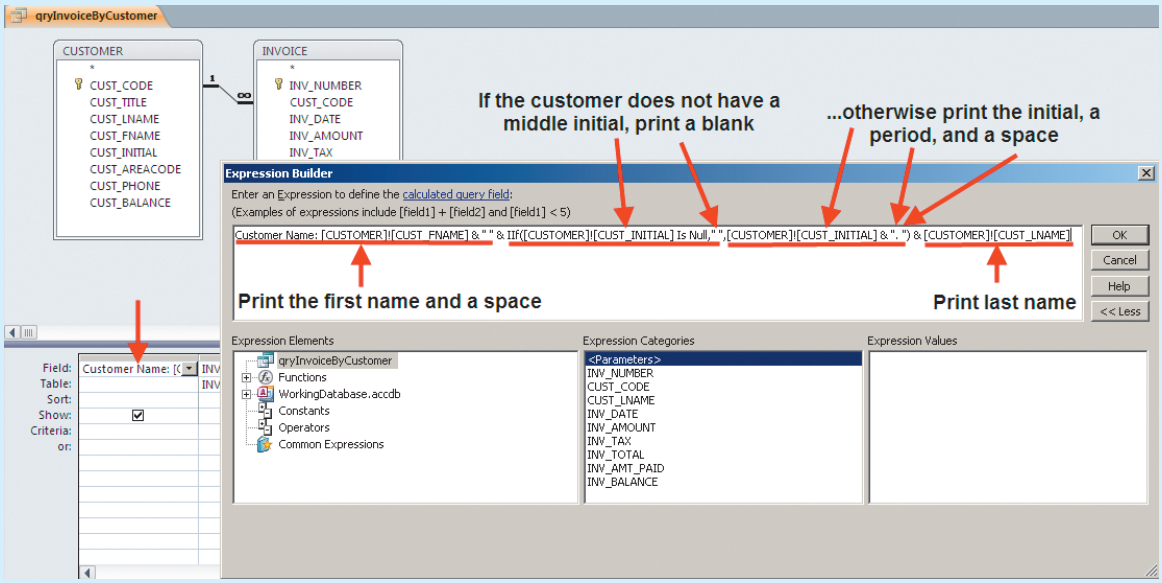
Let's create an invoice form based on the query you see in Figure M.137. Note that this query uses an expression that concatenates the customer's first name, initial, and last name. However, because some customers may not have an initial, a logical function must be used. Note the expression in the Expression Builder in Figure M.137. The logical "if and only if" (**IIf**) function uses the following format:

IIf(expression, do if true, do if false)

Note that the two possible actions are separated by commas.

In the query shown in Figure M.137, if there is no middle initial—that is, the **CUST_INITIAL Is Null**—a blank space must be printed after the customer's first name. If the customer has a middle initial—that is, the customer initial is ***not*** null—the initial must be printed, followed by a period and space. In either case, the customer's last name must be printed, so the CUST_LNAME is not included in the logical **IIf** statement. (To build this query, drag and drop the INVOICE table's INV_NUMBER and then drag and drop the CUSTOMER table's CUST_LNAME—or any of the customer's name components—to reserve a space for the expression you are about to build. Then right-click the CUST_LNAME field and select the Expression Builder to select the CUSTOMER table's CUST_FNAME, CUST_INITIAL, and CUST_LNAME—after you have deleted the CUST_LNAME that you originally dragged to the field—to build the expression. (You can type the **&**, the blank space enclosed by quotes, and the **IIf** function components.)

FIGURE M.137 The Design view of the invoice query



SOURCE: Course Technology/Cengage Learning

When you are done with the expression builder, click OK to save the expression and then drag and drop all the remaining INVOICE table fields to the QBE screen's field spaces. Save the query as **qryInvoicesByCustomer** and then open it in data sheet form as shown in Figure M.138.

FIGURE M.138 The Data Sheet view of the invoice query

INV_NUMBER	CUST_CODE	Customer Name	INV_DATE	INV_AMOUNT	INV_TAX	INV_TOTAL	INV_AMT_PAID	INV_BALANCE
1002	10011	Leona K. Dunne	16-Mar-12	$99.81	$10.78	$10.78	$10.87	$0.00
1004	10011	Leona K. Dunne	17-Mar-12	$34.87	$2.79	$37.66	$37.66	$0.00
1008	10011	Leona K. Dunne	17-Mar-12	$399.15	$31.93	$431.08	$431.08	$0.00
1003	10012	Kathy W. Smith	16-Mar-12	$153.85	$12.31	$166.16	$100.00	$66.16
1001	10014	Myron Orlando	16-Mar-12	$24.94	$2.00	$26.94	$20.00	$6.94
1006	10014	Myron Orlando	17-Mar-12	$397.83	$31.83	$429.66	$300.00	$129.66
1007	10015	Amy B. O'Brian	17-Mar-12	$34.97	$2.80	$37.77	$37.77	$0.00
1005	10018	Anne G. Farriss	17-Mar-12	$70.44	$5.64	$76.08	$76.08	$0.00
1009	10019	Olette K. Smith	27-Mar-12	$180.74	$14.46	$195.20	$0.00	$0.00

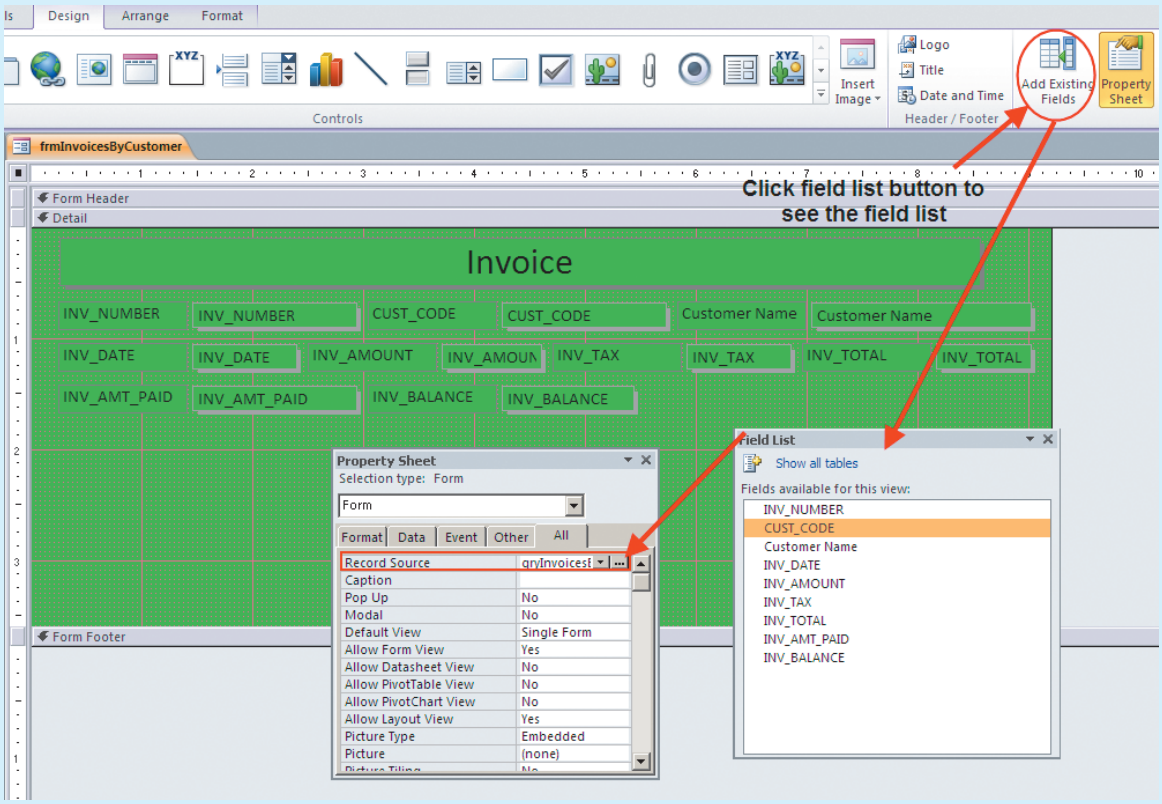
Note initial, period and space

Note that just a blank space is printed

SOURCE: Course Technology/Cengage Learning

Next, begin the form design just as you did in the previous section. However, this time you use the **qryInvoicesByCustomer** query as the data source. Use the formatting techniques you learned in the previous section to produce the form you see in Figure M.139.

FIGURE M.139 The Design view of the invoice form



SOURCE: Course Technology/Cengage Learning

Figure M.140 shows the final form view … the form limits were dragged in to reduce the presentation size and to let you see the form relative to its **Forms** list. (Note that the window has been resized to show both screen components and that the form limits have been dragged to limit its display size.)

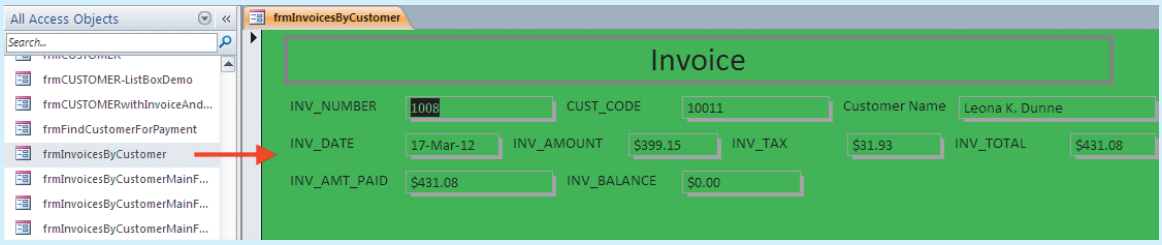
In the next section you will learn how to use one form as a subform to another, so let's go ahead and build an invoice line form next. You will also learn some additional techniques that will turn out to be very useful when you try to control input via a form.

Start by creating the form for the invoice lines by repeating the now familiar routine you used early in Section 3.1, Figure M.116. (You can use the LINE table as the data source.) Format the form to select color, font size, and other characteristics to match the form you see in Figure M.141.

3.1.3 Forms with Subforms

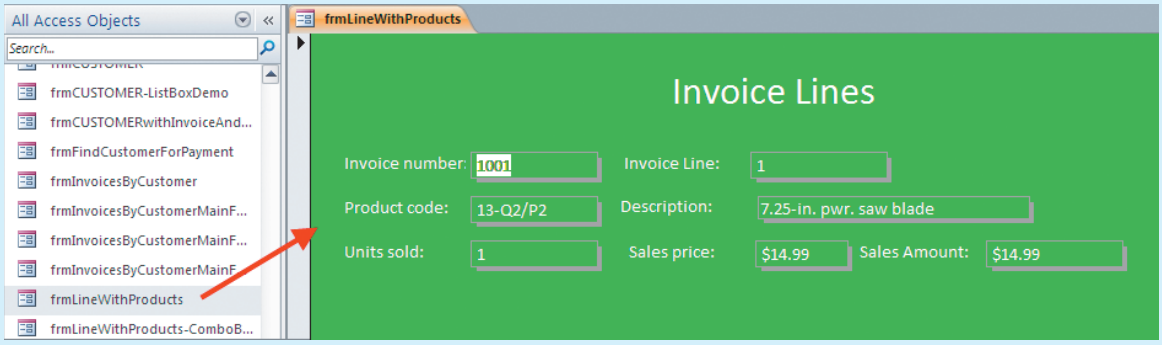
Forms contribute much to the ability to display data and/or information in a meaningful and visually appealing way. However, from an information management point of view it would be very desirable to see the interaction of related data. Fortunately, the information shown in Figures M.140 and M.141 can be combined by showing the invoice and (related) invoice lines on a single screen through a from/subform combination. In this example, the subform contains

FIGURE M.140 The Form view of the invoice form



SOURCE: Course Technology/Cengage Learning

FIGURE M.141 The line form



SOURCE: Course Technology/Cengage Learning

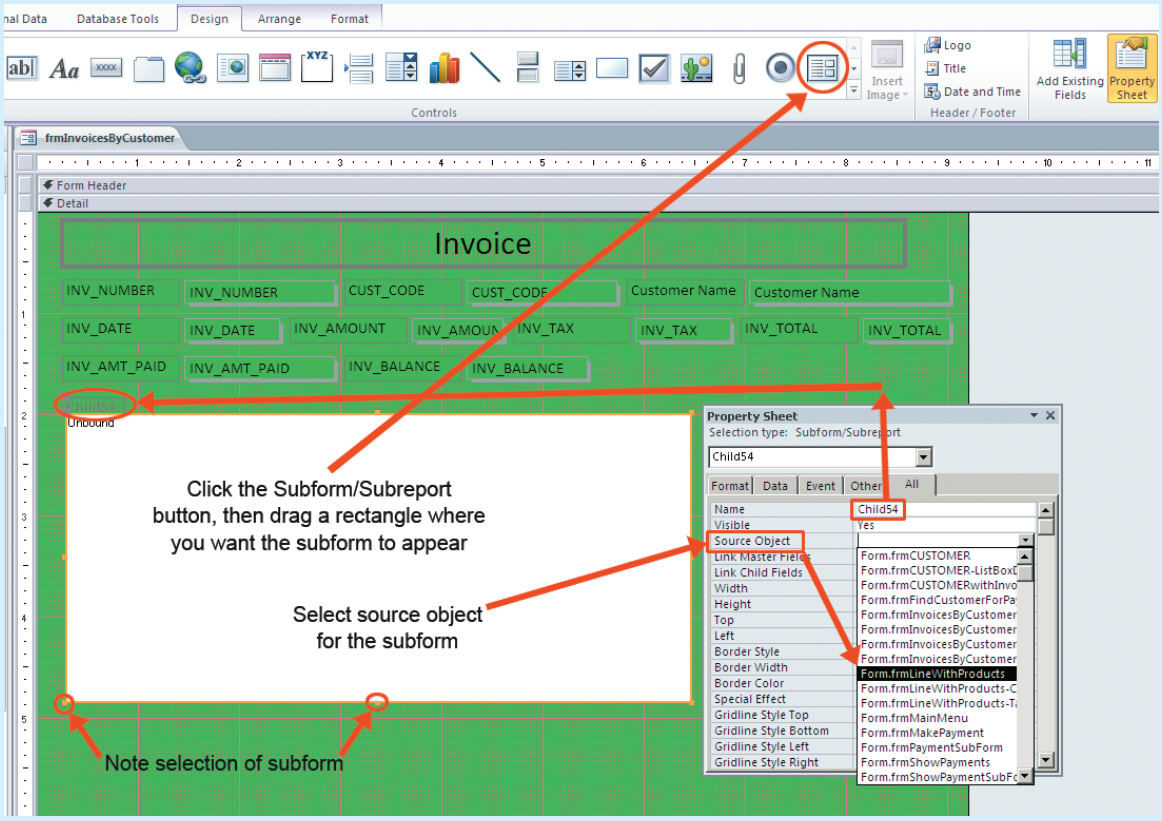
the LINE data, while the main form contains the "parent" INVOICE data. (In a 1:M relationship, the "M" side would be found in the subform and the "1" side would be seen in the main form.)

To create an (invoice) form/(line) subform presentation, begin by opening the **frmInvoicesByCustomer** form shown in Figure M.140 in its Design view, as shown in Figure M.142. (To maximize the work space, maximize the screen.) Next, click the **Form/Subform** button you see in the **Controls group**. (When you do that, the mouse pointer will change to show a small rectangle with a "+" symbol just outside its upper left-hand corner—not shown here.) Using the new pointer symbol, draw the rectangle shown in the bottom portion of the **frmInvoicesByCustomer** (main) form in Figure M.142 by dragging the mouse pointer. As soon as you release the mouse button, you will see **Subform Wizard** dialog box appear. ***Close this wizard without using it*** … you will lose the ability to control the process if you use that wizard. (That's why you don't see the **Subform Wizard** dialog box in Figure M.142.)

As soon as you have "drawn" the subform outline shown in Figure M.142, you will see its automatically generated label—in this example, it is **Child54**. (You will probably see a different "child" number designation, because the number is based on how many times you gave created a "child" object before.)

While the subform is still selected—note the selection squares in Figure M.142—open the **Properties** box for the subform. (Note that the properties box is automatically "labeled" as **Subform/Subreport: Child 20**.) Now click the **Source Object** and select the **frmLineWithProducts** form from the list as shown in Figure M.142.

FIGURE M.142 Creating a subform

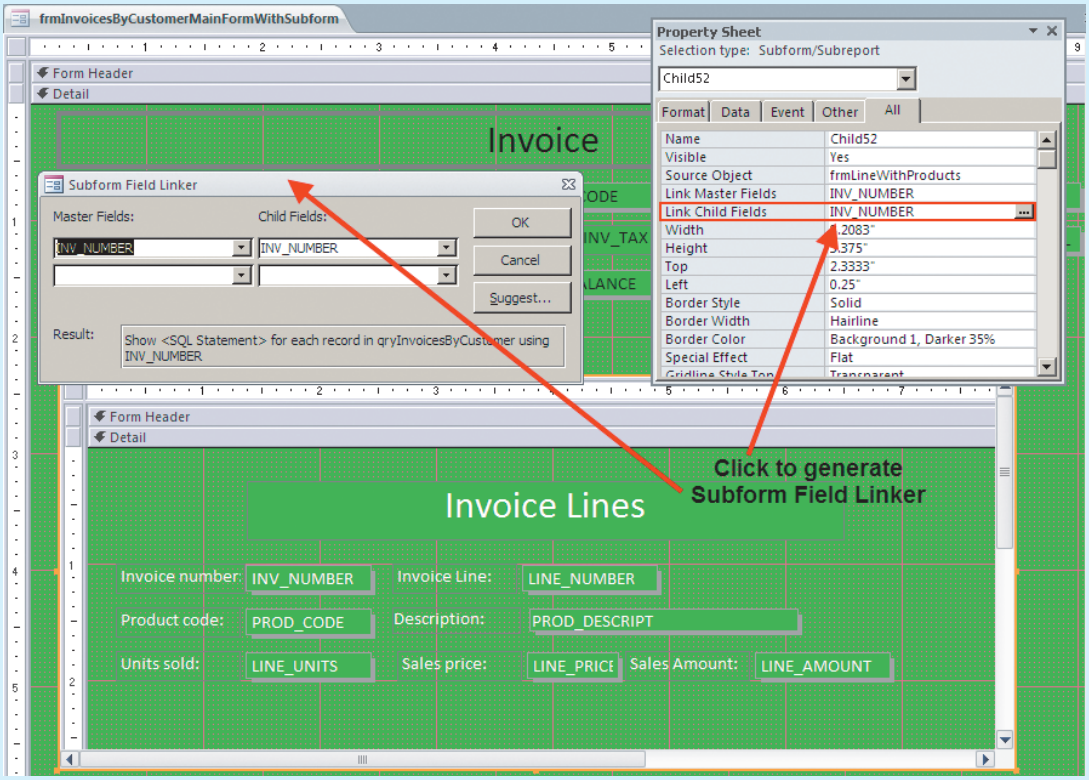


SOURCE: Course Technology/Cengage Learning

Next, click the **Link Child Fields** as shown in Figure M.143. (This action will produce the **Subform Field Linker** you see in Figure M.143.) Note that the (correct) link is made through the INV_NUMBER, because the INV_NUMBER is the foreign key (FK) in the LINE table that links the LINE table to the INVOICE table. Because this selection is correct, just click OK to accept it. You are done … that's all there is to it. You may want to drag the form and/or subform limits to line up components or to perform other formatting chores you learned earlier in this section. When you are done, open the form/subform in form view to see Figure M.144. (Note that the illustration shows the selection of invoice record 4. This invoice contains three invoice lines. To preserve the original **frmInvoicesByCustomer** form, save the new form as **frmInvoicesByCustomerMainFormWithSubform**.)

The form/subform design can be extended to include subforms that themselves include subforms. For example, you can use the just-created **frmInvoicesByCustomerMainFormWithSubform** as the subform to the **frmCUSTOMER**

FIGURE M.143 Setting the link



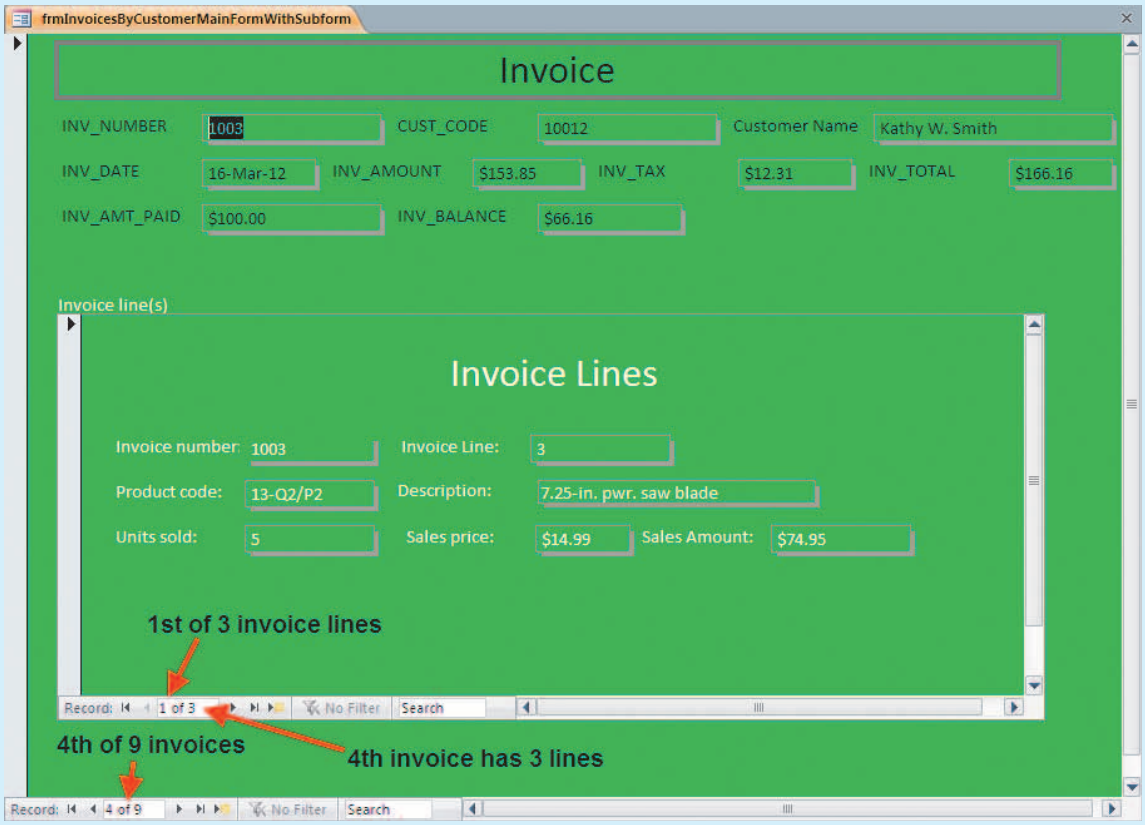
SOURCE: Course Technology/Cengage Learning

you saw in Figure M.136. (Use the techniques you just learned—see Figures M.142 to M.145—to produce the form/subform you see in Figure M.145. Save the new form/subform as **frmCUSTOMERwithInvoiceAndInvoiceLines** to preserve the original **frmCUSTOMER form**.

Thus far, you saw how forms could be used as subforms to other forms. However, you can also create subforms based on queries, without first using a form to format the query output. Start the form/subform design with the same technique that you saw in Figure M.142. (Remember to turn off the **Form/Subform Wizard** to retain direct control of the design process.) Figure M.146 shows that the **Source Object** is the **table** named **LINE**. To preserve the original **frmInvoicesByCustomer** form on which this new form/subform is based, save this second version of the invoice/line form/subform as **frmInvoicesByCustomerMainFormWithSubform-Version2**.

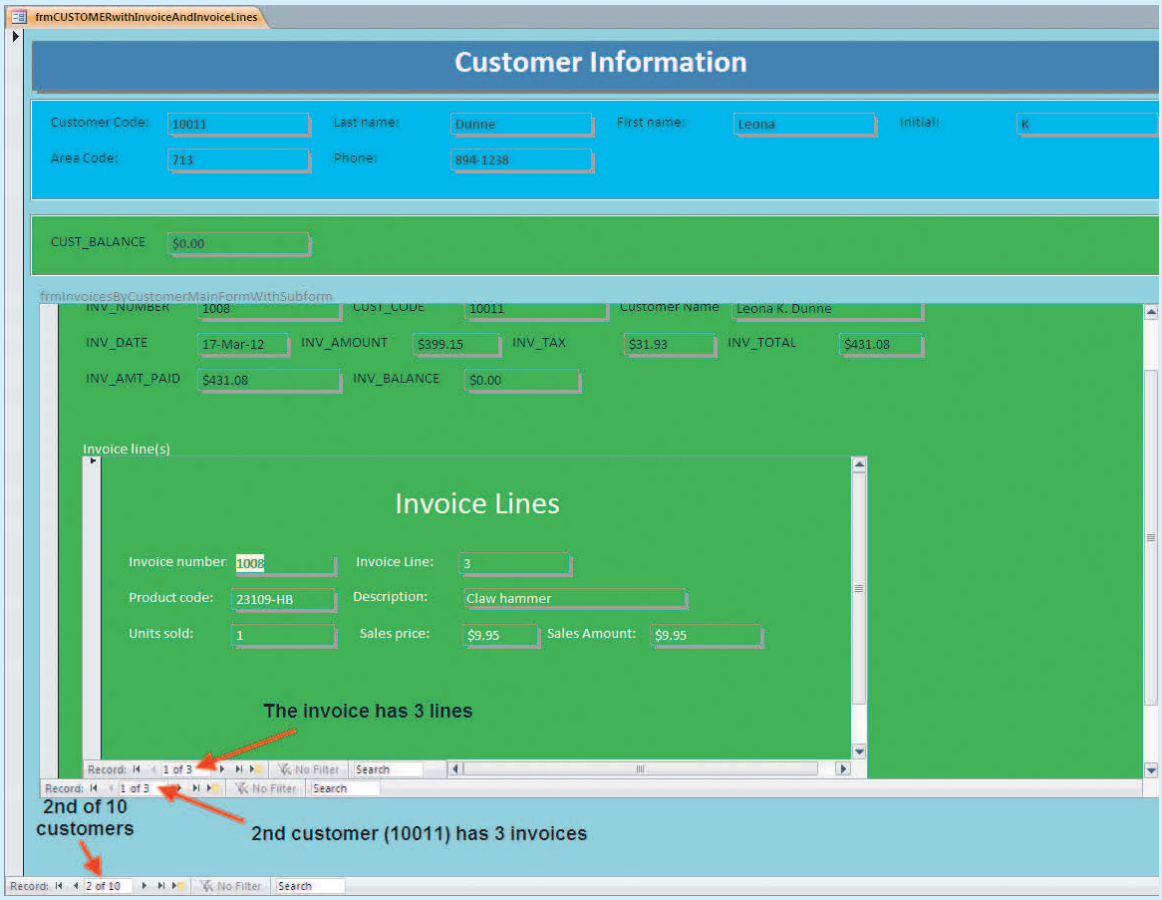
When you open this new form in **Form View**, you will see the output in Figure M.147. Whether you use this form/subform design approach or the one in Figure M.144 depends largely on end user desires and your own preferences. (Figure M.147's output shows all the invoice lines at once, while Figure M.144's invoice lines are shown one at a time as you click through the records.)

FIGURE M.144 Completed form/subform



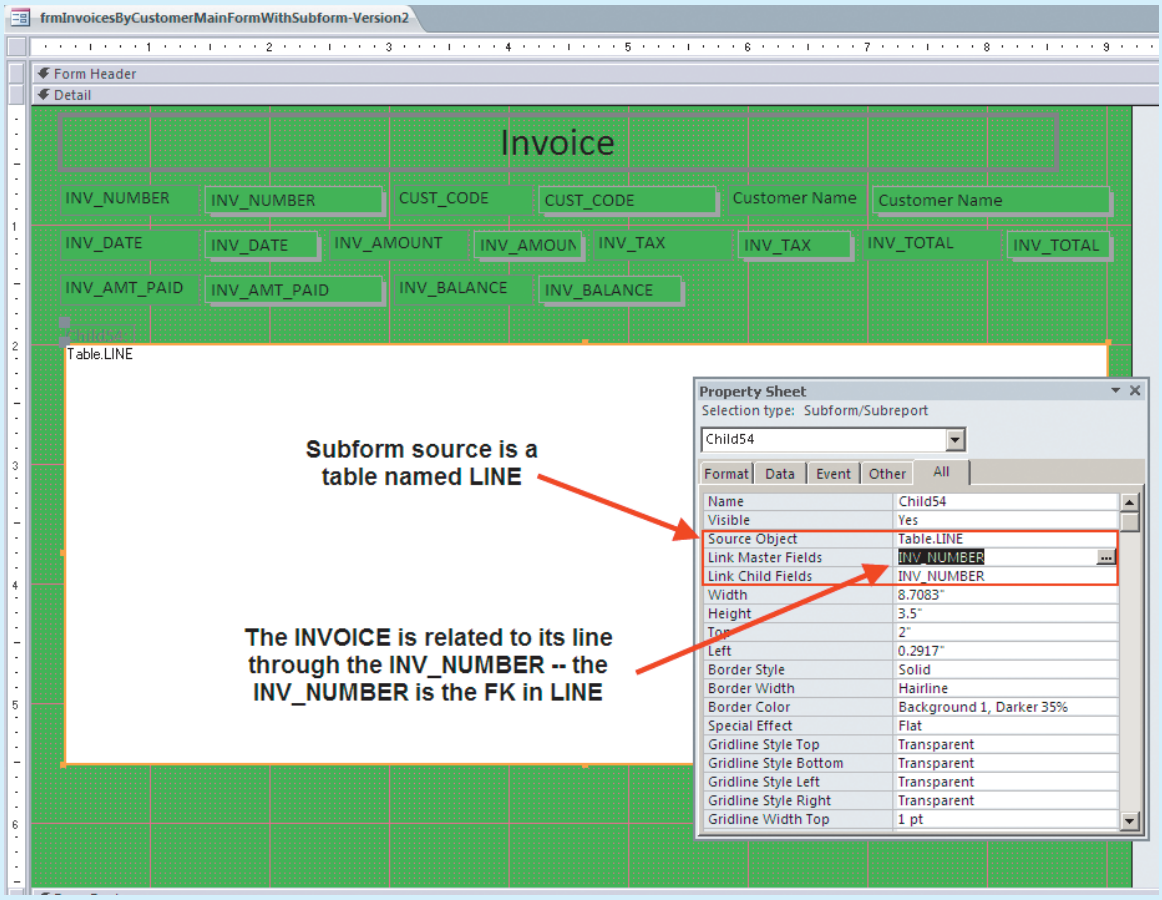
SOURCE: Course Technology/Cengage Learning

FIGURE M.145 A more complex form/subform



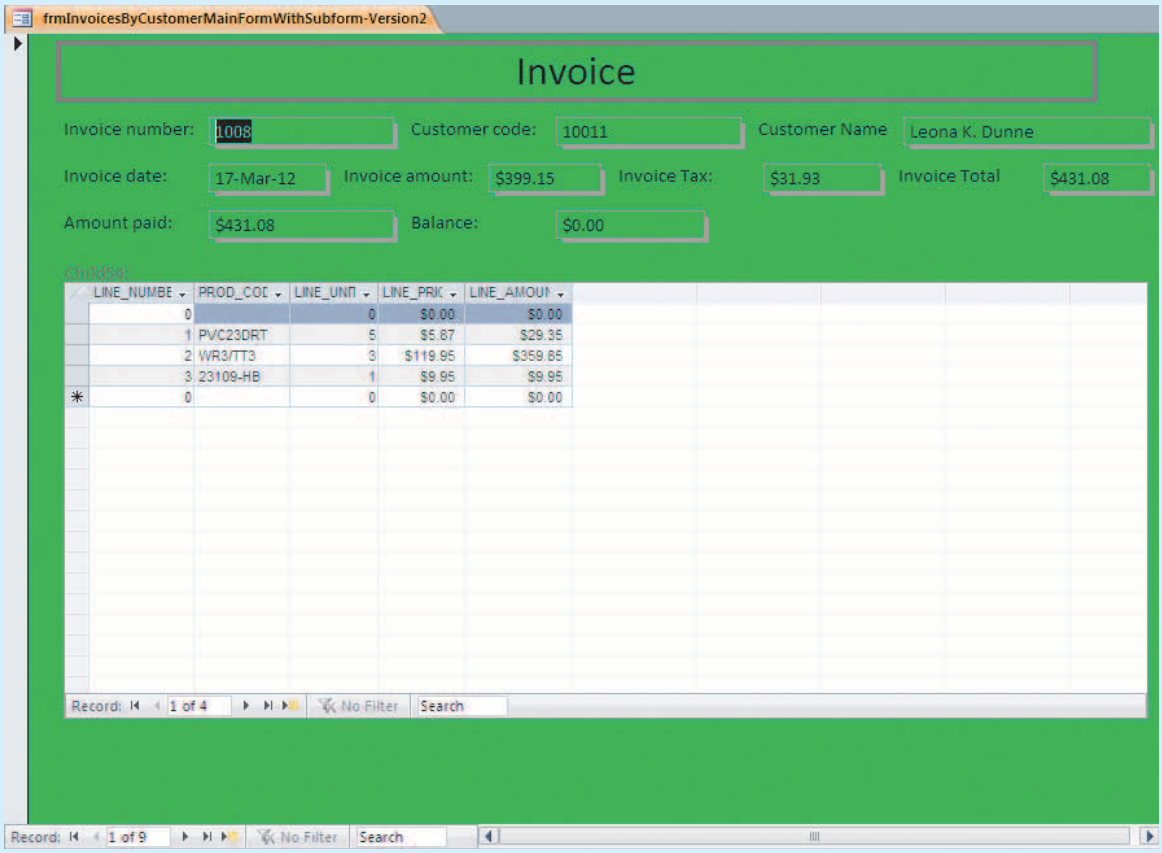
SOURCE: Course Technology/Cengage Learning

FIGURE M.146 Design view of a subform based on a table



SOURCE: Course Technology/Cengage Learning

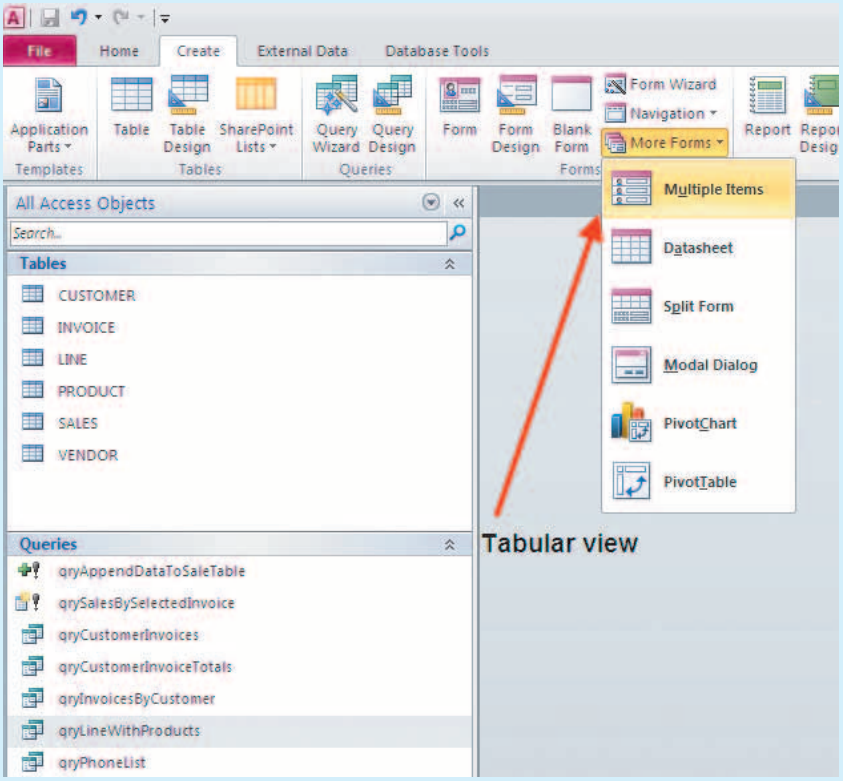
FIGURE M.147 Form view of a subform based on a table



SOURCE: Course Technology/Cengage Learning

Incidentally, you can create a form to produce an output that looks remarkably like a query output. For example, take a look at Figure M.148, which shows the start of the creation of a form based on a tabular presentation format. (Note that this form will be based on the **qryLineWithProducts** query. You will need to import this query from the **Ch07_SaleCo** database stored in the student folder using the same technique described in Section 1.3.)

FIGURE M.148 **Form based on tabular view**



SOURCE: Course Technology/Cengage Learning

When you click **Multiple Items** on the **Forms** box shown in Figure M.148, you'll see the output in Figure M.149. (The window size has been decreased by dragging its limits, using the standard Windows procedure.)

FIGURE M.149 **Form with tabular output**

qryLineWithProducts

INV_NUMBER	LINE_NUMBER	PROD_CODE	PROD_DESCRIPT	LINE_UNITS	LINE_PRICE	LINE_AMOUNT
1001	1	13-Q2/P2	7.25-in. pwr. saw blade	1	$14.99	$14.99
1001	2	23109-HB	Claw hammer	1	$9.95	$9.95
1002	1	54778-2T	Rat-tail file, 1/8-in. fine	2	$4.99	$9.98
1003	1	2238/QPD	B&D cordless drill, 1/2-in.	1	$38.95	$38.95
1003	2	1546-QQ2	Hrd. cloth, 1/4-in., 2x50	1	$39.95	$39.95
1003	3	13-Q2/P2	7.25-in. pwr. saw blade	5	$14.99	$74.95
1004	1	54778-2T	Rat-tail file, 1/8-in. fine	3	$4.99	$14.97
1004	2	23109-HB	Claw hammer	2	$9.95	$19.90
1005	1	PVC23DRT	PVC pipe, 3.5-in., 8-ft	12	$5.87	$70.44
1006	1	SM-18277	1.25-in. metal screw, 25	3	$6.99	$20.97

Record: 2 of 20 No Filter Search

SOURCE: Course Technology/Cengage Learning

Of course, you can modify the wizard-produced form, using the design techniques you learned earlier in this section. Figure M.150 shows the edited form. Save the form as **frmLineWithProducts-TabularView**.

FIGURE M.150 Edited form with tabular output

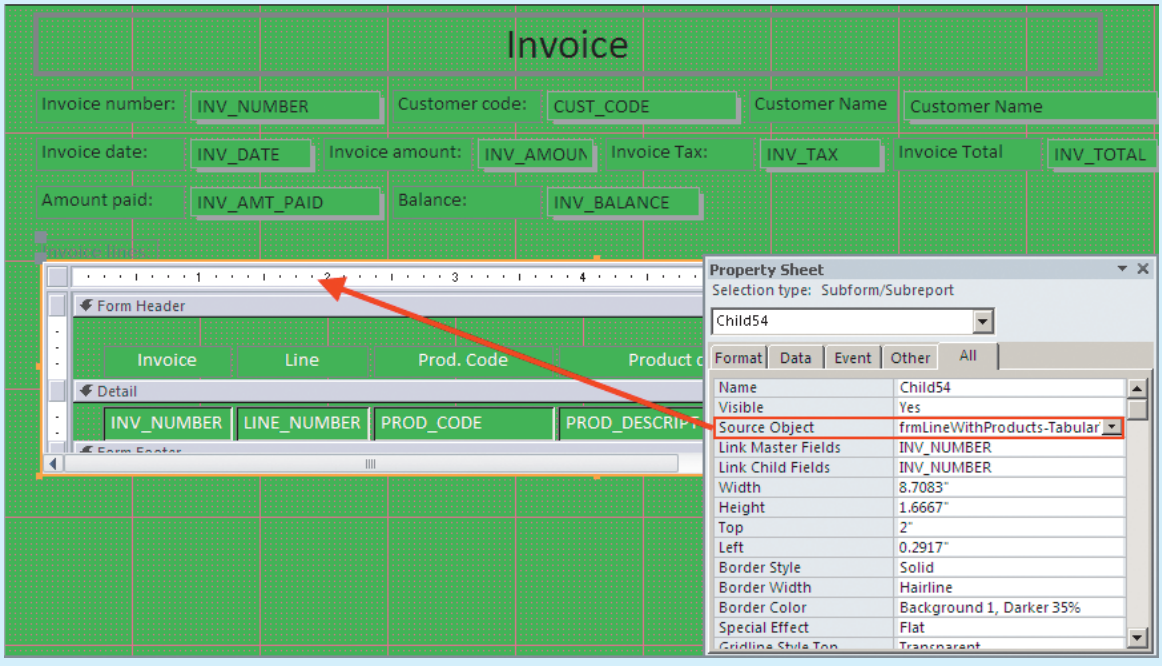
Invoice	Line	Prod. Code	Product Description	Units Sold	Line Price	Amount
1001	1	13-Q2/P2	7.25-in. pwr. saw blade	1	$14.99	$14.99
1003	3	13-Q2/P2	7.25-in. pwr. saw blade	5	$14.99	$74.95
1007	1	13-Q2/P2	7.25-in. pwr. saw blade	2	$14.99	$29.98
1003	2	1546-QQ2	Hrd. cloth, 1/4-in., 2x50	1	$39.95	$39.95
1006	2	2232/QTY	B&D jigsaw, 12-in. blade	1	$109.92	$109.92
1003	1	2238/QPD	B&D cordless drill, 1/2-in.	1	$38.95	$38.95
1001	2	23109-HB	Claw hammer	1	$9.95	$9.95
1004	2	23109-HB	Claw hammer	2	$9.95	$19.90
1006	3	23109-HB	Claw hammer	1	$9.95	$9.95
1008	3	23109-HB	Claw hammer	1	$9.95	$9.95
1002	1	54778-2T	Rat-tail file, 1/8-in. fine	2	$4.99	$9.98
1004	1	54778-2T	Rat-tail file, 1/8-in. fine	3	$4.99	$14.97
1007	2	54778-2T	Rat-tail file, 1/8-in. fine	1	$4.99	$4.99

Record: 1 of 20 No Filter Search

SOURCE: Course Technology/Cengage Learning

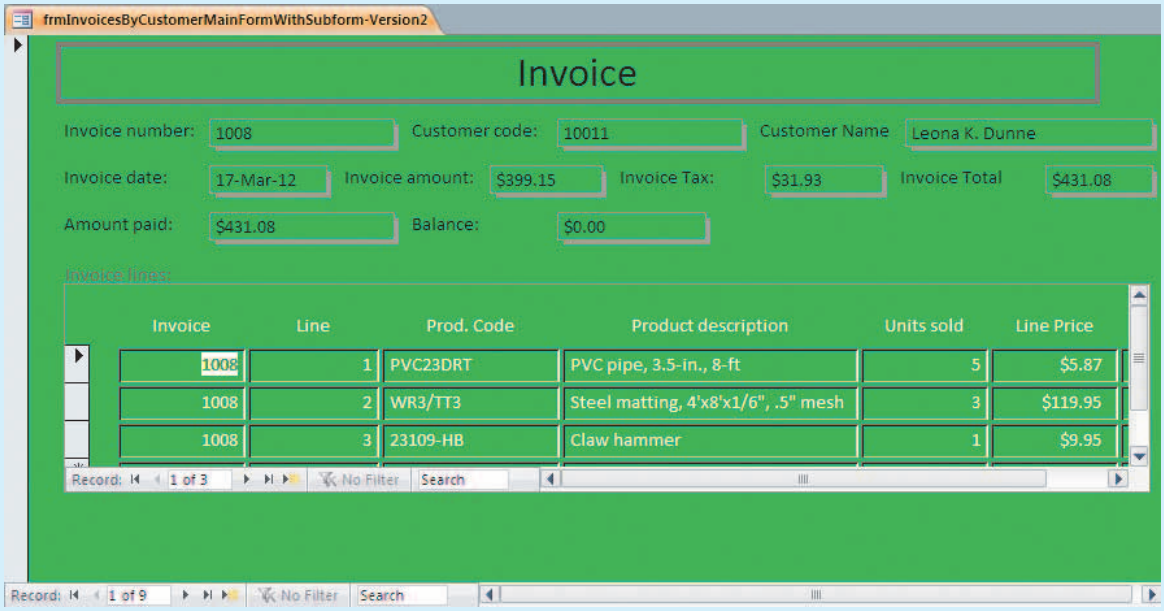
You can substitute the **frmLineWithProducts-TabularView** form in Figure M.150 as the subform in the **frmInvoicesByCustomerMainFormWithSubform-Version2** form/subform you saw in Figure M.146.

FIGURE M.151 Edited form with subform



(Note that Figure M.151 shows that you have simply used the **frmLineWithProducts-TabularView** as the new **Source Object**.)

FIGURE M.152 Edited form with subform output



SOURCE: Course Technology/Cengage Learning

Save the edited form/subform as **frmInvoicesByCustomerMainFormWithSubform-Version3** to preserve the original version 2. Open version 3 to yield the output shown in Figure M.152. (Compare this output with the one shown in Figure M.144.)

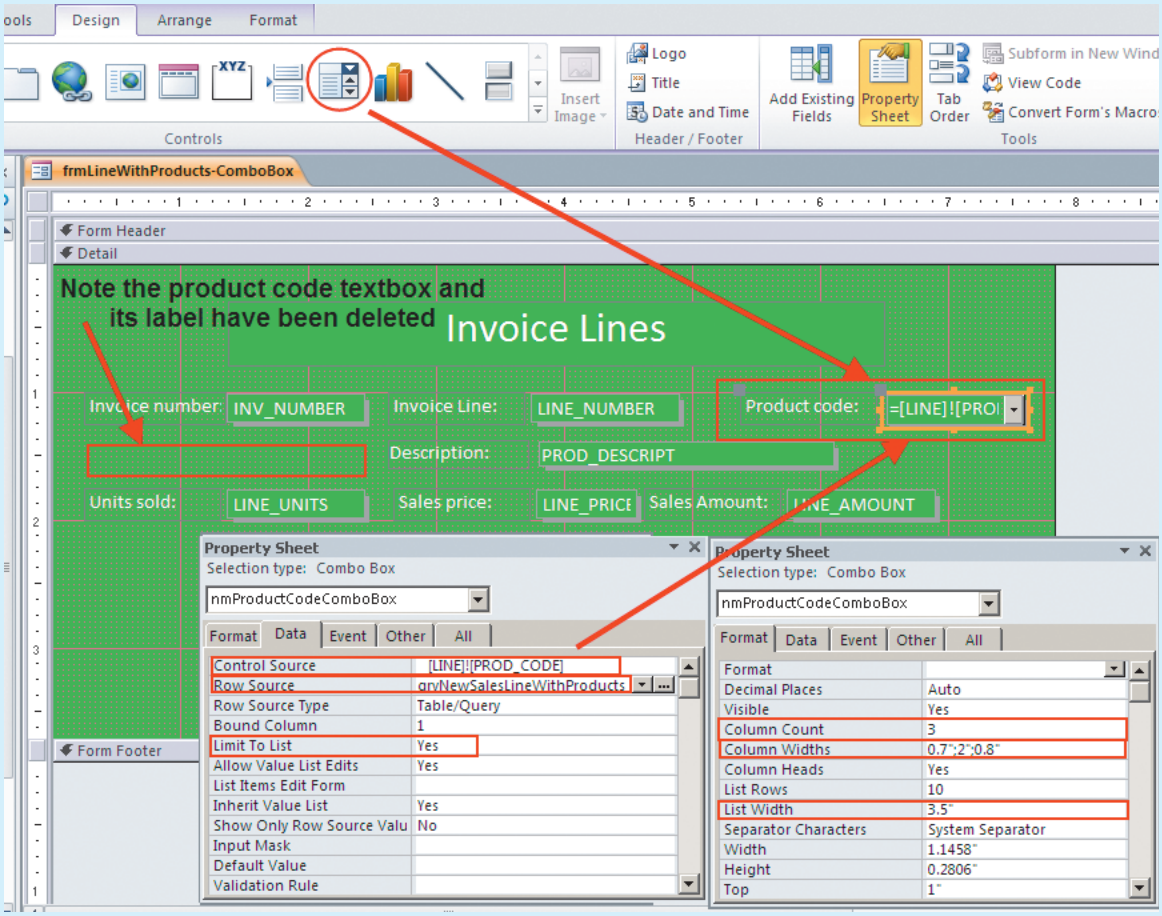
3.1.4 Controlling Input with Combo Boxes

In a real-world environment, the end user is likely to use a scanner to enter product codes and prices. However, when you are prototyping database applications, you probably don't use such "final product" options. So how do you control end user input when it involves something as convoluted as a product code? The answer is simple—just design the form that uses such inputs to include ***combo boxes***. A combo box is simply a text box that lists the available input options from which the end user may choose.

MS Access makes it easy to produce a combo box. To illustrate its design, open the **frmLineWithProducts** in Design view and save it as **frmLineWithProducts-ComboBox** to preserve the original form. Figure M.153 displays the **frmLineWithProducts-ComboBox** in Design view. Note the following procedures and features:

1. Open the **frmLineWithProducts** in **Design View** and delete the original **Product code:** field. (Check Figure M.141 to see this field. The location of the deleted field is marked in Figure M.153.)
2. Click the **Combo Box** tool in the **Controls group**.
3. Drag a space on the screen—in this example, the combo box space was created in the upper right-hand portion of the screen. In this illustration, the default name is **Combo23**. (Because this default name is selected by Access based on the previous use of combo box designs, your combo box default name is likely to be different.)
4. Right-click the combo box then generate its Properties sheet. Note that the **Combo Box** properties are shown in Figure M.153.

FIGURE M.153 Creating a combo box

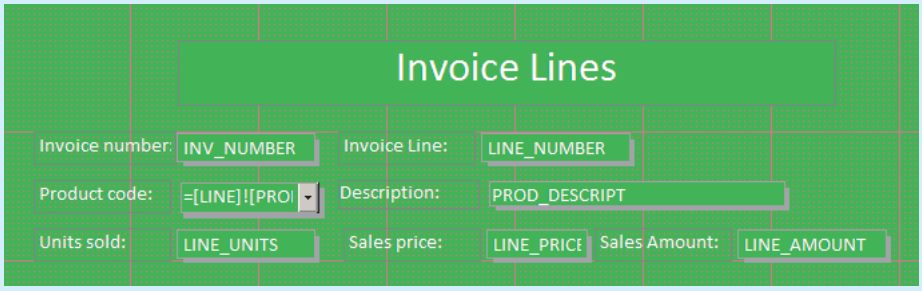


SOURCE: Course Technology/Cengage Learning

5. Select the appropriate properties—take a look at the figure. You should be familiar with the Properties box drop-down lists, so use them. For example, to select the **Control Source**, click the item and then click the down arrow—not shown here—to see the sources. In this example, the selected control source is the PROD_CODE in the LINE table, shown as **LINE.PROD_CODE**.
6. The **Row Source Type** is **Table/Query**.
7. Select the **Row Source** as shown. In other words, the row source for this combo box will be the query named **qryNewSalesLineWithProducts** (imported from the **Ch07_SaleCo** database in the student folder).
8. To let the end user see what is expected to be entered, three fields will be shown. Therefore, the **Column Count** is set at **3**.
9. Because the control source is the PROD_CODE in the LINE table, only the PROD_CODE field can be bound to the LINE table. This field value is the first one in the query, so the **Bound Column** is column 1.
10. To let the end user know the meaning of the combo box items, select **Yes** to mark the **Column Heads** option. (You can edit the column head text by using the properties for each of the three fields at the query level. If you want to edit the column heads, you can open the **qryNewSalesLineWithProducts** query in **Design View** and make the necessary changes.

11. There are three columns to be displayed. To control the **Column Widths**, enter the column width values in inches. Note the selection of **0.7";2";0.8"** in this option line. (Actually, if you type **.7,2,.8**, Access will format the entry for you.) This first entry is an approximation, so may have to toggle back and forth between the form's **Design View** and **Form View** to get the precise column width setting.
12. The **List Rows** option shows the default value **10**. You can change this value by simply typing the number of rows you would like to see displayed. Access automatically creates a scroll bar for you if the actual number of rows exceeds the specified number of rows.
13. For documentation purposes, change the combo box **Name** to **nmProductCodeComboBox**.
14. Edit the combo box label on the form to change the **Combo23** shown here to **Product Code:**
15. Drag the completed combo box to the original product code location—marked in Figure M.153—and drop it. (Figure M.154 shows the combo box in its new location.)
16. Save the newly edited **frmLineWithProducts-ComboBox** form.

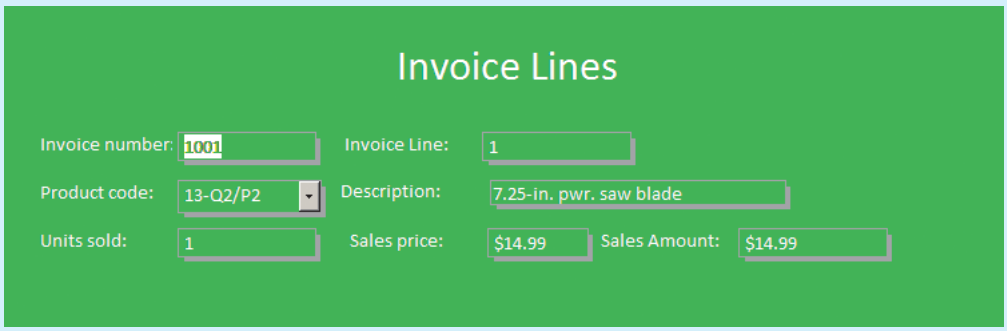
FIGURE M.154 Edited and moved combo box



SOURCE: Course Technology/Cengage Learning

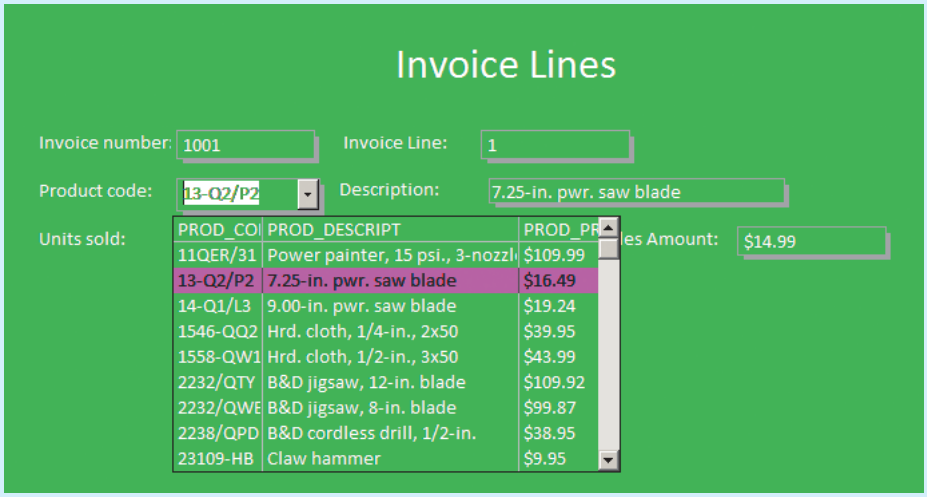
Open the **frmLineWithProducts-ComboBox** form in Form view to inspect the results shown in Figure M.155. If you now click the down arrow at the combo box margin, you will see the available entries in Figure M.156. (You can make a selection by simply clicking on a listed item. However, ***do not do so at this point, because you will change the invoice line product code, thus destroying the historical accuracy of the transaction you see here***. You can experiment with the combo box when you make a new sales transaction.)

FIGURE M.155 Combo box in Form view



SOURCE: Course Technology/Cengage Learning

FIGURE M.156 Activated combo box



SOURCE: Course Technology/Cengage Learning

3.1.5 CONTROLLING INPUT VIA LIST BOXES

You saw in the previous section that you can control input via a combo box. You can accomplish the same task through a list box. (Generally, list boxes are used to control input for items with a relatively small number of values that may or may not be generated in a query or stored in a table.) To demonstrate the creation and use of a list box, start by editing the CUSTOMER table to add a CUST_TITLE field, with a field size of 8. The results are shown in Figure M.157.

FIGURE M.157 Added CUST_TITLE field

Field Name	Data Type
CUST_CODE	Number
CUST_TITLE	Text
CUST_LNAME	Text
CUST_FNAME	Text
CUST_INITIAL	Text
CUST_ADDRESS	Text
CUST_CITY	Text
CUST_STATE	Text
CUST_ZIPCODE	Text
CUST_AREACODE	Text
CUST_PHONE	Text
CUST_BALANCE	Currency

SOURCE: Course Technology/Cengage Learning

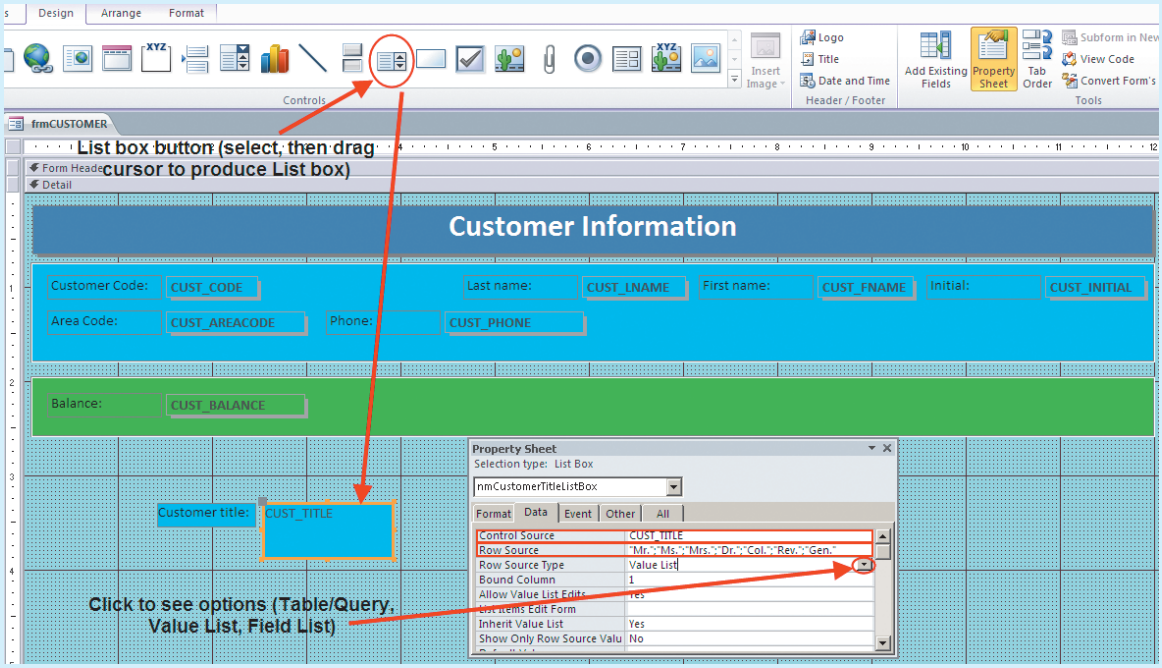
The procedures for creating a list box are similar to those used to create a combo box. One major difference is the starting point. Click the **List Box** button in the **Controls group** to get the process started, then drag a space for the list box just as you did for the combo box. Figure M.157A shows these features based on the **frmCUSTOMER** form:

1. The list box was created and that its properties show a CUST_TITLE **Control Source**.
2. The **Row Source Type** is a **Value List**. (You can click this property and then click the resulting down arrow to see the other row source type options.
3. The list of values is shown in the **Row Source**. Note that the available options are "Mr."; "Ms." … and so on. The value list option was selected because there are only a few available values. If the list of values had been long, the values might be listed in a table created for this purpose and the row source type would then be a table.
4. Note that there is only a single column of values, so the **Column Count** is **1**.
5. The (default) **Bound Column** is **1**—there is only a single column available.
6. The value list does not have a column head, so the **Column Head** selection is **No**.

7. Change the **Name** of the list box shown in Figure M.157a from **List22** to the more descriptive **nmCustomerTitleListBox**.

Save the **frmCUSTOMER** form as **frmCUSTOMER-ListBoxDemo** to preserve the original form for further experimentation.

FIGURE M.157a **Input control via list box**

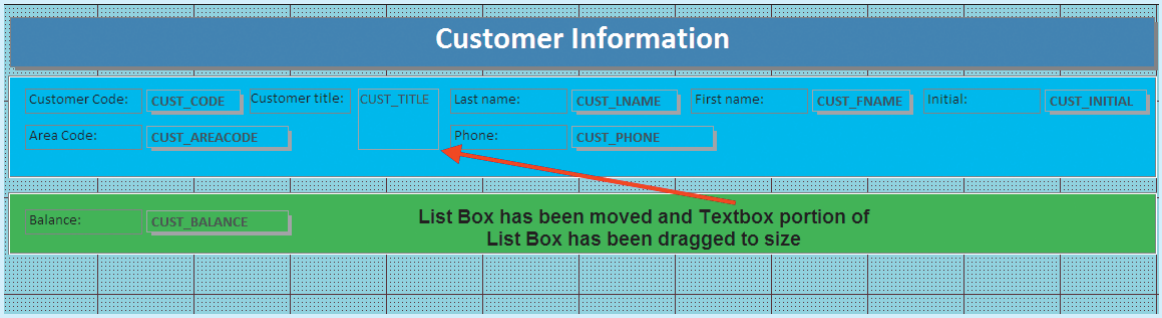


SOURCE: Course Technology/Cengage Learning

As you examine the modified form in Figure M.158, note that the original **Customer title:** text box has been deleted from the form and the new list box has been substituted.

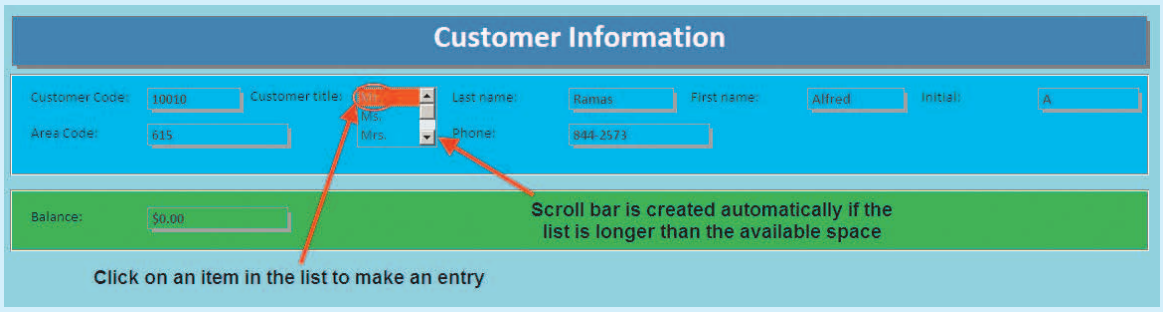
Figure M.159 shows the results of the list box modification when the form is opened in its form view.

FIGURE M.158 Edited list box



SOURCE: Course Technology/Cengage Learning

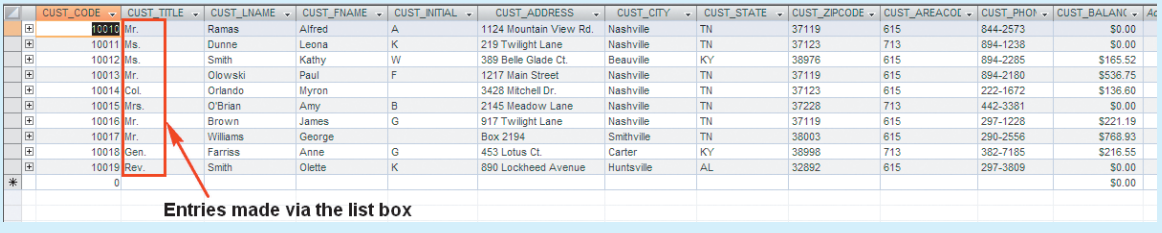
FIGURE M.159 List box in Form view



SOURCE: Course Technology/Cengage Learning

To select a customer title, simply click an item of the list and then move to the next field. Use the scroll bar to select any item that may be located below the lower limit of the text box. Figure M.160 shows that several customer titles were entered via the list box.

FIGURE M.160 CUSTOMER table entries made via list box



CUST_CODE	CUST_TITLE	CUST_LNAME	CUST_FNAME	CUST_INITIAL	CUST_ADDRESS	CUST_CITY	CUST_STATE	CUST_ZIPCODE	CUST_AREACODE	CUST_PHONE	CUST_BALANCE
10010	Mr.	Ramas	Alfred	A	1124 Mountain View Rd.	Nashville	TN	37119	615	844-2573	$0.00
10011	Ms.	Dunne	Leona	K	219 Twilight Lane	Nashville	TN	37123	713	894-1238	$0.00
10012	Ms.	Smith	Kathy	W	389 Belle Glade Ct.	Beauville	KY	38976	615	894-2285	$165.52
10013	Mr.	Olowski	Paul	F	1217 Main Street	Nashville	TN	37119	615	894-2180	$536.75
10014	Col.	Orlando	Myron		3428 Mitchell Dr.	Nashville	TN	37123	615	222-1672	$136.60
10015	Mrs.	O'Brian	Amy	B	2145 Meadow Lane	Nashville	TN	37228	713	442-3381	$0.00
10016	Mr.	Brown	James	G	917 Twilight Lane	Nashville	TN	37119	615	297-1228	$221.19
10017	Mr.	Williams	George		Box 2194	Smithville	TN	38003	615	290-2556	$768.93
10018	Gen.	Farriss	Anne	G	453 Lotus Ct.	Carter	KY	38998	713	382-7185	$216.55
10019	Rev.	Smith	Olette	K	890 Lockheed Avenue	Huntsville	AL	32892	615	297-3809	$0.00
0											$0.00

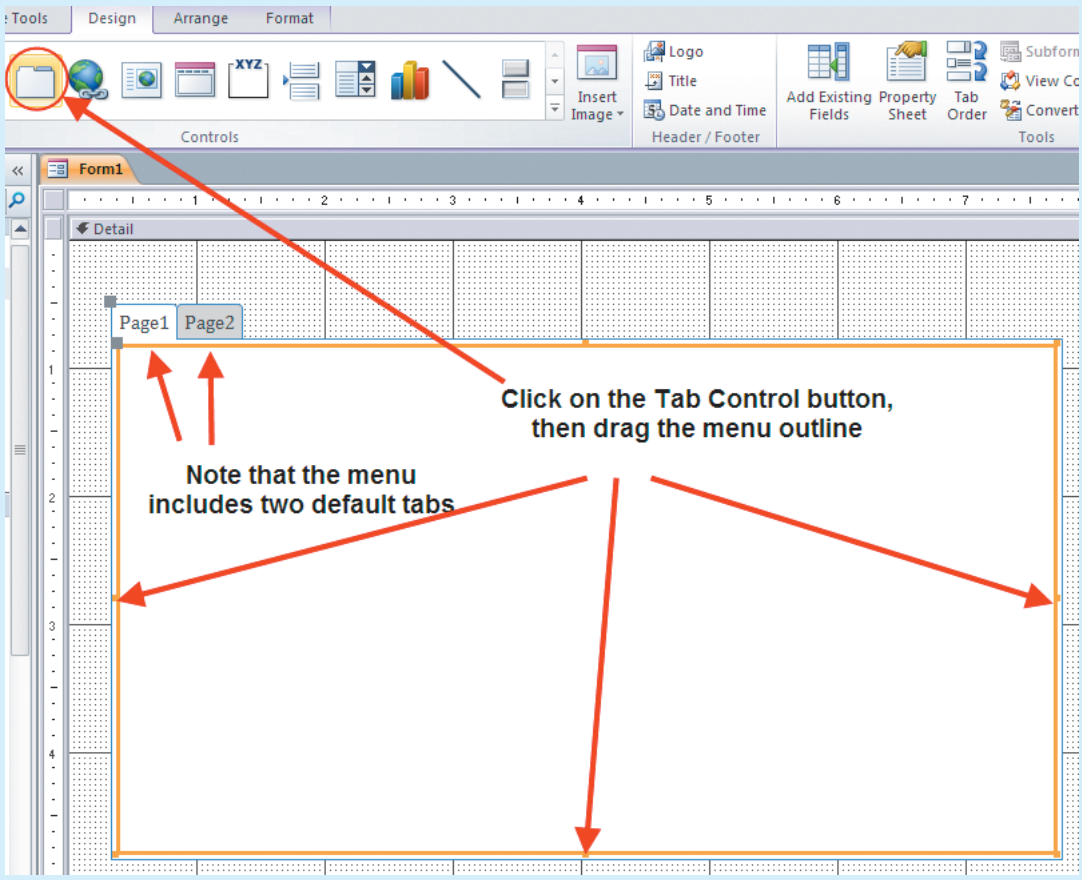
SOURCE: Course Technology/Cengage Learning

3.1.6 MENUS

If you want to make selected queries, forms, and reports easily available, a menu is a good way to get the job done. Creating menus is easy—just select **Create/Blank Form/Design View**. (***Do not select a table or query, because the menu will not be tied to any one query or table***.) When you click the **Blank Form** button, you will see a "clean" design screen on which to create the menu.

Next, select the **Tab Control** option shown in Figure M.161—note that the pointer changes shape when you do that—and draw the menu outline by dragging the pointer. Figure M.161 shows the result of dragging the menu outline. Note also that the initial use of the Tab control automatically generates two tabbed menu pages, **Page1** and **Page2**.

FIGURE M.161 Initial menu form in Design View

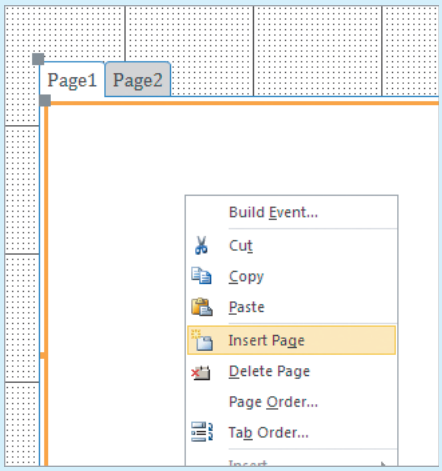


SOURCE: Course Technology/Cengage Learning

While the menu form is still selected, right-click the **Page2** tab to display the options list you see in Figure M.162. Select the **Insert Page** option. This selection will produce the **Page 3** tab you see in Figure M.163.

As you can tell by looking at Figure M.163, the tab labels can be edited through the **Property Sheet**. (The Property Sheet is generated by right-clicking the form component and selecting **Properties** from the list—that's the last item you see on the list in Figure M.162.) Go ahead and change the first tab label to **Queries** as shown in Figure M.163.

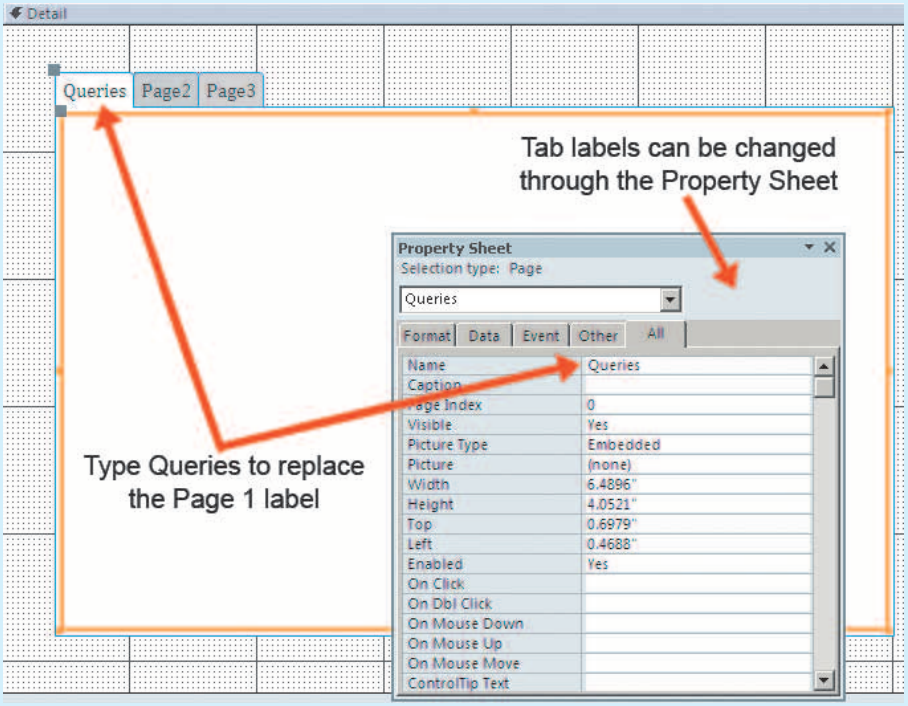
FIGURE M.162 Insert a menu page



SOURCE: Course Technology/Cengage Learning

You are now ready to place a few command buttons on the menu pages. Such buttons will be used to let the end user select a particular object—in this case, a query, a form, or a report. (You will learn about report generation in Section 4.) As the name ***command*** button suggests, clicking on such a button will generate a command that will execute the selected option. (That is, the command button will perform that duty when it has been linked to a macro or Visual Basic code. You will learn how to create and use macros in Section 5.1. This tutorial will not cover Visual Basic programming.) Figure M.164 shows how a command button is created on the first (queries) page. Make sure that the first page of the menu form has been selected—click the **Queries** tab.

FIGURE M.163 Edit the Tab Label text

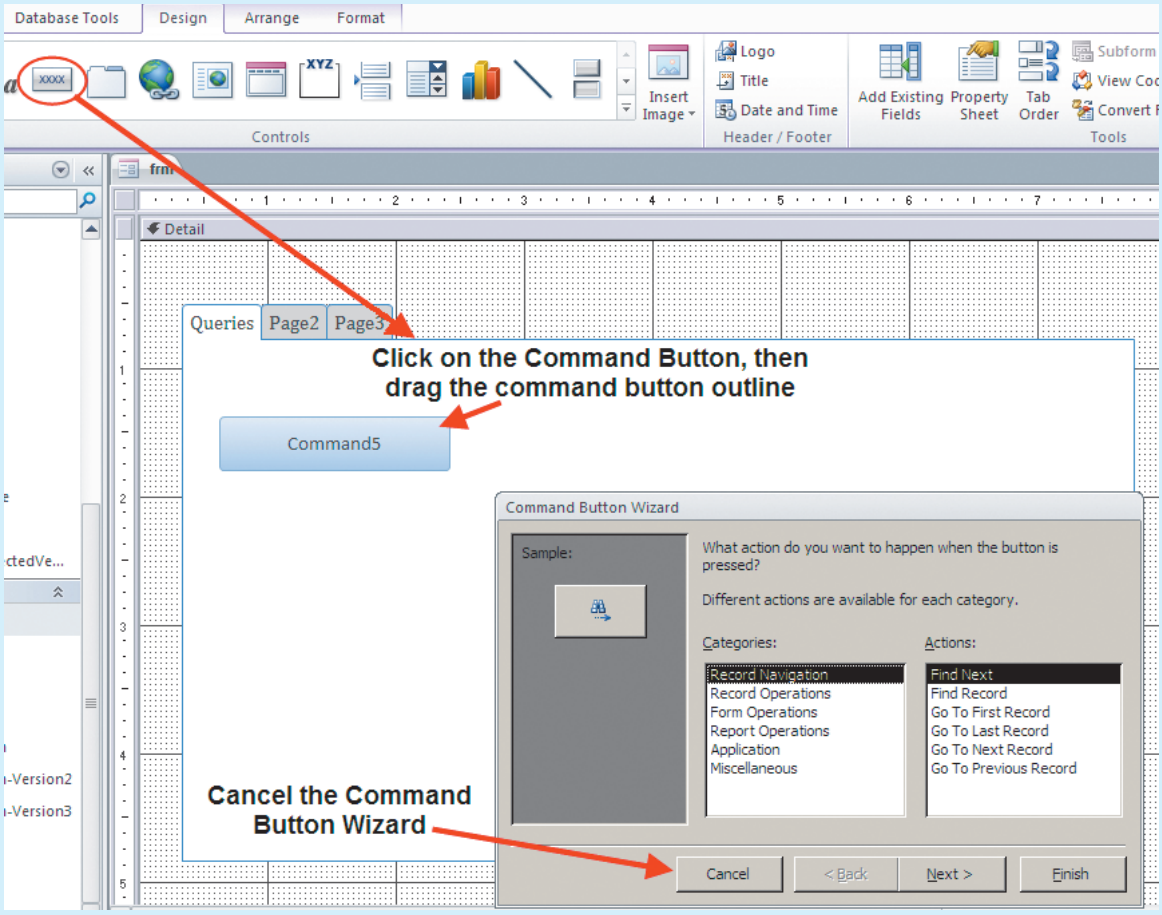


SOURCE: Course Technology/Cengage Learning

NOTE

If you do not pay attention to the page you are supposed to be on, you may wind up placing command buttons on all pages simultaneously or you may place a queries command button on a forms or reports page. Placing a command button on the wrong page—or on multiple pages at once—will have major (and unwelcome) consequences when you later connect a macro to open a query and discover that it also opens a form or some other object. Therefore, if you intend to produce one or more command buttons on the **Queries** page, make sure that you select the tab ***for that page*** when you are in design mode.

FIGURE M.164 **Creating a command button**



SOURCE: Course Technology/Cengage Learning

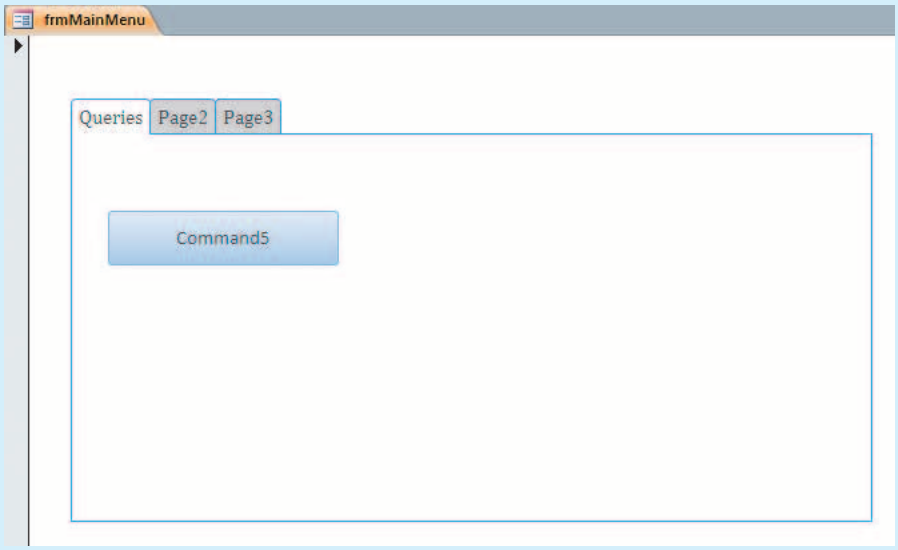
Go ahead and look at the form in Form View. You can see in Figure M.165 that the menu has a few unwanted properties such as record selectors. (The default design of a form also includes the now-familiar dividing lines—not shown here.) Therefore, "clean" the menu form by performing the actions indicated in Figure M.166. Save the menu form as **frmMainMenu**.

To edit the first command button's name and caption, stay in Design View, right-click the command button to produce the Property Sheet for that button. Change the name and caption as shown in Figure M.167.

When you are done with the editing shown in Figure M.167, continue the editing to produce the results in Figure M.168. (You should know how to boldface the command button's text, how to change the text color, how to change

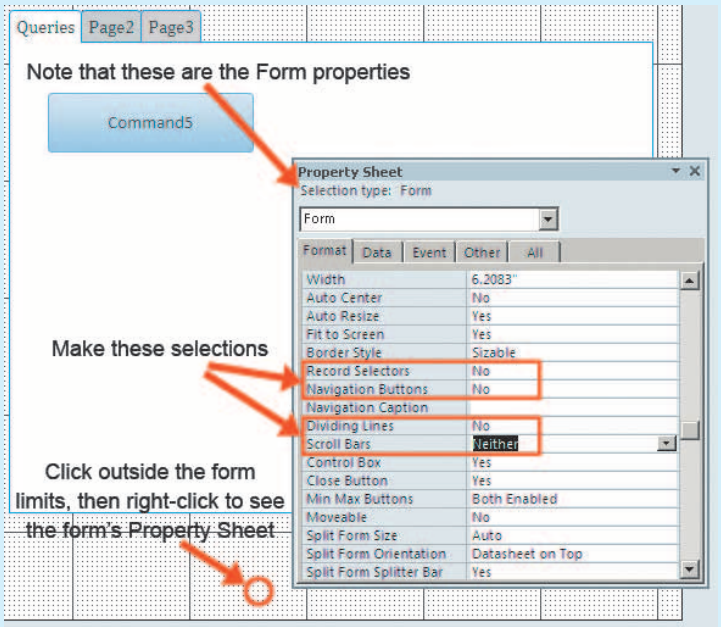
the size and location of the command button by dragging its limits, and how to edit the tabs.) Note that this command button occurs on the Queries page only—the remaining two pages are still blank. (Go ahead and click from tab to tab to see the results.)

FIGURE M.165 The initial menu in Form view



SOURCE: Course Technology/Cengage Learning

FIGURE M.166 Clean up the menu

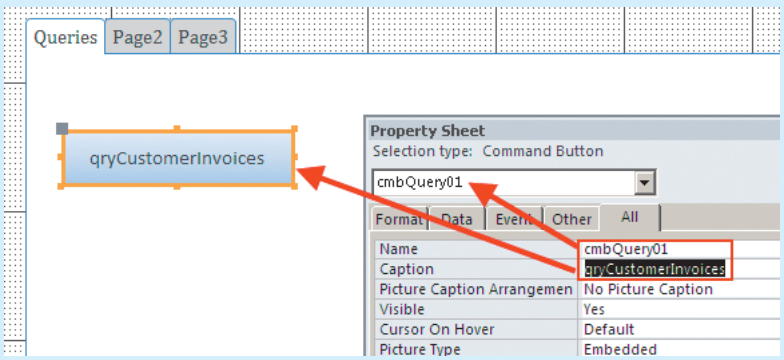


SOURCE: Course Technology/Cengage Learning

Let's create few more buttons for the **Queries** page. The easiest way to do this is to use a procedure you should know if you are familiar with Windows. That is, select the object in Design View and then use the Edit/Copy/Edit/Paste routine to make and place copies. (Remember to make sure that you are on the Queries page!) Drag the buttons

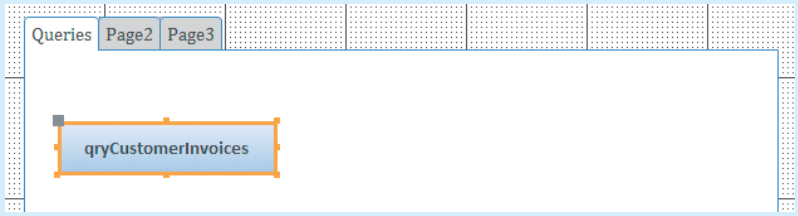
to their intended positions. Note that the editing routine ensures that all the buttons have the same properties. When you have made and placed the button copies, edit each to match the results in Figure M.169. Don't forget to save your efforts from time to time. (Save the form as **frmMainMenu**.)

FIGURE M.167 First edited query command button



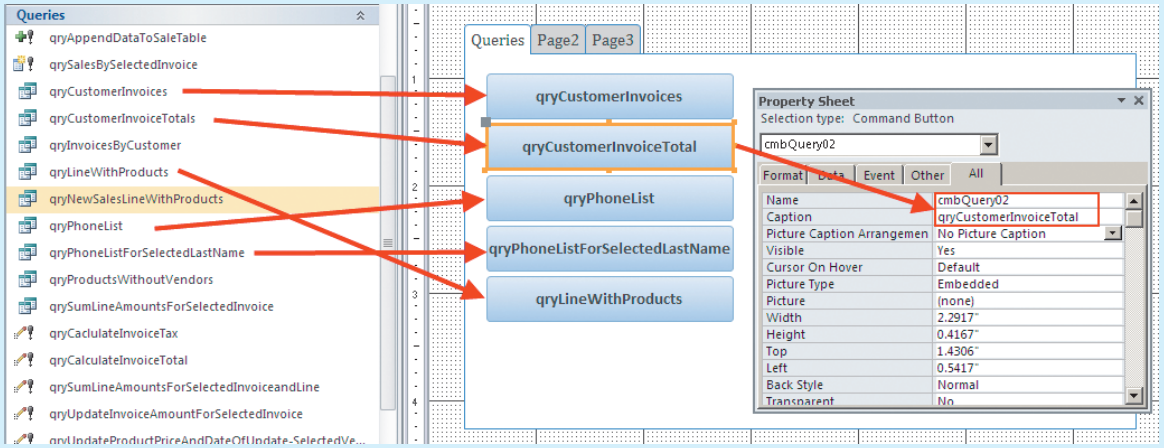
SOURCE: Course Technology/Cengage Learning

FIGURE M.168 Additional basic edits



SOURCE: Course Technology/Cengage Learning

FIGURE M.169 Multiple Queries page command buttons



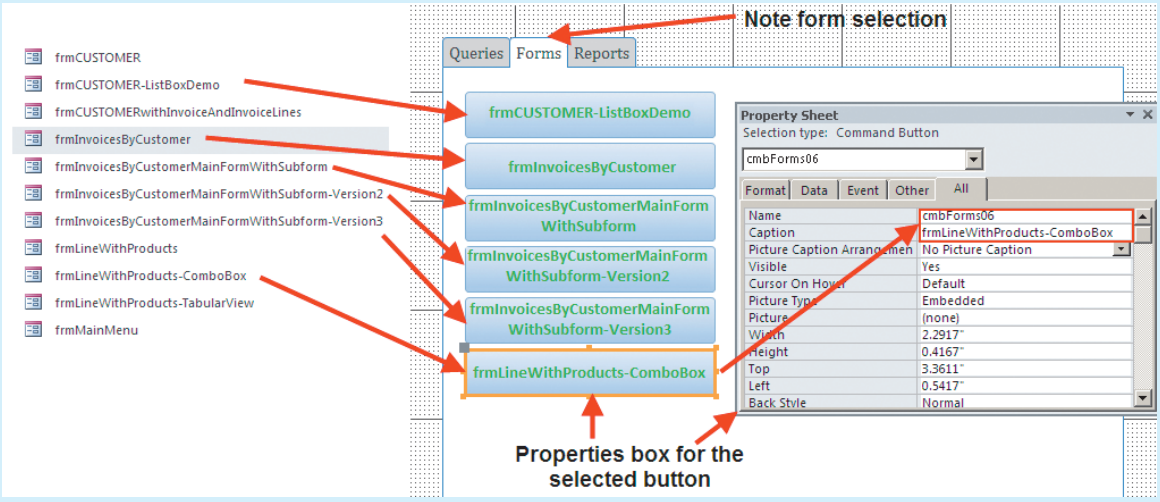
SOURCE: Course Technology/Cengage Learning

Next, copy the buttons you see here and paste them into the **Forms** page, then edit the buttons to match the forms that will be included. When you are done, your page—in Design View—should match Figure M.170. Note that the letter color has been changed to dark green and that this page has one more button than the query page.

You have not yet created any reports, so leave the **Reports** page blank. (You will learn how to create reports in Section 3.4.)

Next, let's create a **Close Menu** command button that shows up on each page. Figure M.171 shows that a single button has been copied and pasted ***outside*** the pages.

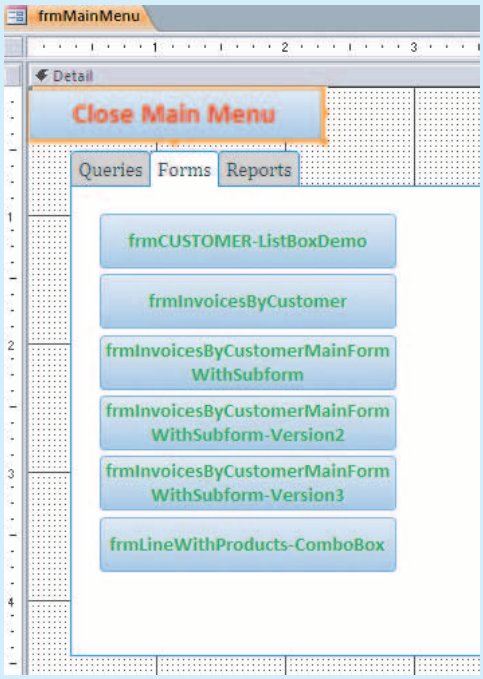
FIGURE M.170 Form page command buttons



SOURCE: Course Technology/Cengage Learning

When you drag the command button from its "outside the page" location and drop it on the first page, ***it will show up on each page***. When this button is activated via a macro you will develop in Section 3.5, you will be able to close the menu from any page. (Go ahead and do the just-described drag-and-drop routine and then click through each page and note that the **Close Main Menu** button is located on each page.)

FIGURE M.171 Close Menu button initial location



SOURCE: Course Technology/Cengage Learning

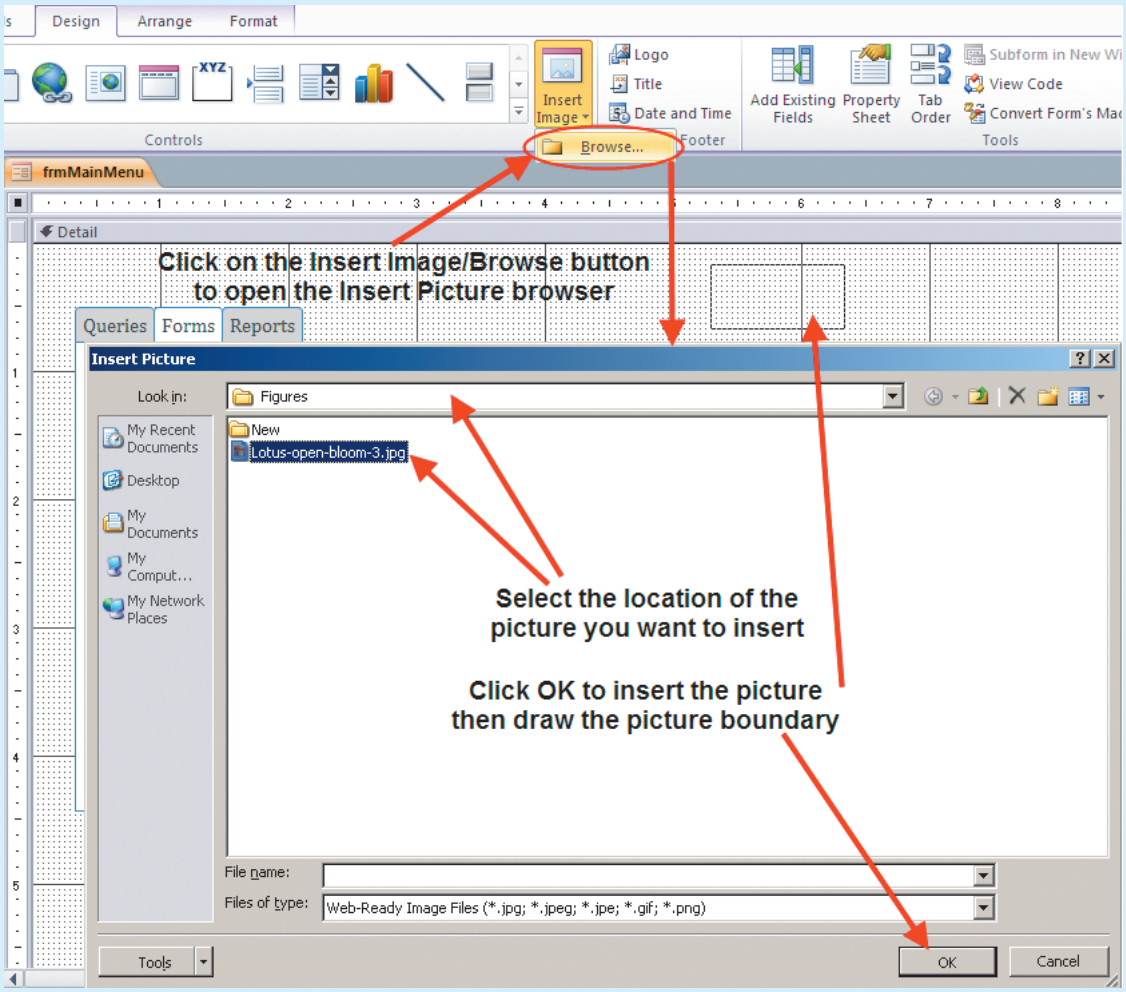
Finally, let's dress up the menu by using a picture. Any picture will do, but there happens to be a picture of a Lotus flower in the same folder as the database, so let's see how that picture may be inserted into the menu form. Figure M.172 shows the procedure. Make sure that the picture frame is initially drawn outside the page limits and then drag-and-drop it on the first page to make sure that the picture shows up on all pages.

When you click **OK** as shown in Figure M.172, the picture will be inserted. However, all you see is a small part of it. That's because the default setting is **Clip**. Use the picture's Properties box to reset the presentation to **Zoom** as shown in Figure M.173.

Note that the **Image** object in Figure M.173 was created ***outside*** the menu form's detail section. ***Therefore, if you drag this new object to any page, it will show up on all pages***. Go ahead and perform the drag and drop routine, then drag the image object's boundaries to enlarge the image as shown in Figure M.174.

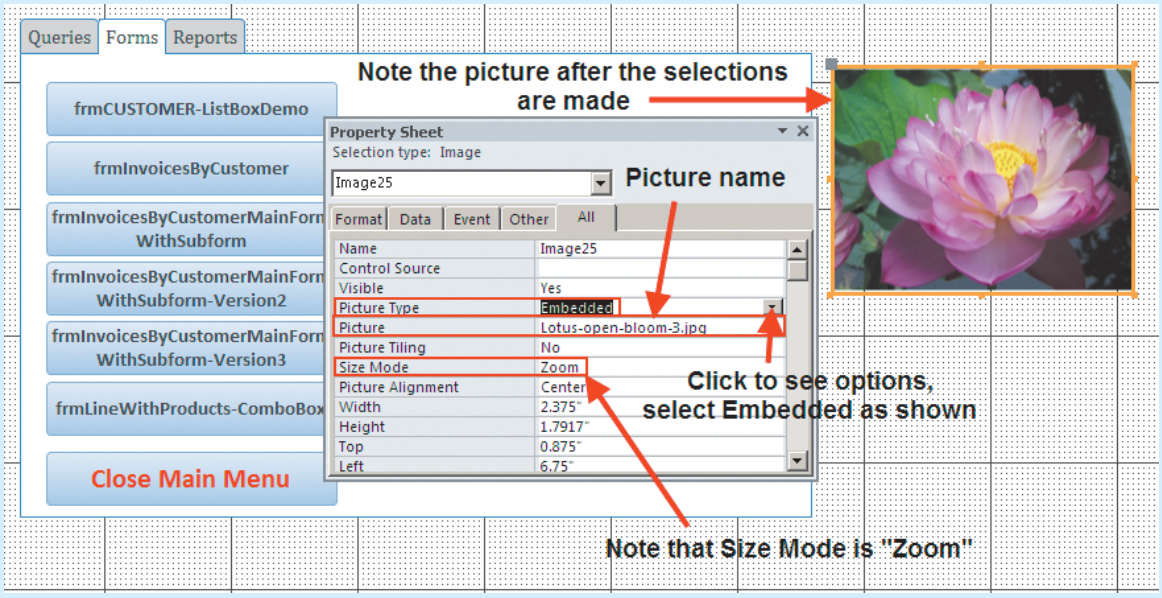
You now have an attractive menu that will become the hub for managing the Access applications. (You can move from menu page to menu page by clicking the tabs.) Save the form again when you are satisfied with the form's layout. (The form was saved earlier as **frmMainMenu**.)

FIGURE M.172 Insert a picture



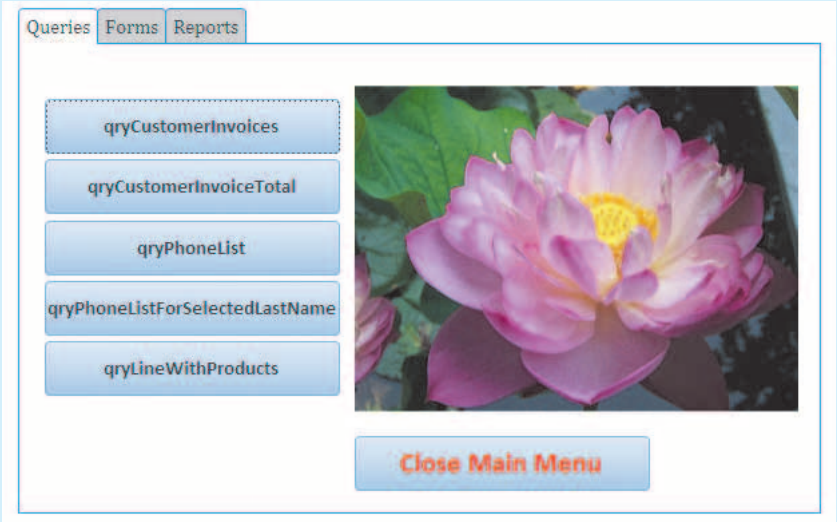
SOURCE: Course Technology/Cengage Learning

FIGURE M.173 Formatted picture



SOURCE: Course Technology/Cengage Learning

FIGURE M.174 Completed menu in Form view

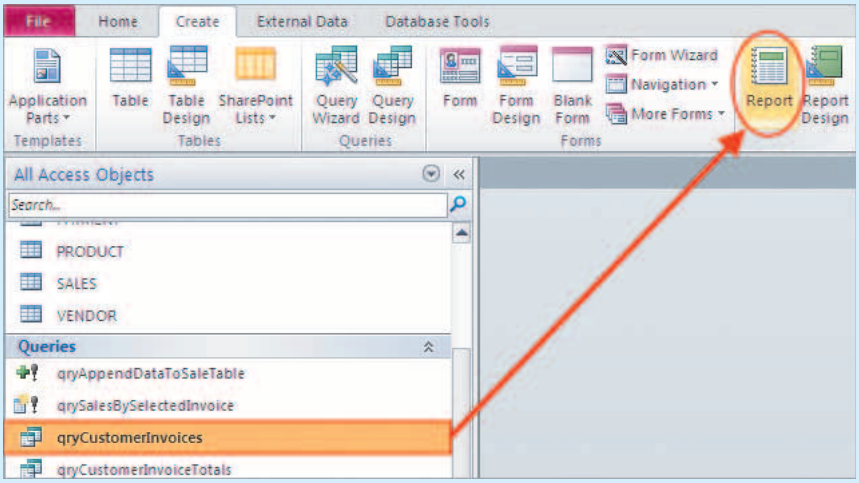


SOURCE: Course Technology/Cengage Learning

4.1 Reports

Although reports often contain the same information as forms, they do have several advantages. First, it is much easier to show multiple-record information in reports than in forms. Second, given their layout, reports are much easier to print than forms. In addition, if you have a lot of numeric data to present, the Access report format enables you to produce subtotals, totals, and grand totals as the report is generated.

FIGURE M.175 Designing a new report



SOURCE: Course Technology/Cengage Learning

FIGURE M.176 New report output

Customer

Monday, September 12, 2011
11:41:55 AM

CUST_CODE	CUST_LNAME	CUST_FNAME	CUST_INITIAL	INV_NUMBER	INV_DATE	INV_TOTAL
10014	Orlando	Myron		1001	16-Mar-12	$26.94
10011	Dunne	Leona	K	1002	16-Mar-12	$10.78
10012	Smith	Kathy	W	1003	16-Mar-12	$166.16
10011	Dunne	Leona	K	1004	17-Mar-12	$37.66
10018	Farriss	Anne	G	1005	17-Mar-12	$76.08
10014	Orlando	Myron		1006	17-Mar-12	$429.66
10015	O'Brian	Amy	B	1007	17-Mar-12	$37.77
10011	Dunne	Leona	K	1008	17-Mar-12	$431.08
10019	Smith	Olette	K	1009	27-Mar-12	$195.20
						$1,411.33

Page 1 of 1

SOURCE: Course Technology/Cengage Learning

The Report wizard is excellent and it requires little effort on your part to produce a usable report. The report generating sequence—**Create/Report**—is shown in Figure M.175. Note that the data source is a query named **qryCustomerInvoices**.

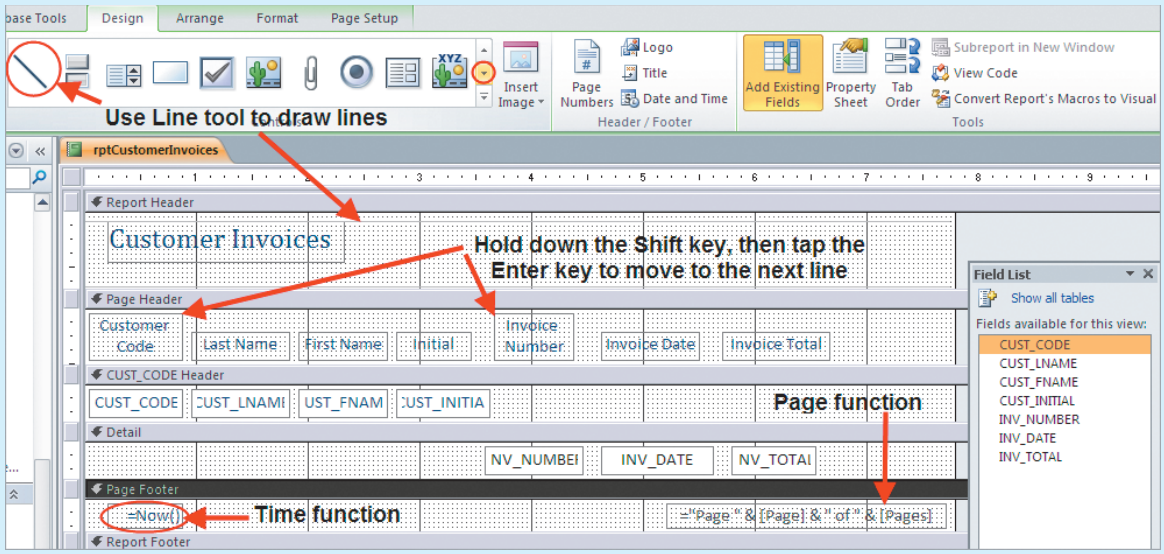
As soon as you click the **Report** button shown in Figure M.175, Access will generate the report and show its output. Only a few of the report records are shown. (See Figure M.176.)

Although the report output shown in Figure M.176 is already quite usable, you should use the now-familiar design tools and practices to add functionality and visual appeal. (After all, end users are likely to "grade" your professional expertise on the basis of the presentation quality.) However, before you start editing the report, save the report as **rptCustomerInvoices**. Note that naming the report to match its source—in this case, a query named **qryCustomerInvoices**—maintains the self-documentation standard. (Note the **rpt** prefix denotes a **rep**ort.)

Let's begin the editing by improving the headers. Figure M.177 shows that the font was changed and moved to improve the presentation format. Note also that Access has placed some handy functions at the bottom of the report … you can move those to the top of the page later and then add the date function, **date()**. Because all the necessary fields have been included, go ahead and close the field list for the **qryCustomerInvoices**.

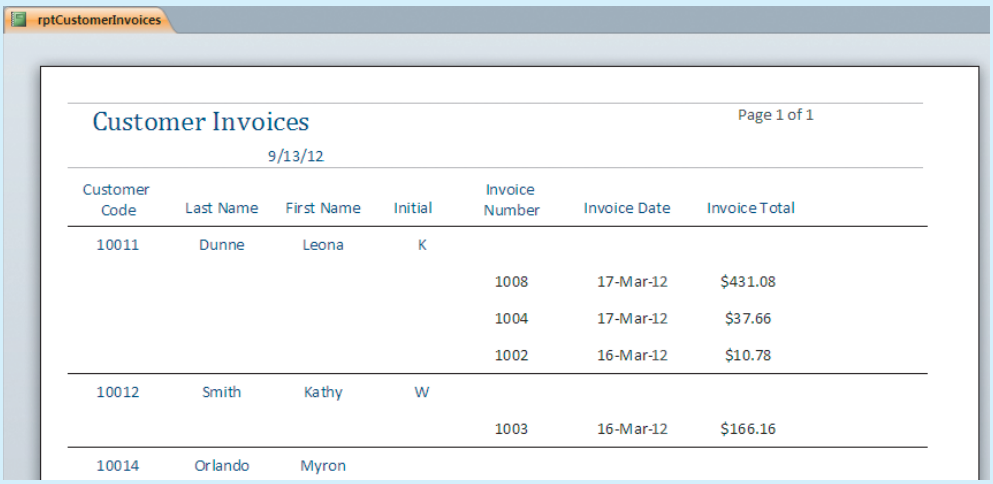
Save the changes you have just made and then open the report in Print Preview to generate the results shown in Figure M.178. (Only a few records are shown. You may have to go back and forth between the design and print preview to help you line up the various text boxes.)

FIGURE M.177 **Edited report format**



SOURCE: Course Technology/Cengage Learning

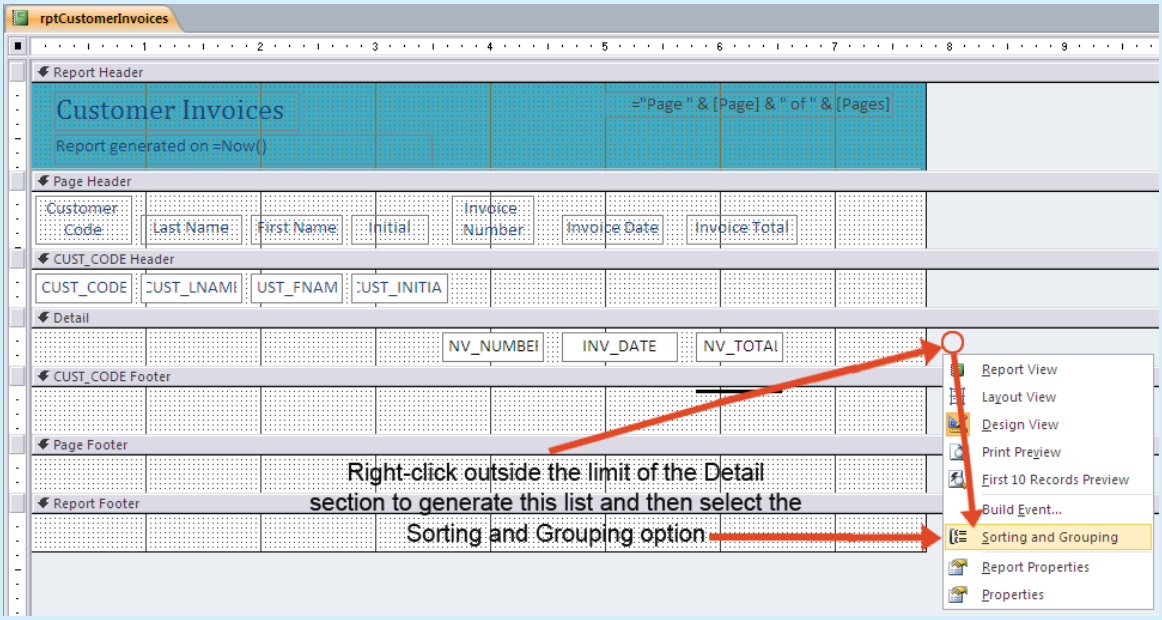
FIGURE M.178 **Print preview of edited report**



SOURCE: Course Technology/Cengage Learning

Let's do some additional report editing. First, move the time and page functions to the top of the page and add the =NOW( ) date function as shown in Figure M.179. Next, let's generate the invoice total ***for each customer***; that is, you need a customer subtotal. Therefore, the invoices must be grouped by the CUST_CODE field and you must have a place—known as a ***detail footer***—to display the output. To create a detail footer, right-click just outside the Detail section to generate the options list for the Detail section. The edits are shown in Figure M.179 and the edit results are shown in Figure M.180.

FIGURE M.179 **Additional edits**

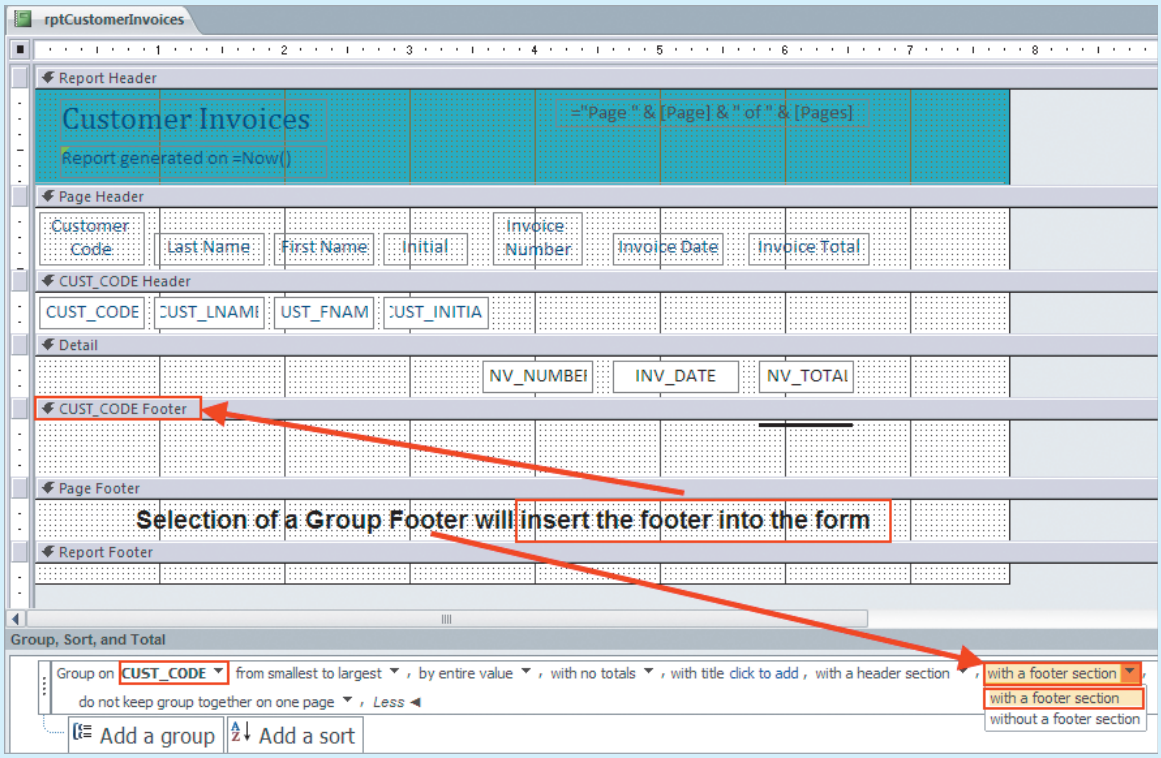


SOURCE: Course Technology/Cengage Learning

Now that you have added the section footer, you can insert subtotals for the section and then insert totals for all the customer invoices in the report footer. Note the text box formats in Figure M.181 and then look at the revised report's output. Make sure that you save the report again to preserve all the changes you have made. Figures 181 and 182 show the various formatting changes and the final output. Save the report from time to time, using the report name **rptCustomerInvoices**.

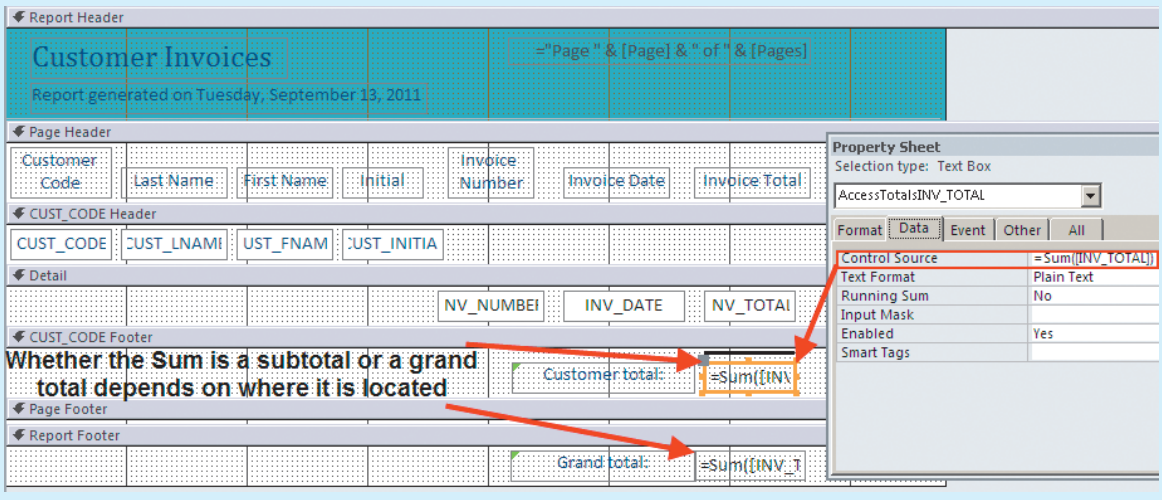
As you examine Figure M.181, note the creation of the (computed) customer total text box and especially note its computation through the **Sum** function. *The same function is used to compute the grand total*. Aside from the fact

FIGURE M.180 Addition of section footer



SOURCE: Course Technology/Cengage Learning

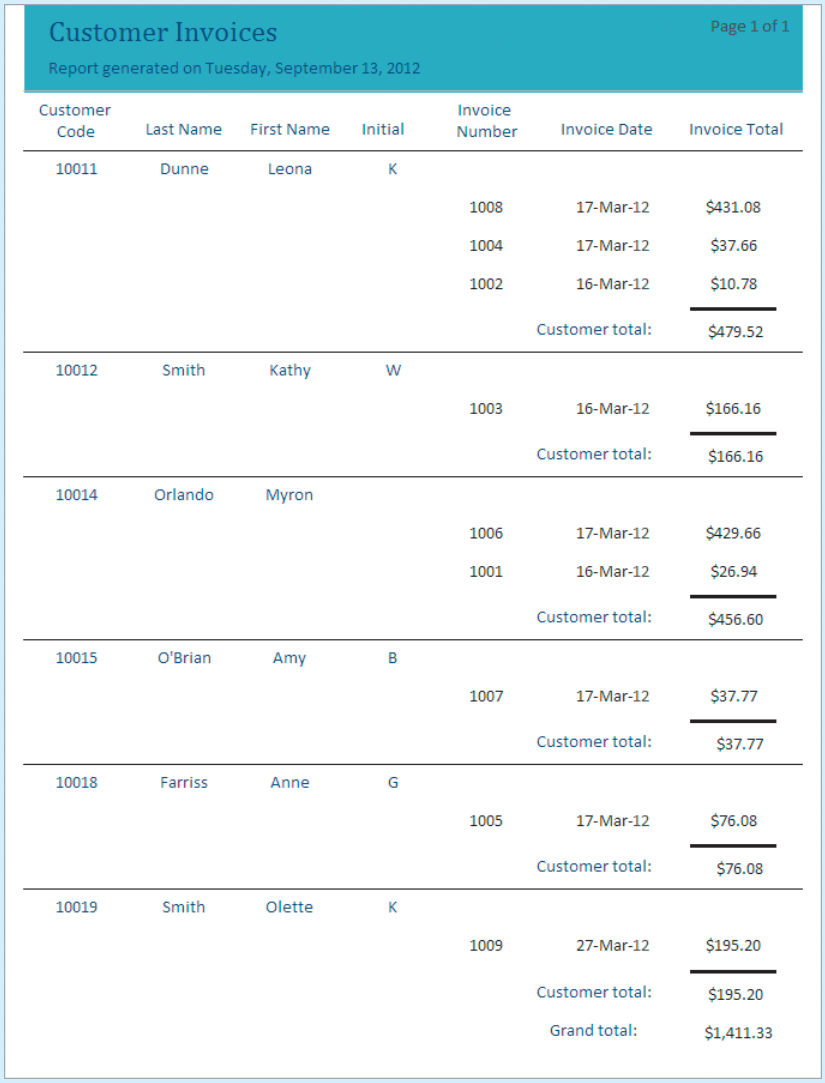
FIGURE M.181 Subtotals and totals



SOURCE: Course Technology/Cengage Learning

that the two new text boxes have different labels, the important difference between the two text boxes—Customer total and Grand total—is their ***location***. Figure M.182 shows the report output.

FIGURE M.182 Completed report output



Customer Invoices

Page 1 of 1

Report generated on Tuesday, September 13, 2012

Customer Code	Last Name	First Name	Initial	Invoice Number	Invoice Date	Invoice Total
10011	Dunne	Leona	K			
				1008	17-Mar-12	$431.08
				1004	17-Mar-12	$37.66
				1002	16-Mar-12	$10.78
					Customer total:	$479.52
10012	Smith	Kathy	W			
				1003	16-Mar-12	$166.16
					Customer total:	$166.16
10014	Orlando	Myron				
				1006	17-Mar-12	$429.66
				1001	16-Mar-12	$26.94
					Customer total:	$456.60
10015	O'Brian	Amy	B			
				1007	17-Mar-12	$37.77
					Customer total:	$37.77
10018	Farriss	Anne	G			
				1005	17-Mar-12	$76.08
					Customer total:	$76.08
10019	Smith	Olette	K			
				1009	27-Mar-12	$195.20
					Customer total:	$195.20
					Grand total:	$1,411.33

SOURCE: Course Technology/Cengage Learning

4.1.1 Special Report: Labels

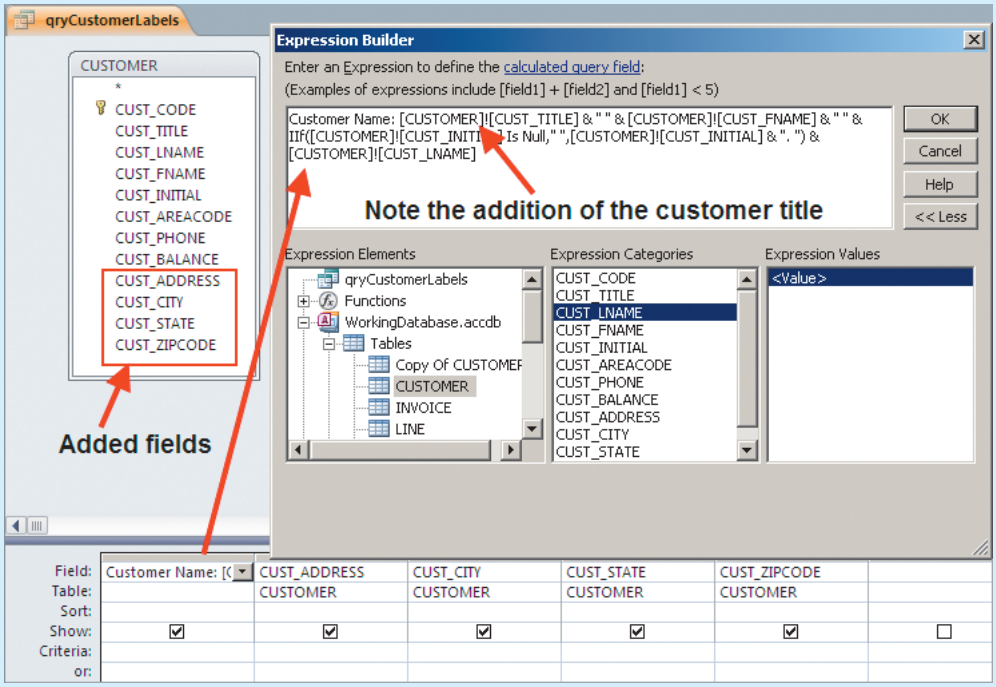
Printed customer address labels are a useful addition to any application set. (Such labels are useful when you mail out special promotions or billing statements.)

Access makes it easy to develop an effective and professional looking set of labels. Before creating the labels, make sure that you add the four marked fields in the CUSTOMER table as shown in Figure M.183. Then create the query shown in Figure M.183. This query will produce the output shown in Figure M.184.

You are now ready to create the label report. In this case, the label wizard is a good place to start. (You will be able to edit the label wizard's results later.) Figure M.185 shows the start of the process. (Note that the query named **qryCustomerLabels** was selected to be the data source for the labels.)

When you click the **Labels** button shown in Figure M.185, you'll see the label size selection dialog box in Figure M.186. As you can tell the current selection is an **Avery** label set, three across the label page, using labels that measure 1 ½" × 2½".

FIGURE M.183 Customer label query Design View



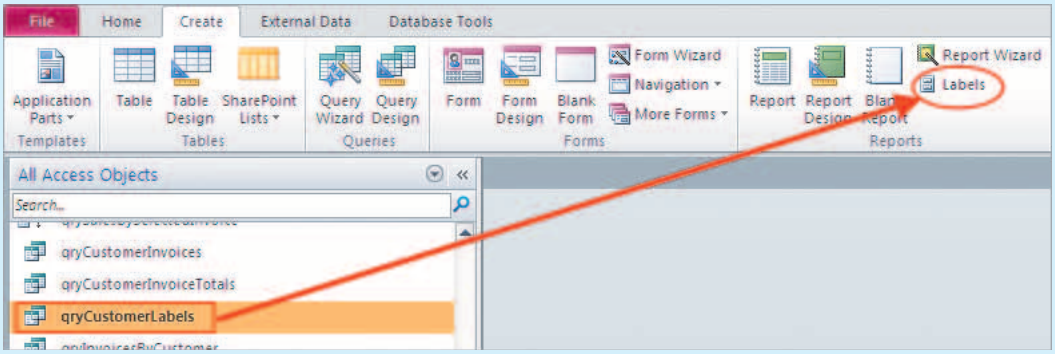
SOURCE: Course Technology/Cengage Learning

FIGURE M.184 Customer label query output

Customer Name	CUST_ADDRESS	CUST_CITY	CUST_STATE	CUST_ZIPCODE
Mr. Alfred A. Ramas	1124 Mountain View Rd.	Nashville	TN	37119
Ms. Leona K. Dunne	219 Twilight Lane	Nashville	TN	37123
Ms. Kathy W. Smith	389 Belle Glade Ct.	Beauville	KY	38976
Mr. Paul F. Olowski	1217 Main Street	Nashville	TN	37119
Col. Myron Orlando	3428 Mitchell Dr.	Nashville	TN	37123
Mrs. Amy B. O'Brian	2145 Meadow Lane	Nashville	TN	37228
Mr. James G. Brown	917 Twilight Lane	Nashville	TN	37119
Mr. George Williams	Box 2194	Smithville	TN	38003
Gen. Anne G. Farriss	453 Lotus Ct.	Carter	KY	38998
Rev. Olette K. Smith	890 Lockheed Avenue	Huntsville	AL	32892

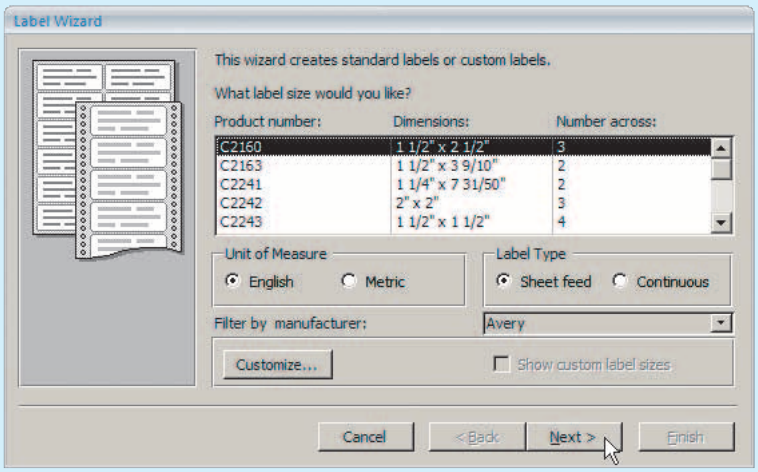
SOURCE: Course Technology/Cengage Learning

FIGURE M.185 Starting the label wizard



SOURCE: Course Technology/Cengage Learning

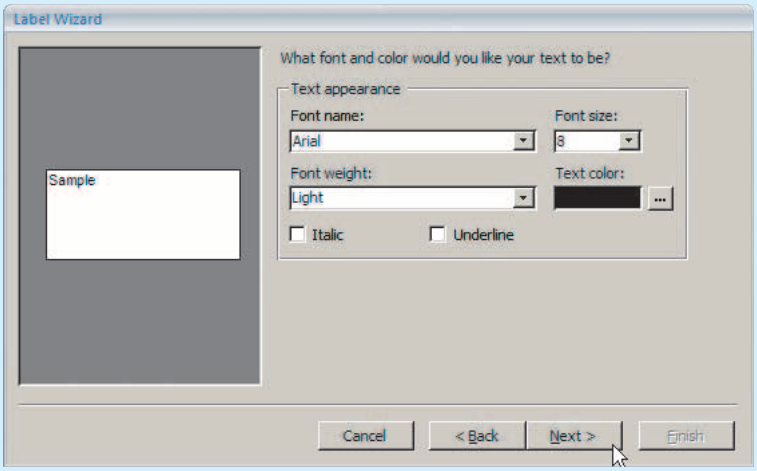
FIGURE M.186 Label size selection



SOURCE: Course Technology/Cengage Learning

Because this is a very popular label format, go ahead and accept this selection by clicking on the **Next >** button in Figure M.186 to generate Figure M.187. (As you can tell by looking at Figure M.186, there are quite a few label format options available—only five are shown here, but the scroll bar will reveal quite a few more.)

FIGURE M.187 **Font size selection**



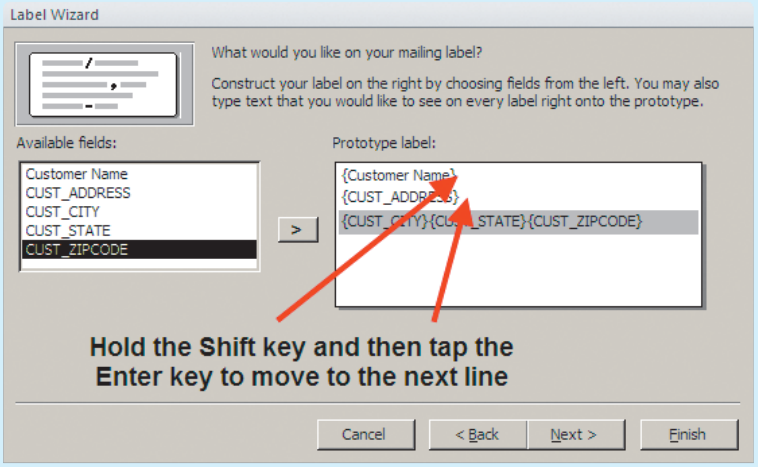
SOURCE: Course Technology/Cengage Learning

After selecting the label font in Figure M.187, click the **Next >** button in Figure M.187 to generate Figure M.188 to select the fields to be placed on the label.

To place a field on the label, select it and then click the **>** button shown in Figure M.188 to move it to the **Prototype label**. The annotation in Figure M.188 tells you how to get a line break so that you can move to the next label line. When you have moved all required fields to the **Prototype label** section, click the **Next >** button in Figure M.188 to generate Figure M.189 and then sort by zip code.

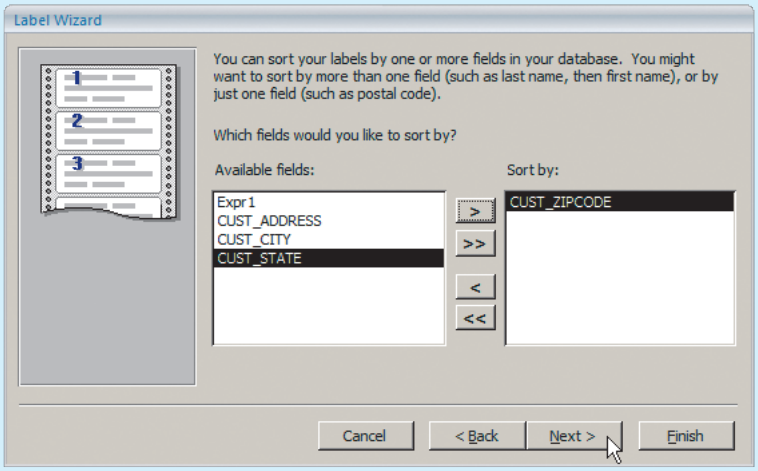
Then click the **Next >** button to generate Figure M.190 and then type the name for the label report.

FIGURE M.188 Field selection



SOURCE: Course Technology/Cengage Learning

FIGURE M.189 Sort selection

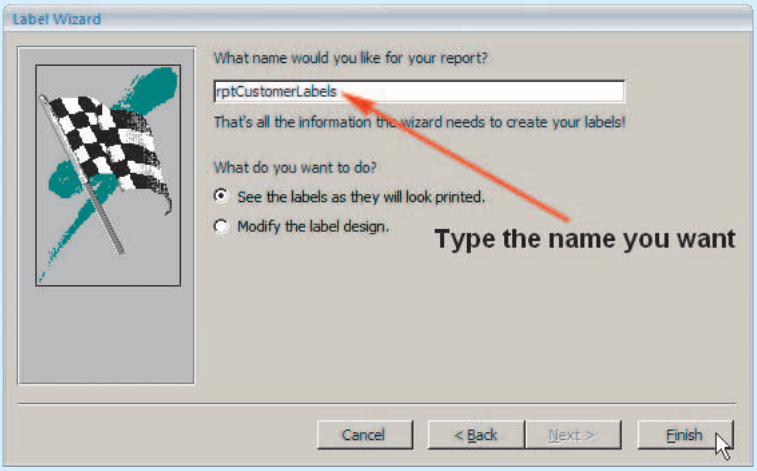


SOURCE: Course Technology/Cengage Learning

As you can tell by looking at Figure M.190, the name **rptCustomerLabels** was used to reflect its use of the **qryCustomerLabels** query as the report's data source. Click the Finish button in Figure M.190 to save the report and to see its label output in Figure M.191.

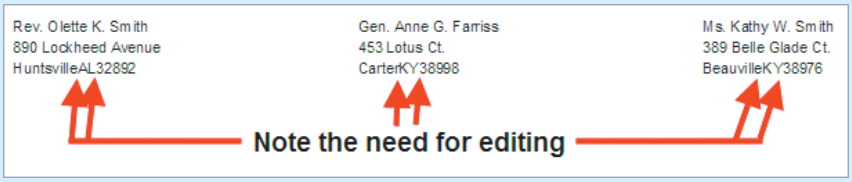
As you can tell by looking at Figure M.191, the output needs some touch-up editing. The output would have a much more professional look if there was some space between the city and the state and between the state and the zip code. Also, putting a comma after the city name is standard. Therefore, open the label report in its Design view and then generate the Property Sheet for the third text box. Scroll down to the **Control Source** and make the insertions shown in Figure M.192. (Use the **Expression Builder** to make the editing changes.)

FIGURE M.190 Type the report label name



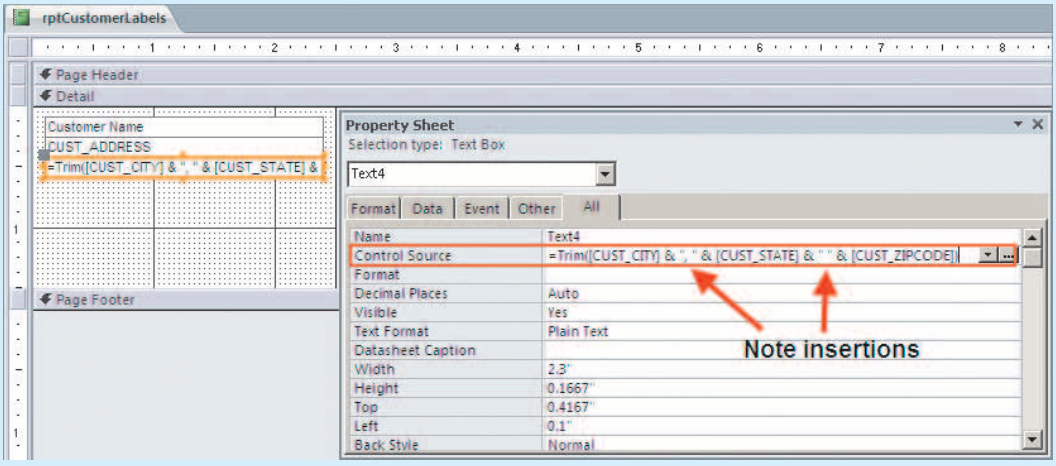
SOURCE: Course Technology/Cengage Learning

FIGURE M.191 Editing requirements



SOURCE: Course Technology/Cengage Learning

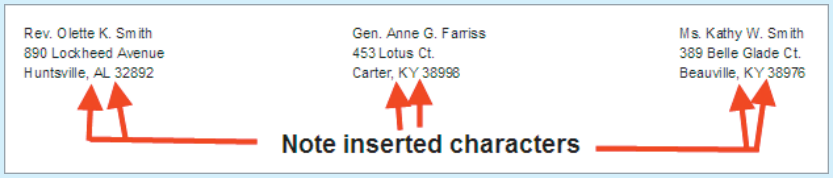
FIGURE M.192 Design View of edited label report



SOURCE: Course Technology/Cengage Learning

After you have completed the edits in Figure M.192, save the label report again and then open it in its **Print Preview** to see the output in Figure M.193. Note that the output has a much more professional look than the one you saw earlier in Figure M.191.

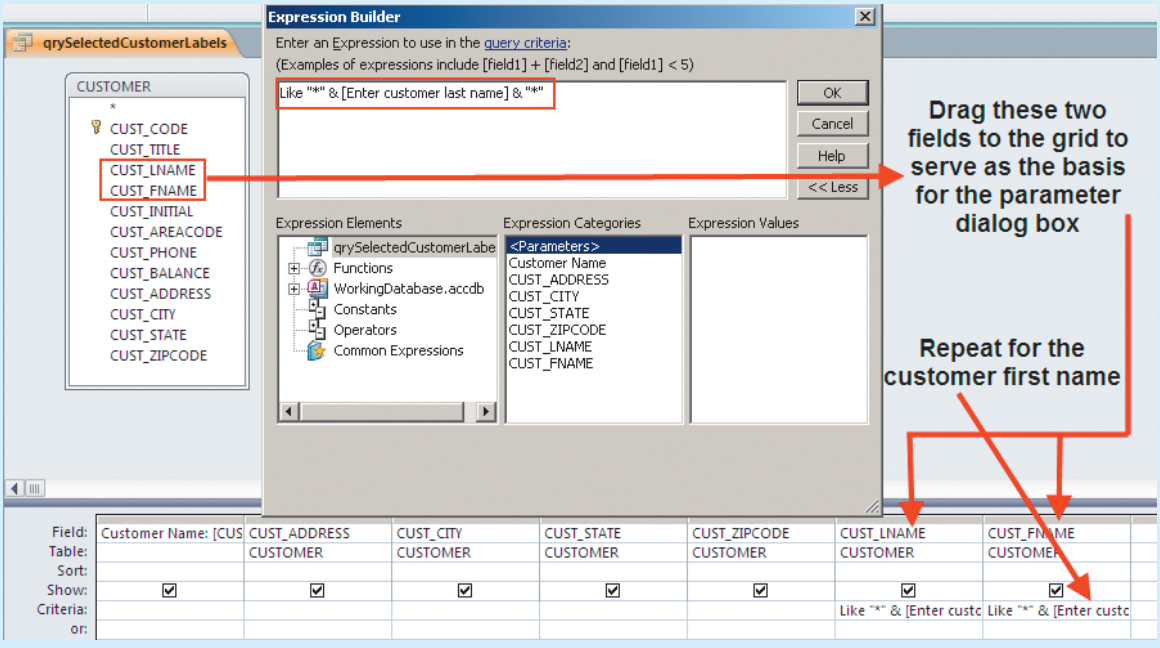
FIGURE M.193 Print preview of edited label report



SOURCE: Course Technology/Cengage Learning

Using the techniques you have learned in this tutorial, you can easily modify the query to enter last and first name parameter entries and then to use this modified query as the data source for a label report that prints labels for one or more selected customers. For example, save the **qryCustomerLabels** query as **qrySelectedCustomerLabels** and then make the changes you see in Figure M.193a. Next, save the **rptCustomerLabels** report as **rptSelectedCustomerLabels** and then use the new query as its data source. (See Figure M.193b.)

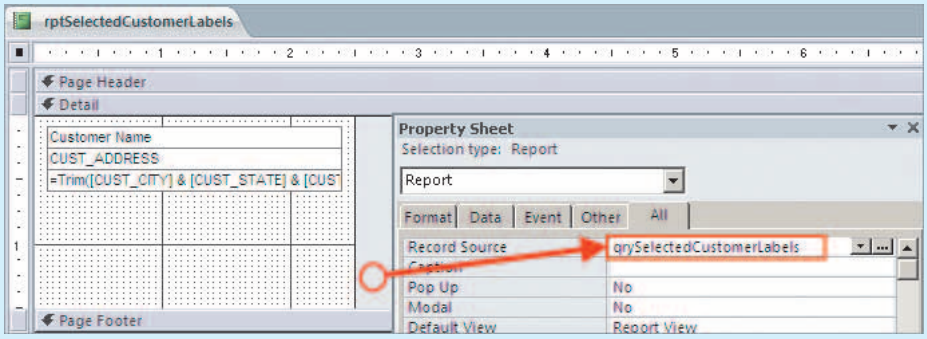
FIGURE M.193a Selected customers query



SOURCE: Course Technology/Cengage Learning

Now add three command buttons on the main menu form's **Reports** page, one for the customer invoice report, one for the customer labels, and one for selected customer labels.

FIGURE M.193b Selected customers label report



SOURCE: Course Technology/Cengage Learning

5.1 Macro Groups and the Submacro Blocks within Them

Macros are code sets that let you perform a wide variety of actions that range from opening and closing forms, queries, and reports to calculating and inserting values on a form. You can even manage pictures with the help of macros. Given the wide range of actions you can take with macros, this tutorial covers only a few options. However, once you have seen how macros are created and used, you have a sufficient basis for exploring additional macro options on your own. Ultimately, the most valuable contribution of macros is that they help you tie the many database applications together to create a system. In this tutorial, that system is based on the menu form you created in Section 3.1.6.

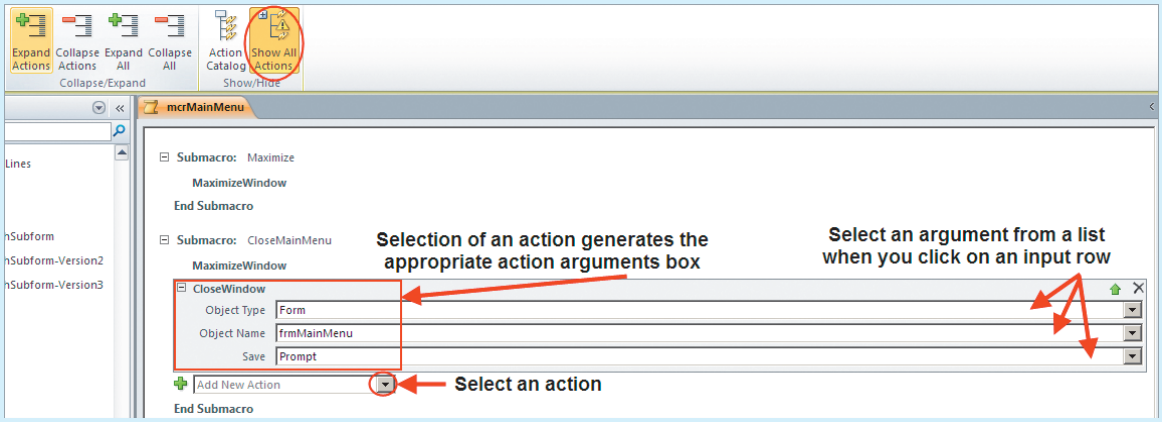
Remember that you should continue to strive to make the system self-documenting. Start with a plan to ***organize the macros*** you intend to create. The best way to organize macros is to make each set of related macros part of a named macro ***group***. The macro group name should reflect the group's contents. In this case, you have created a menu that will help you access queries, forms, and reports. At least one of the macros will control a general menu action, such as closing the menu when you are done. Because you are about to create ***macro*** groups, it seems reasonable to use **mcr** as a preface to the macro group names. Therefore, it would be appropriate to begin with four macro groups: **mcrMainMenu**, **mcrMainMenuQueriesPage**, **mcrMainMenuFormsPage**, and **mcrMainMenuReportsPage**.

To create a macro group, select the **Create** option from the now-familiar list of objects—see, for example, the list of objects in Figure M.175—and then select the **Macro** option to open the macro design screen you see in Figure M.194. (Click the **Show All Actions** button to ensure that all actions are available for you to use.)

Let's start with the general main menu form's anticipated actions. Let's suppose you want to make sure that the menu form is maximized when it opens and that you close the main menu form by clicking on the **Close Main Menu** button you created in Section 3, Figure M.174. Figure M.194 shows both submacro blocks, named **Maximize** and **CloseMainMenu**, respectively. In the interest of self-documentation, remember to use descriptive names that reflect the submacro's activities. The submacros are contained within a macro group that already has the **mcr** prefix, so the submacro names within the macro group do not need to have the prefix repeated. Access will automatically use the macro group name to label each submacro block within that group. Therefore, Access will reference the **Maximize** submacro in the **mcrMainMenu** group as **mcrMainMenu.Maximize**. Do not use "group blocks" right now because these are only useful for organizational purposes and cannot be referenced the same way as "submacro blocks."

To create the submacro named **Maximize,** start by selecting MaximizeWindow from the dropdown menu labeled Add New Action. This will create a new action in the macro group, but it needs to be contained in a submacro block. To do

FIGURE M.194 Design View of macro group



SOURCE: Course Technology/Cengage Learning

this, right click the newly added action and select **Make Submacro Block**. Now you can give the submacro block its name: **Maximize.**

The second submacro block will close the main menu, so add another action using the same dropdown box as before. Make sure that you add the action to the macro group and not to the submacro block named **Maximize**. Select the **CloseWindow** action from the dropdown box, then right click the new action to convert it to a submacro block. You can also ***type*** **CloseWindow**, without selecting it from the option list. In fact, if you type the first few characters of any action command, Access will fill in the remaining characters. Name this submacro block **CloseMainMenu**. This block will include two actions. First, it will maximize the screen—just in case you decided to restore the screen manually—and, second, it will close the form.

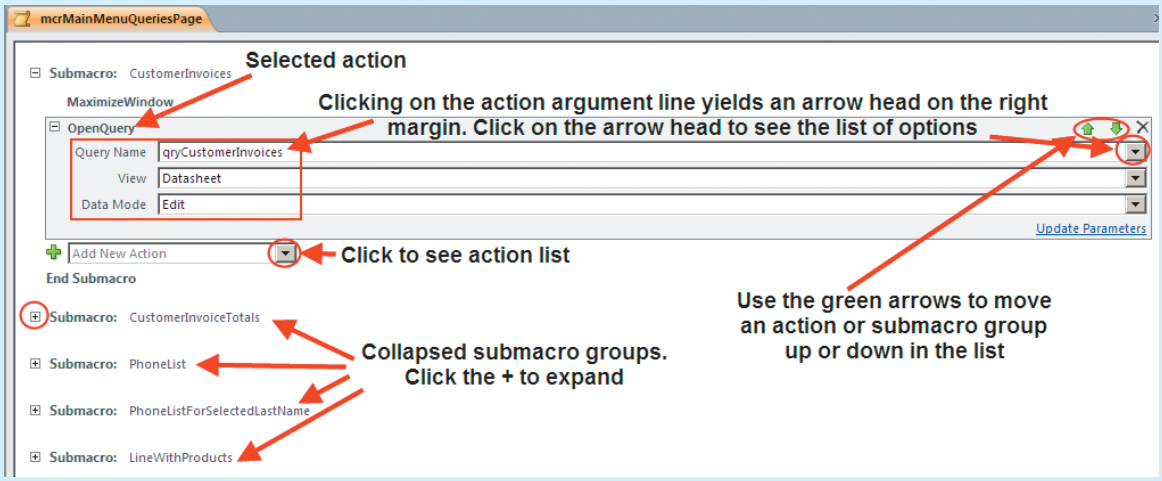
The **CloseWindow** action triggers an action argument box such as the one you see in Figure M.194. In this case, the action argument has three components: an **Object Type**, an **Object Name**, and a **Save**. Clicking on the down arrow next to **Object Type** will produce a list of object types. Because this action involves the main menu ***form***, select the **Form** option as shown here. Select the **Object Name** the same way—the object is a form named **frmMainMenu**, so that's the one you select from the list. The **Save** option has a **Prompt** default selection, so go ahead and keep that default.

Next, add the **MaximizeWindow** action to the submacro block **CloseMainMenu**. To do this, first click the submacro block to select it (you should see a grey box around the whole block). Then select **MaximizeWindow** from the "Add New Action" menu inside of that grey box. The new action should be executed before the **CloseWindow** action, so use the green up arrow to move the action to the top of the submacro (or use the **CloseWindow** action's green down arrow to move that action down).

Save the new macro group as **mcrMainMenu**. That's it. You can now create the remaining macro groups the same way. Figure M.195 shows the submacro blocks in the macro group named **mcrMainMenuQueriesPage**. Note that the **OpenQuery** action triggers an **Action Arguments** box that contains a **Query Name**, the desired **View**, and the **Data Mode**.

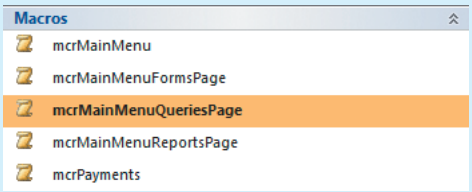
Repeat the process for the forms and the reports. Naturally, when you select the action **OpenForm** from an action list, the action arguments box contains references to ***form*** actions and when you select the action **OpenReport** from an action list, the action arguments box contains references to ***report*** actions. Save each macro group, using the macro group names suggested earlier in this section. When you are done, your macro groups will show up as listed in Figure M.196.

FIGURE M.195 The Queries macro group



SOURCE: Course Technology/Cengage Learning

FIGURE M.196 Completed macro groups



SOURCE: Course Technology/Cengage Learning

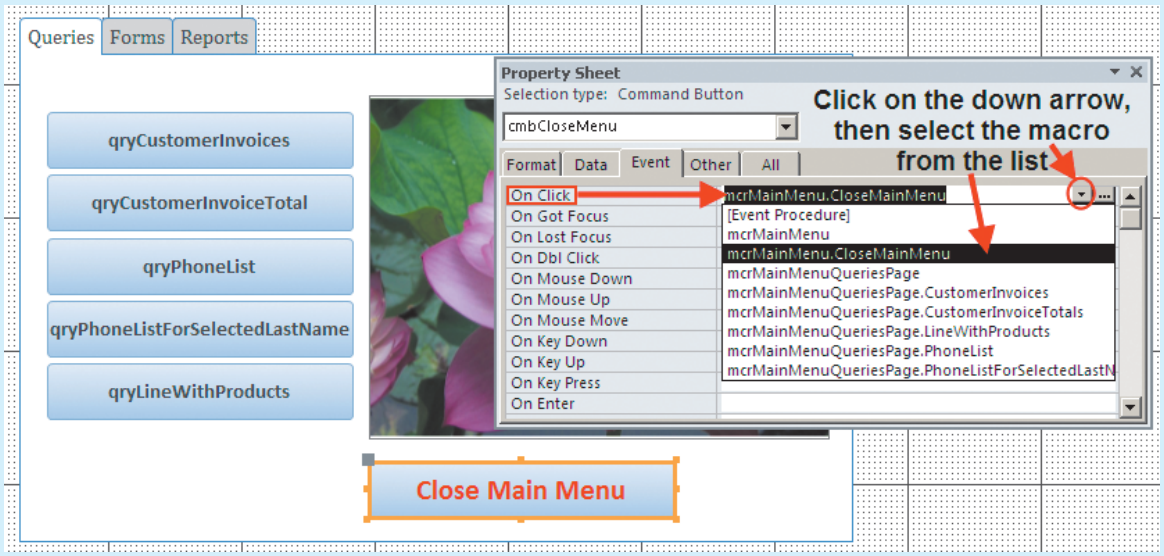
5.1.1 Attaching the Macro Blocks

To execute each macro within a macro group, it must be attached to the object that will trigger it. (Note: for the remainder of the tutorial we will refer to submacro blocks simply as "macros".) Let's take a look at the **Close Main Menu** button on the Main Menu form. When you click that button, the main menu form should close. Therefore, note that the macro will be triggered via the "On Click" property in the properties box for that button. Figure M.197 summarizes the process of attaching the macro to the **Close Main Menu** button on the **Main Menu** form.

Now open the main menu in form view and click the **Close Main Menu** command button and note that the macro closes the form as intended.

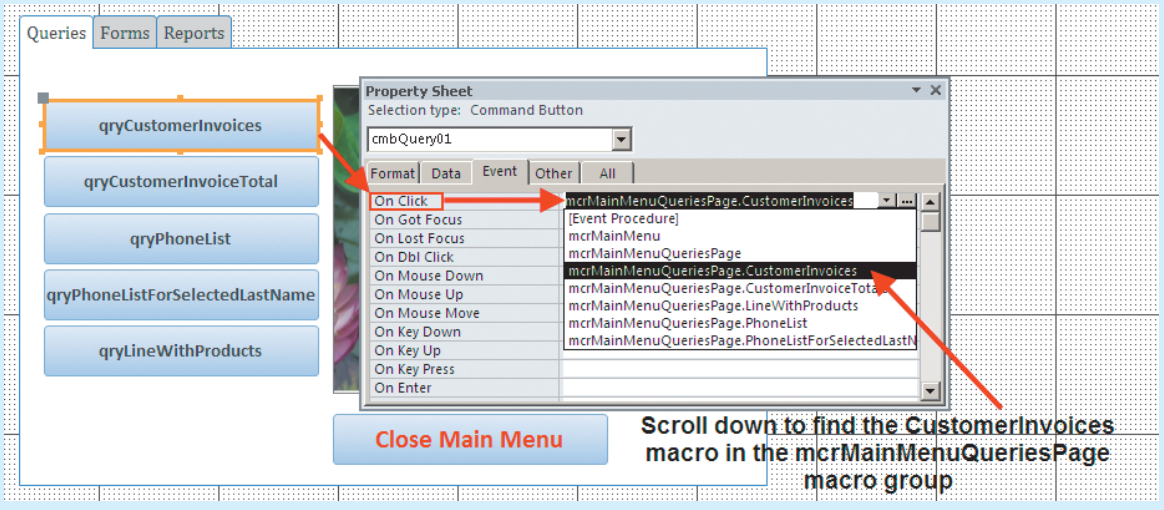
Use the same procedure to connect the appropriate macros in the **mcrMainMenuQueriesPage** and attach them to the command buttons on the main menu's query page. Figure M.198 summarizes the procedure.

FIGURE M.197 Attach the CloseMainMenu macro



SOURCE: Course Technology/Cengage Learning

FIGURE M.198 Attach the CustomerInvoices macro



SOURCE: Course Technology/Cengage Learning

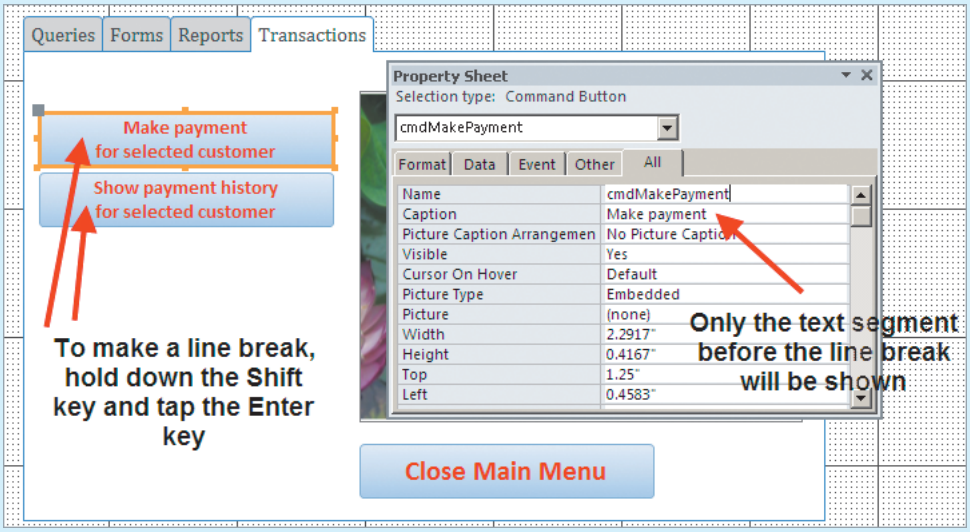
Repeat the process for the forms and the reports. You have now created a simple menu-driven system. Naturally, you can place command buttons on the forms and reports to let you close them as well as open them. (At this point, you have to close the forms and reports by using the Windows close option on the screens.)

5.1.2 A More Complex Set of Macros

Thus far, you have seen how macros can be used to perform simple actions, such as open and close queries, forms, and reports. However, macros can do a lot more than that. In this last set of examples, you will create some macros in a macro group named **mcrPayments** to manage customer payment on account.

Let's start by creating a new **Transactions** page with a **Make payment** command button on the main menu form. The new page and its command button are shown in Figure M.199. If necessary, review Figure M.162 to see how you can insert a menu page.

FIGURE M.199 Main menu transactions page



SOURCE: Course Technology/Cengage Learning

Now that you have created the new menu page, map out what you want the command buttons to do. The **Make payment** caption on the first command button shown in Figure M.199 is largely self-documenting. But what precisely do you want to manage through this command option? Before you try to write the macros to accomplish the intended tasks, write a short, but precise, narrative. In this example, the narrative spells out what is to be done:

1. Find the customer who wants to make a payment.
2. When the correct customer is found, make a payment entry.
3. When the payment is made, the customer balance must be updated to reflect the payment.
4. The end user must be able to track all payments by customer, date, and amount.
5. The end user should enter only the payment amount—customer numbers, date entries, and new balance calculations must be done automatically.

Item 4 in the preceding list makes it clear that you need to create a new table in which to store all the payment transactions. Therefore, create a new PAYMENT table that includes the attributes shown in Figure M.200. Note that the CUST_CODE is the foreign key in this new table.

Make sure that you relate this PAYMENT table to the CUSTOMER table, so your relationships window should look like Figure M.201.

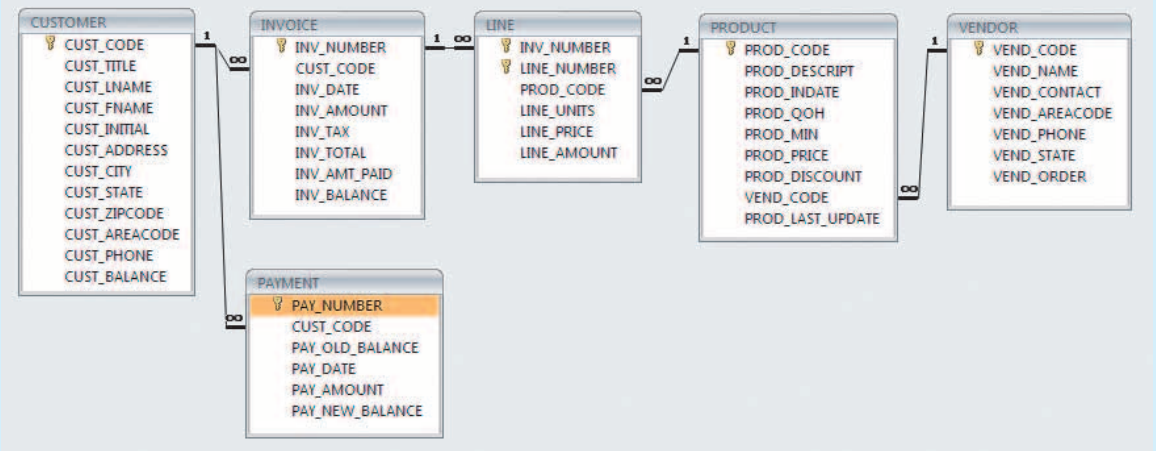
FIGURE M.200 PAYMENT table structure

Field Name	Data Type
PAY_NUMBER	AutoNumber
CUST_CODE	Number
PAY_OLD_BALANCE	Currency
PAY_DATE	Date/Time
PAY_AMOUNT	Currency
PAY_NEW_BALANCE	Currency

SOURCE: Course Technology/Cengage Learning

If you click the **Make payments** command button shown in Figure M.199, the first thing you will have to do is find the customer for whom the payment is to be recorded. Therefore, you must first design the **frmFindCustomerForPayment** form you see in Figure M.202. This form's contents will be based on a parameter query named **qryFindCustomerForPayment**. This query uses all the CUSTOMER table fields. To enable the end user to select a customer by that customer's last name, use the following criteria statement for the CUST_LNAME field:

FIGURE M.201 The updated Relationships window



SOURCE: Course Technology/Cengage Learning

Like "*" & [Enter the last name] & "*".

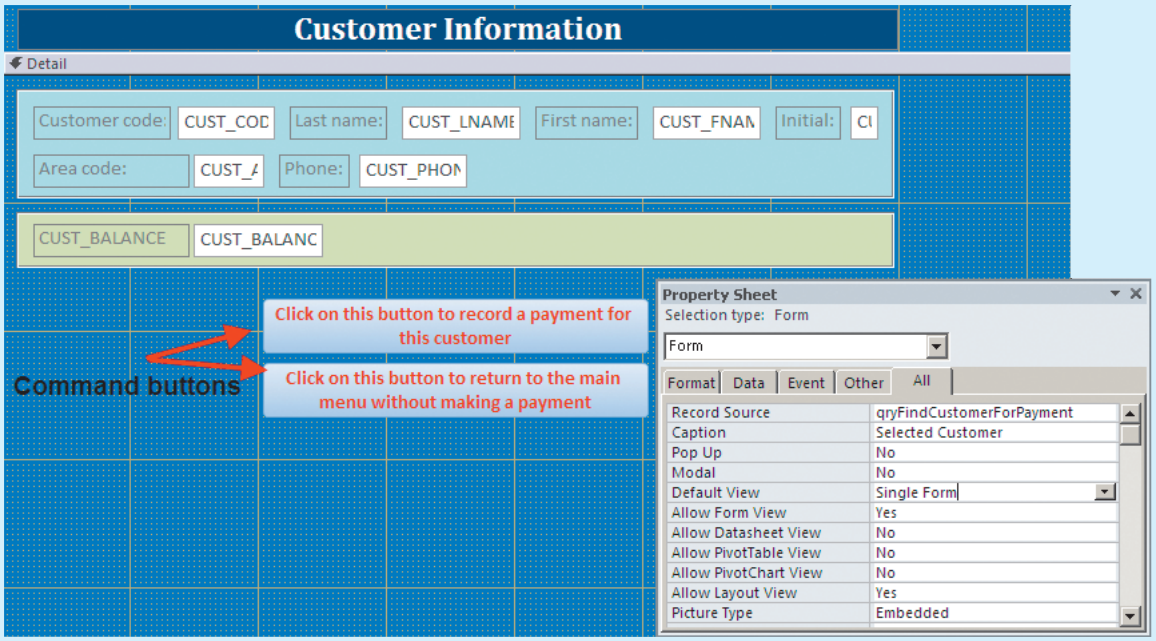
Note that the form shown in Figure M.202 uses two command buttons. The first command button will be used to execute a macro that lets the end user record a customer payment. (You will see how this macro works after the remaining form components are completed.) The second command button shown in Figure M.202 lets the end user return to the menu without recording a payment. Clicking this second command button merely closes the **frmFindCustomerForPayment** form and returns you to the menu.

NOTE

If you open the **frmFindCustomerForPayment** form from the menu ***without closing the menu***, that form simply covers up the menu form. (That's why you want to maximize each form.) Therefore, if you close the **frmFindCustomerForPayment** form, the menu form becomes visible again. Because all of a form's components, including its data components, are available in the computer's memory, such "stacking" becomes a very handy tool to transfer values form one form to another.

If the end user clicks on the first command button shown in Figure M.202, the macro must open a form that can record the payment. Therefore, make this form now. (Its structure is shown in Figure M.203.) Note that the form's data source is the PAYMENT table. (The data source for any form can be found by clicking outside its form limits.)

FIGURE M.202 frmFindCustomerForPayment Design View



SOURCE: Course Technology/Cengage Learning

You want to ensure that the end user makes as few data entries as possible. For example, the payment number is entered through the **Autonumber** format and the payment date (PAY_DATE) uses the **Medium Date** default format. You will use a macro to automate the process of entering the customer code and the current payment balance. You will also use a macro to calculate the updated payment balance. Therefore, the only data entry will be the payment amount.

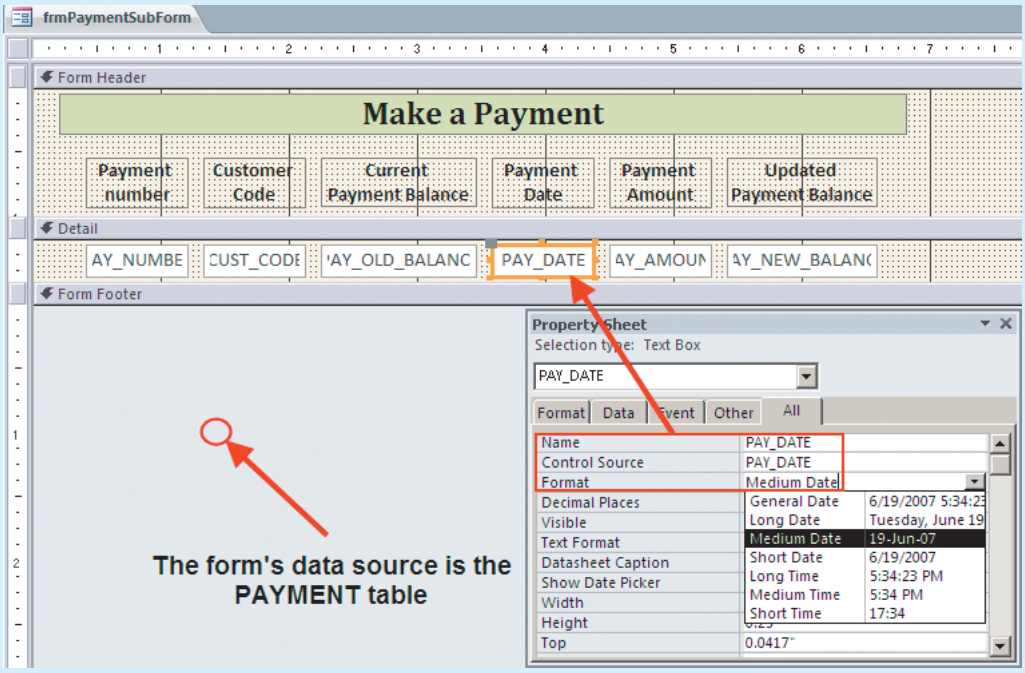
If the end user clicks on the second command button shown in Figure M.199, the customer information and the payment history must be shown. Therefore, create the appropriate main form as shown in Figure M.204 and the subform as shown. Create the main form (**frmShowPayments**) first and then create the subform named **frmShowPaymentSubform**. Note the link between the main form and its subform in Figure M.204. The data source for the main form is a query named **qryFindCustomerForPayment** and the subform data source is the table named **PAYMENT**.

5.1.3 Payment Options Organization

All the macros required to process the payment transaction will be stored in the **mcrPayments** macro group shown in Figure M.205 and M.205a.

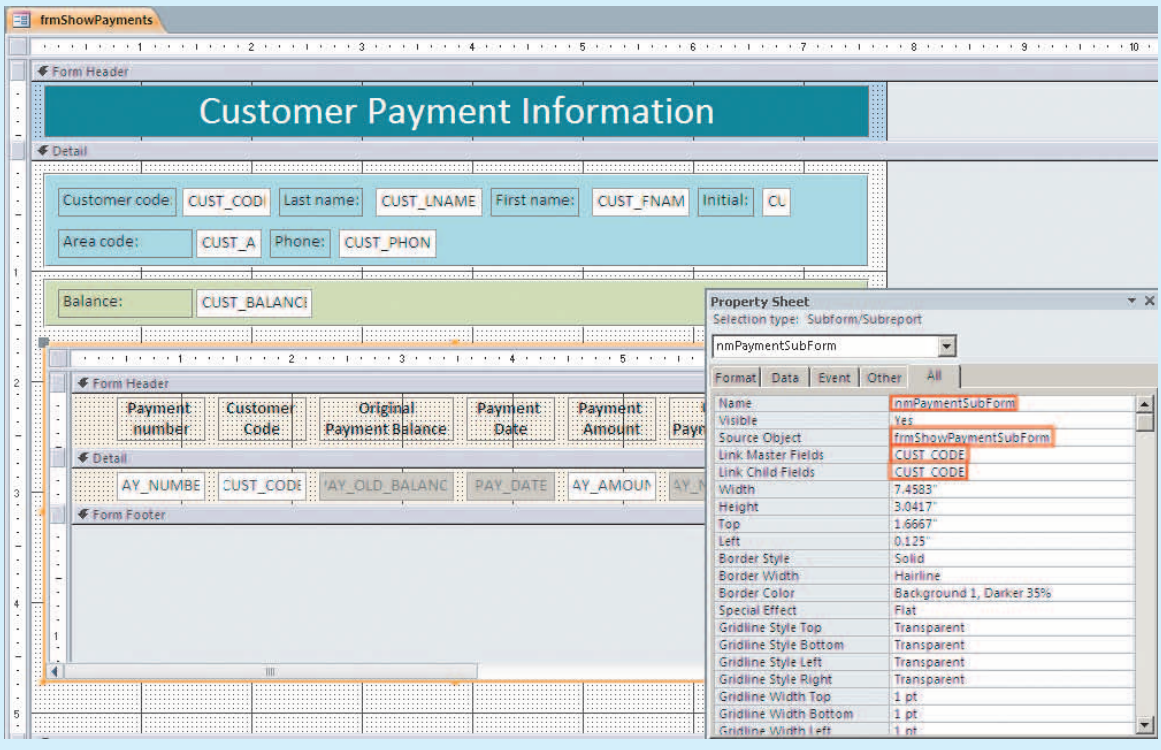
Take a few moments to read the contents of all the macro components in Figures M.205 and 205a. Note that the **MakePayment** macro opens the **frmMakePayment** form.

FIGURE M.203 The frmMakePayment form



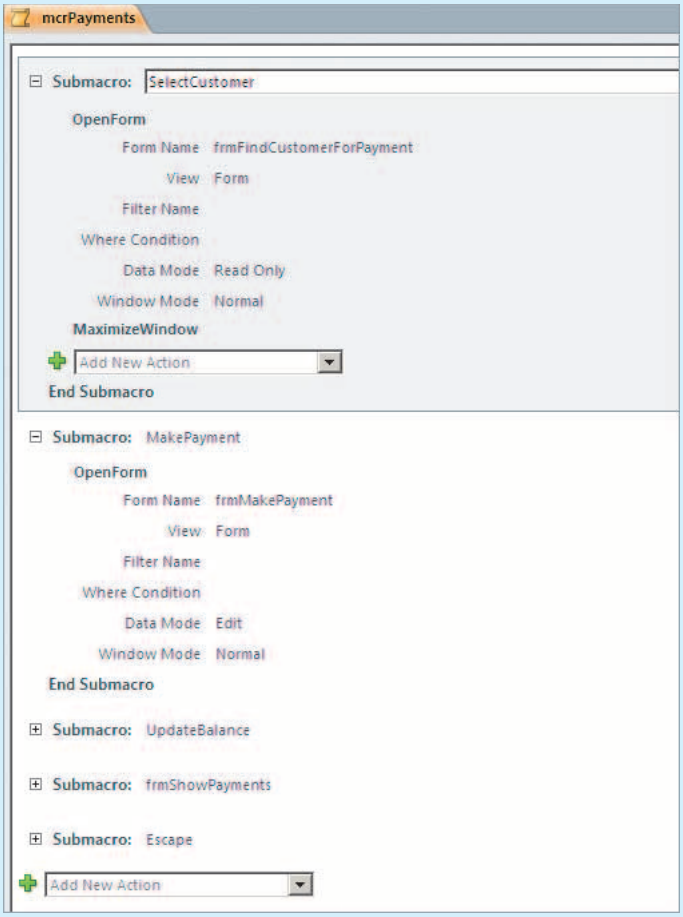
SOURCE: Course Technology/Cengage Learning

FIGURE M.204 The frmShowPayments form with subform



SOURCE: Course Technology/Cengage Learning

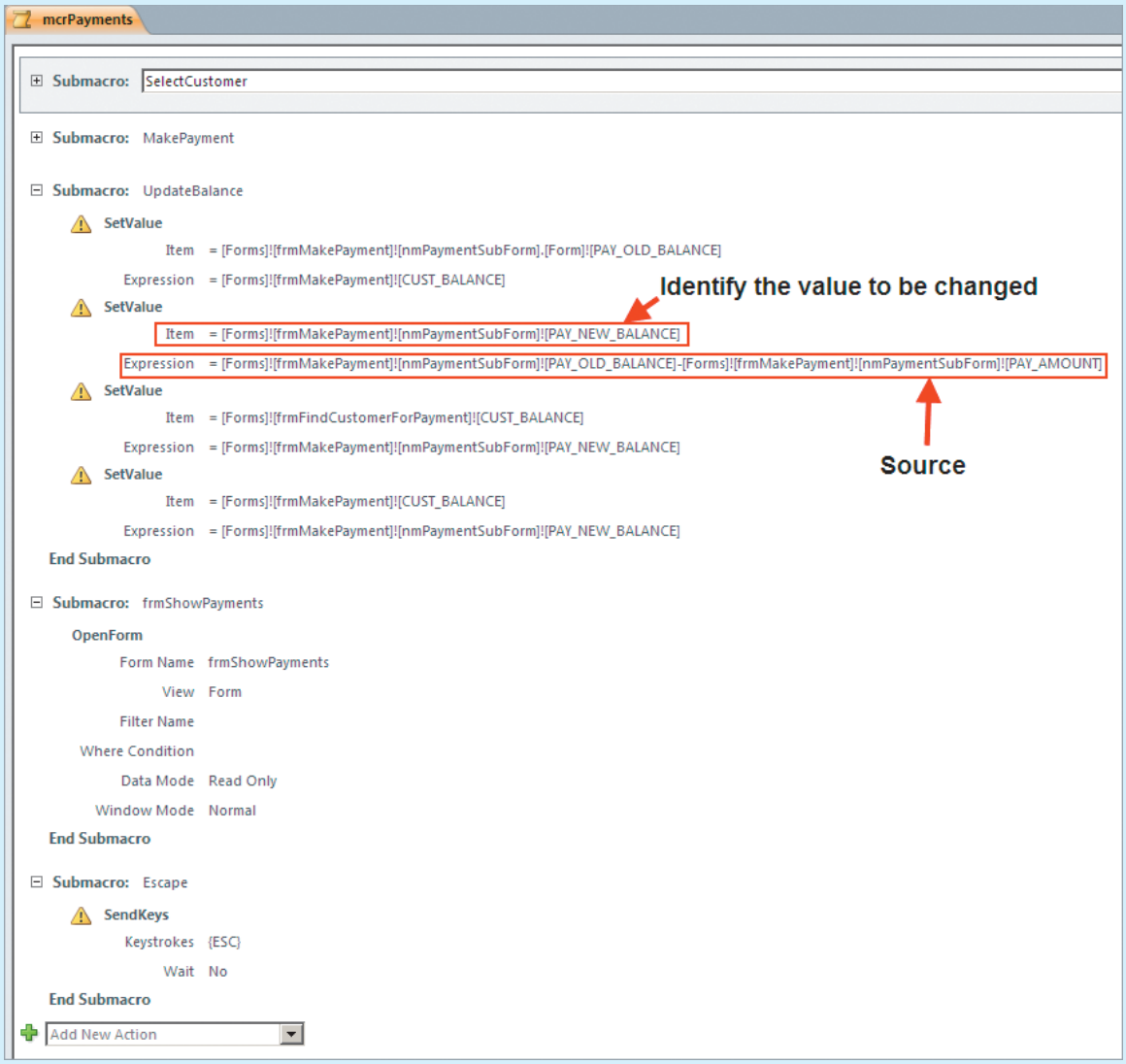
FIGURE M.205 The macros in the mcrPayments macro group



SOURCE: Course Technology/Cengage Learning

The **UpdateBalance** macro shown in Figure M.205a takes the PAY_AMOUNT value from the form named **frmMakePayment** and subtracts it from the PAY_OLD_BALANCE on the **frmMakePayment** form. (Note the second **SetValue** action's **Item:** and **Expression:** lines in the **UpdateBalance** macro.) It then updates the CUST_BALANCE in **frmMakePayment** and **frmFindCustomerForPayment**.

FIGURE M.205a More macros in the mcrPayments macro group

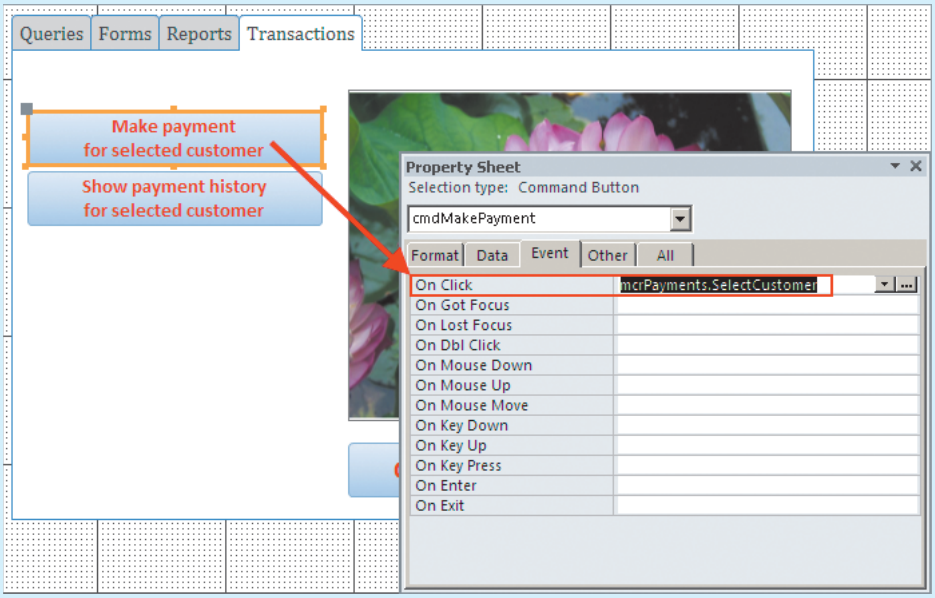


SOURCE: Course Technology/Cengage Learning

Once the macros in the **mcrPayments** macro group have been written, you must attach them to the command buttons you created on the main menu. For example, Figure M.206 shows that the **SelectCustomer** macro was attached to the **Payments** command button, using the **On Click** format. After you have attached the macro as shown in Figure M.206, open the main menu form and click the **Payments** command button to generate the parameter entry box shown in Figure M.207. Note that the macro worked as intended.

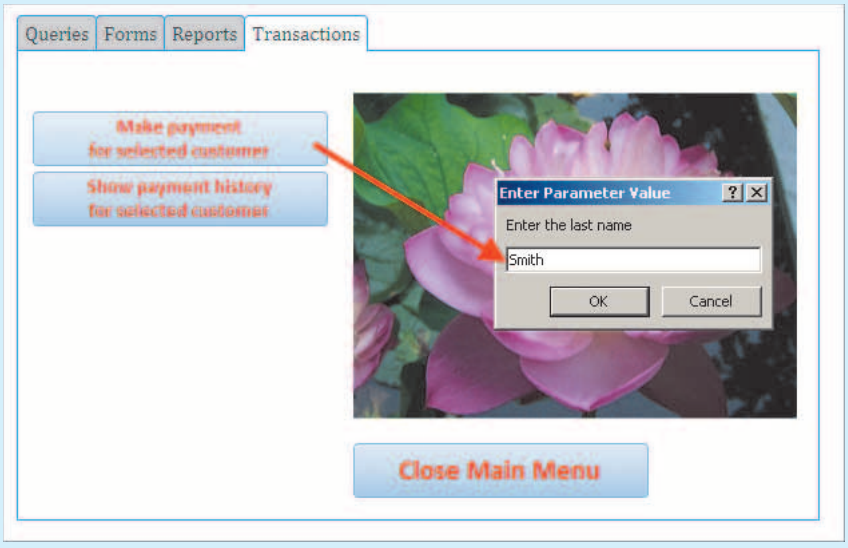
Making the **Smith** entry shown in Figure M.207 will open the form shown in Figure M.208. (Note that the form shows two records, because the CUSTOMER table contains two customers whose last name is Smith.) After deciding that this first record is the one you intend to update, click the first command button to trigger the next event, which is the payment transaction that will record a payment for customer Kathy Smith's account.

FIGURE M.206 Macro payment attachment



SOURCE: Course Technology/Cengage Learning

FIGURE M.207 Payment selection

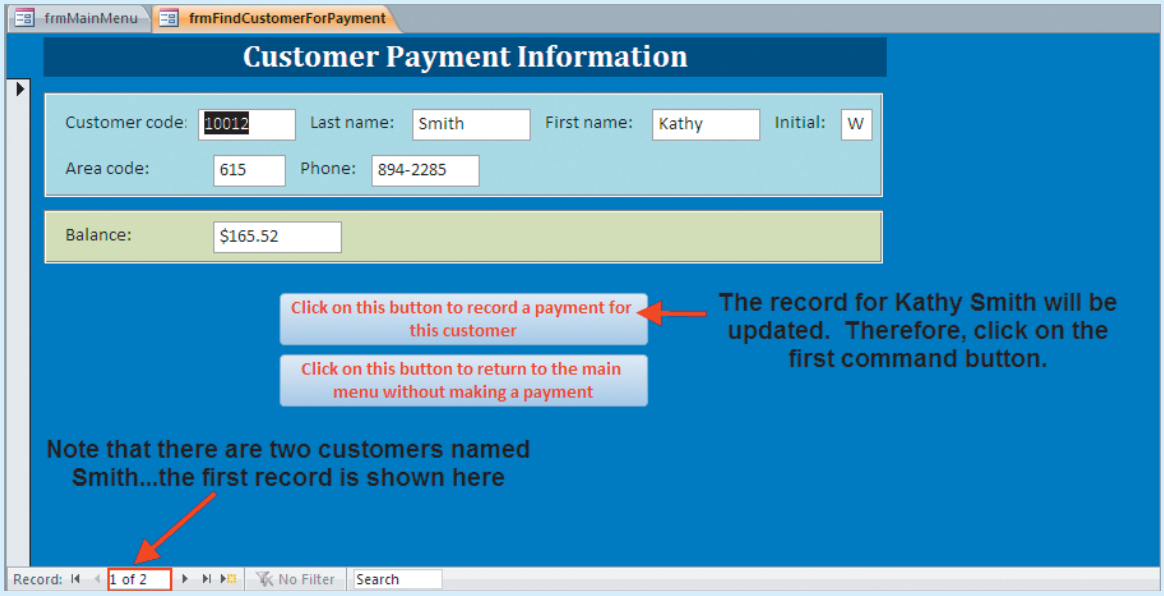


SOURCE: Course Technology/Cengage Learning

Clicking the first command button shown in Figure M.208 will open the payment form containing the subform shown in Figure M.209. Review the **MakePayment** and **UpdateBalance** macros in Figure M.205 to see what actions will be taken. You will discover that the macros will perform their intended tasks … note that the payment of $100 will update the original $165.52 balance to yield the updated payment balance of $65.52.

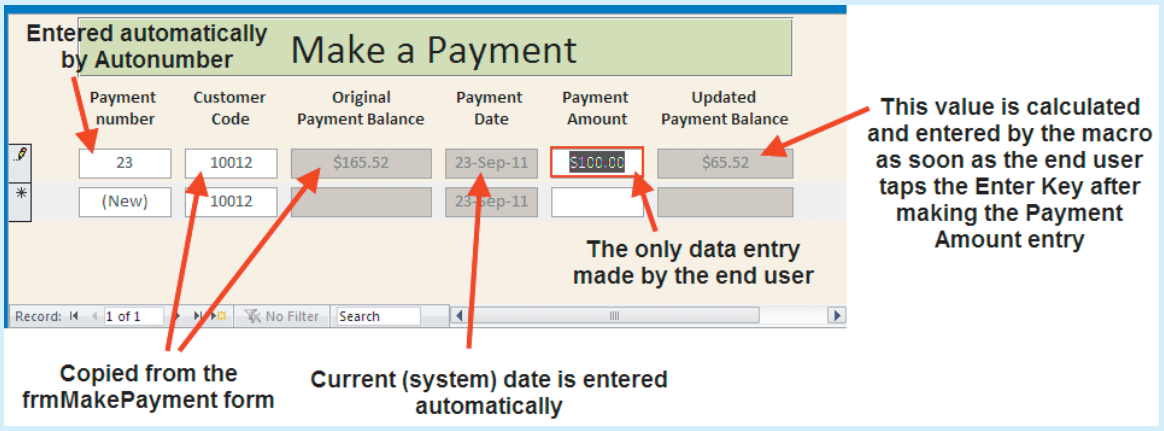
The macro attachment for the **Payment Amount** is shown in Figure M.209a. Note that the calculation is to be made **After Update**, because you cannot calculate the balance until after you have made the payment amount entry.

FIGURE M.208 Selected customer



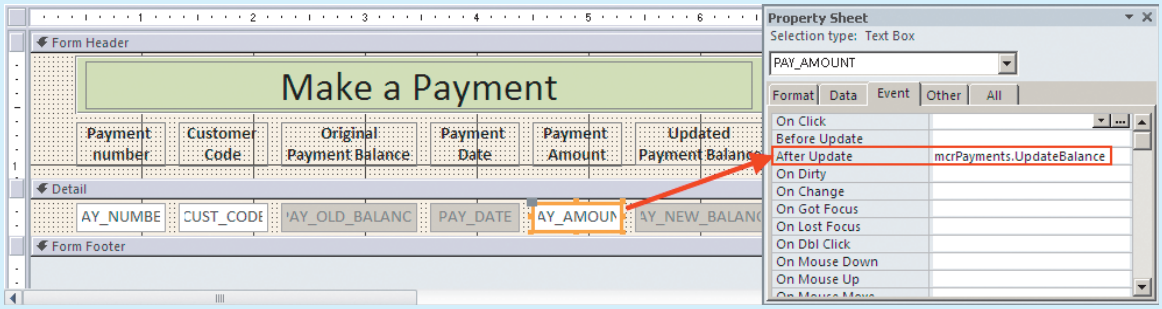
SOURCE: Course Technology/Cengage Learning

FIGURE M.209 Payment entry events



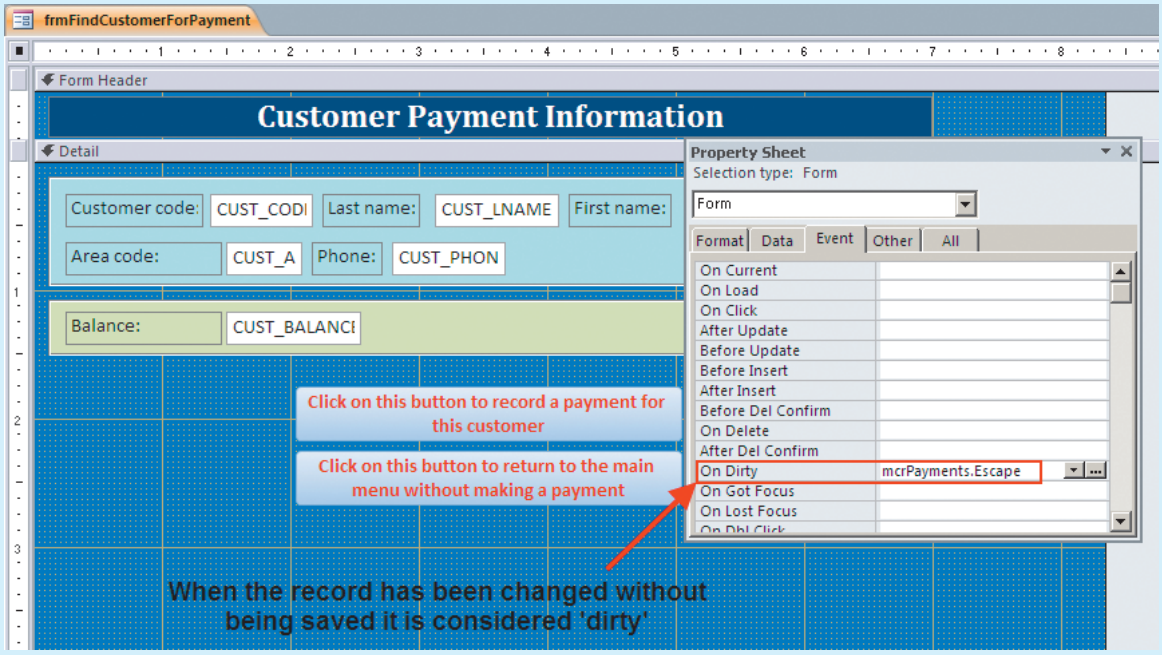
SOURCE: Course Technology/Cengage Learning

FIGURE M.209a Balance update macro attachment



SOURCE: Course Technology/Cengage Learning

FIGURE M.209b Send the escape key

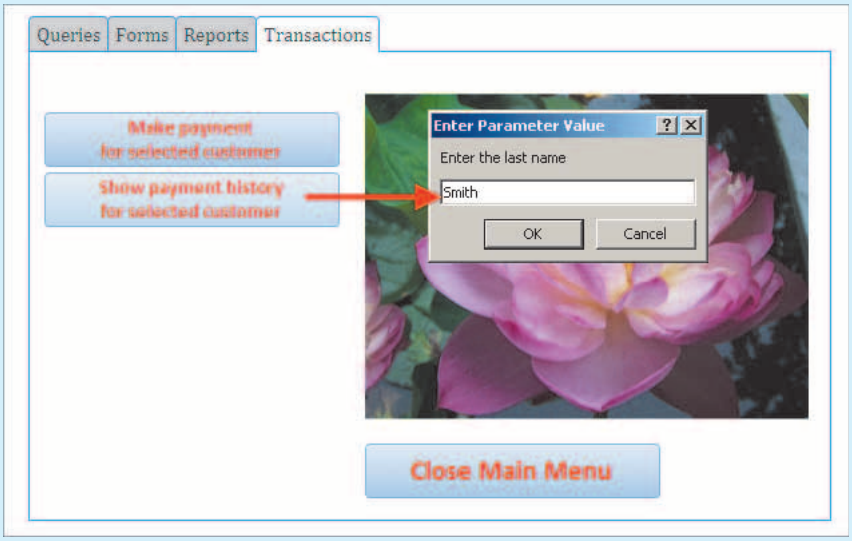


SOURCE: Course Technology/Cengage Learning

After making a payment and exiting the **MakePayment** form, the form called **FindCustomerForPayment** will still have an open record so the user will not be able to close the form. To close the record the user must press the escape key, or use a macro to send the key automatically. To set up this option, create a macro called **Escape** shown in Figure M.205a. Add this macro to the form's **On Dirty** field to send the escape key after the balance has been updated. This process is shown in Figure M.209b.

Now close **frmFindCustomerForPayment** to return to the main menu. To track the payment history for any customer, use the second command button on the main menu form shown in Figure M.210. The second command button shown here uses the **frmShowPayments** macro—see Figure M.205a—to ensure that the selected customer is found and that this customer's payment record will be shown.

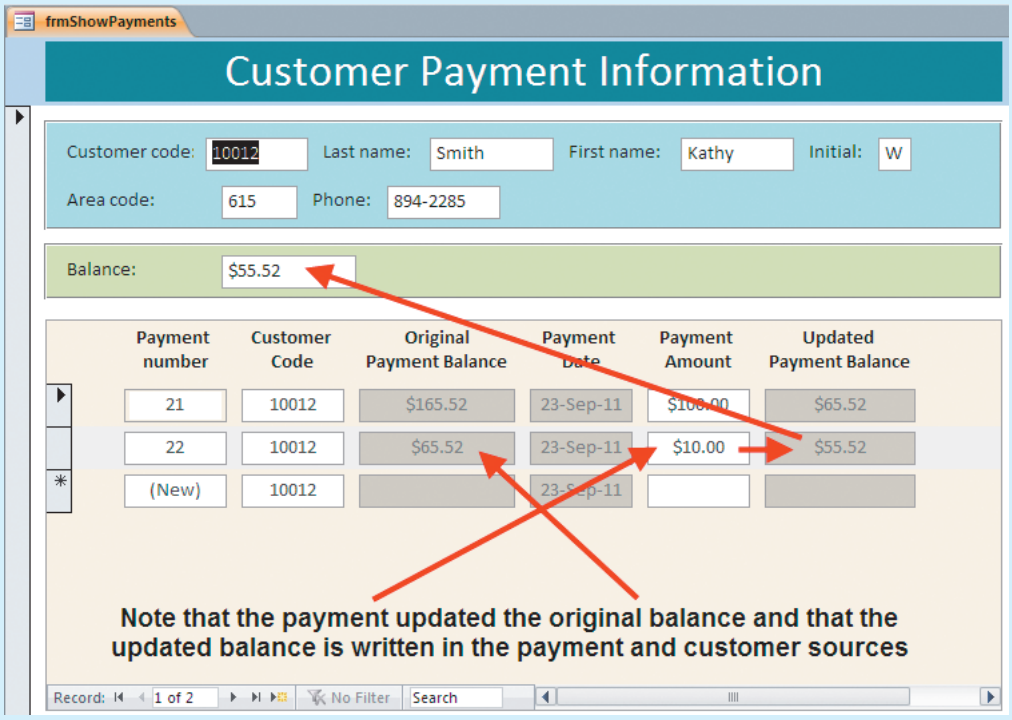
FIGURE M.210 Selection of current balance option



SOURCE: Course Technology/Cengage Learning

Note that the last name entry **Smith** shown in Figure M.210 yields the results shown in Figure M.211. Because Kathy Smith is the only customer for whom a payment entry was made thus far, only one record in the PAYMENT table matches the **Smith** last name entry.

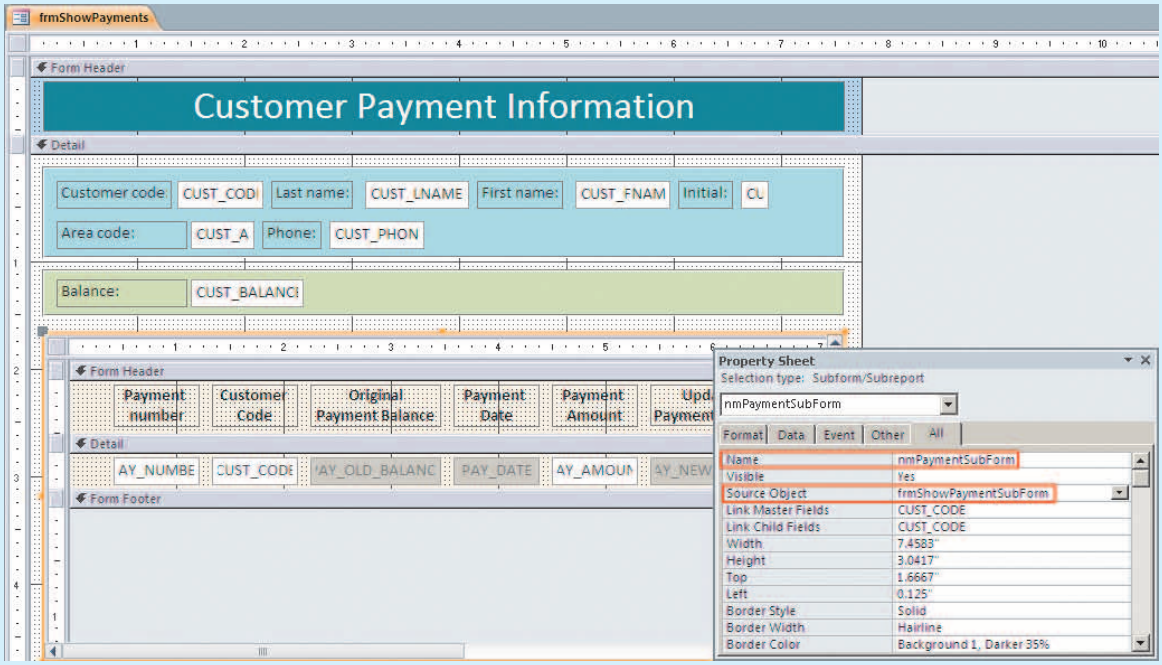
FIGURE M.211 Updated customer payment balance



SOURCE: Course Technology/Cengage Learning

As you examine Figure M.211, note that a form/subform structure was used to produce a detailed view of all the relevant information. The main form is basically a copy of the original customer form and the subform is basically a copy of the payments form you created earlier. The form/subform structure is shown in Figure M.212. If you have used a copy of the payment form shown in Figure M.209 that still includes the update macro, you can make payments via this form/subform, too.

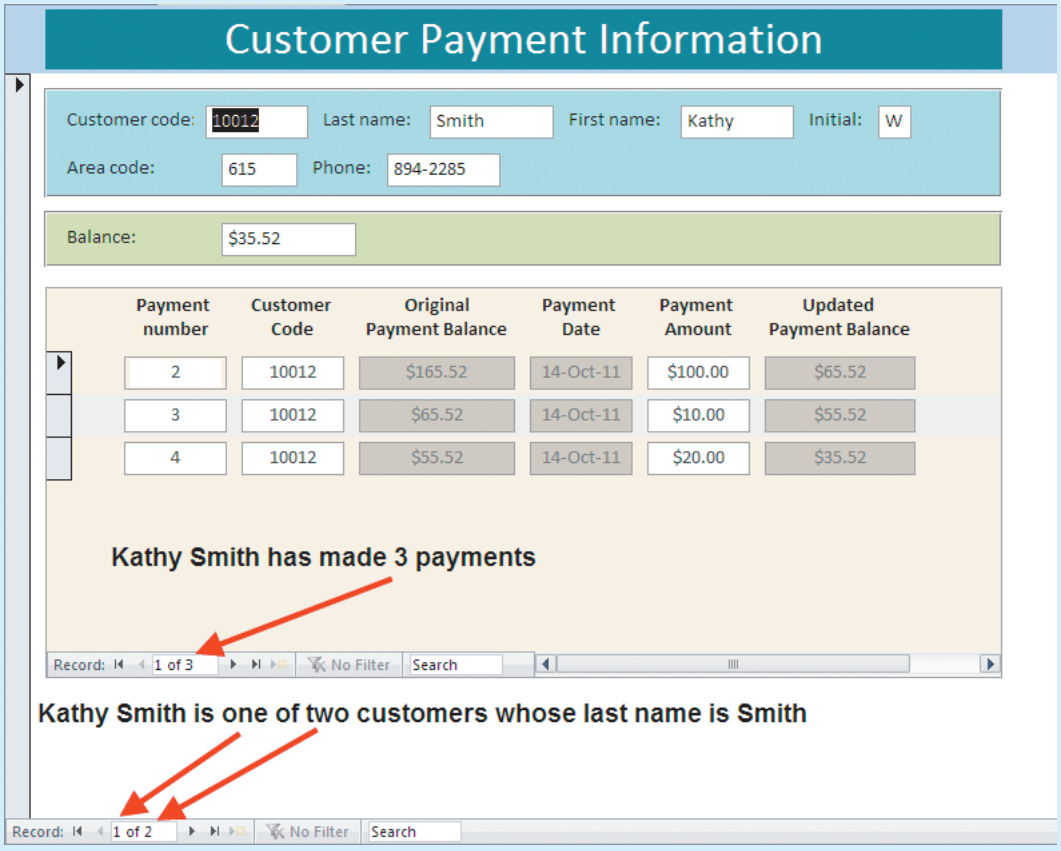
FIGURE M.212 **Customer payments**



SOURCE: Course Technology/Cengage Learning

Go ahead and make a few more payment on account entries for Kathy Smith. A sample set of several payment entries is shown in Figure M.213.

FIGURE M.213 Payment history



SOURCE: Course Technology/Cengage Learning

CONCLUSION

Only a few examples are shown in this tutorial. The objective is not to develop full-blown applications, but to show you some examples of what can be done in the Microsoft Access environment. Once you have seen those examples, you have a foundation on which to build greater expertise. Keep in mind that Access is a superb prototyping tool, but it is not capable of serving the full database and information needs of even medium-sized organizations, let alone large ones. Products such as Microsoft's SQL Server, IBM's DB2, or Oracle are better candidates for such environments. Nevertheless, given its ability to let you develop superb prototypes, Access has earned a place of honor in the ranks of database professionals.