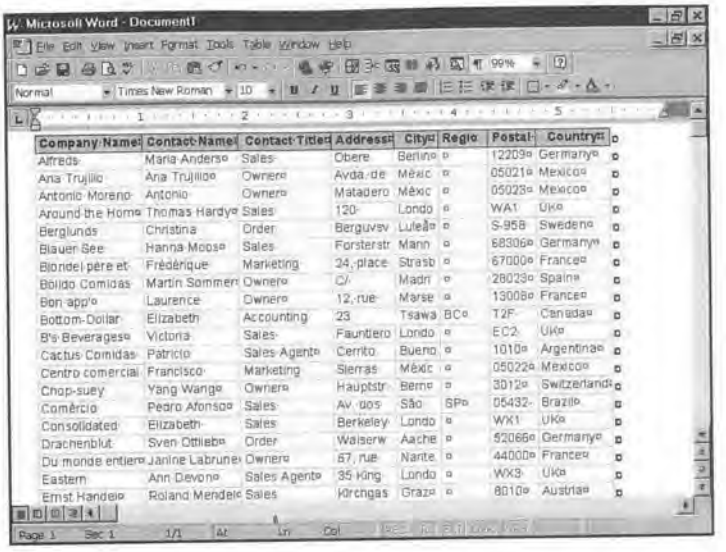


FIG. 7.60 Access data pasted into a Word document as a fully formatted table.

When you copy Access records to the Clipboard, the first line contains field names, no matter what group of records you select. If you append individual records or groups of records to those already pasted to a document in another application, you must manually remove the duplicated field names.

> **Note**
>
> The field names pasted into Word documents contain spaces. Spaces, however, are not allowed in the first (field names) row for merge data documents. Delete the spaces or replace them with underscores so that Word accepts the names. If you do not remove the spaces, you receive an error message when you try to use the document during the Merge operation.

Exporting Data as a Text File

Exporting a table involves a sequence of operations similar to importing a file with the same format. To export a table as a comma-delimited file that you can use as a merge file with a wide variety of word processing applications, complete these steps:





1. Activate the Database window, display the Tables tab, and select the table you want to export.
2. Choose File, Save As/Export. The Save As dialog appears (see Figure 7.61). Select the To an External File or Database option, and then click OK.
3. Access now displays the Save Table *Filename* In dialog (see Figure 7.62). Select Text Files in the Save As Type drop-down list (the title of the dialog changes to Save Table *Filename* As, as shown in Figure 7.62). Use the Save In drop-down list to select the drive and folder in which you want to store the exported file, enter a name for the exported file in the File Name text box, and then click Export.

Oracle Ex. 1009, p. 281
Oracle v. DeBellis
IPR2024-00221



FIG. 7.61 Choosing whether to export a table to an external file.

4. Access starts the Text Export Wizard. Using the Text Export Wizard, including its advanced options, is the same as using the Text Import Wizard described previously, except that the end result is an external file instead of an Access table. (When exporting a text file, the Text Export Wizard doesn't have a step to edit field names, or select field data types; these options aren't relevant when exporting data.)

   To finish exporting the text file, follow the procedures as if you were importing a text file. Figure 7.63 shows the exported Customers table from the Northwind.mdb database displayed by Windows Notepad.

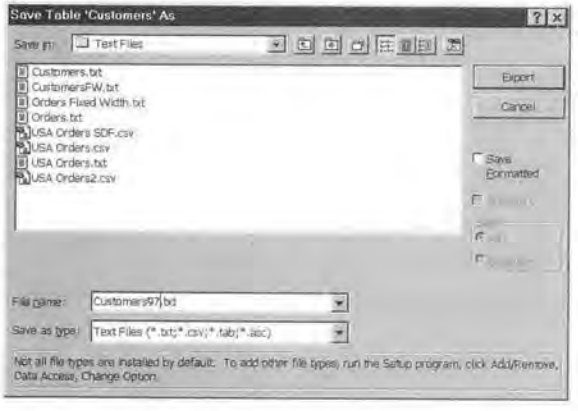


FIG. 7.62 Starting the Text Export Wizard by saving a table as a text file.

Note

The two highlighted lines in Figure 7.63 are a single record from the Access table that was split into two text records during the export process. A newline pair is included in the Address field of the record for Consolidated Holdings. The purpose of the newline pair is to separate a single field into two lines—Berkeley Gardens and 12 Brewery. Use of newline pairs within fields causes many problems with exported files. As mentioned previously in this chapter, use of embedded newline pairs in text fields isn't good database design practice. Use two address fields if you need secondary address lines.

Oracle Ex. 1009, p. 282
Oracle v. DeBellis
IPR2024-00221

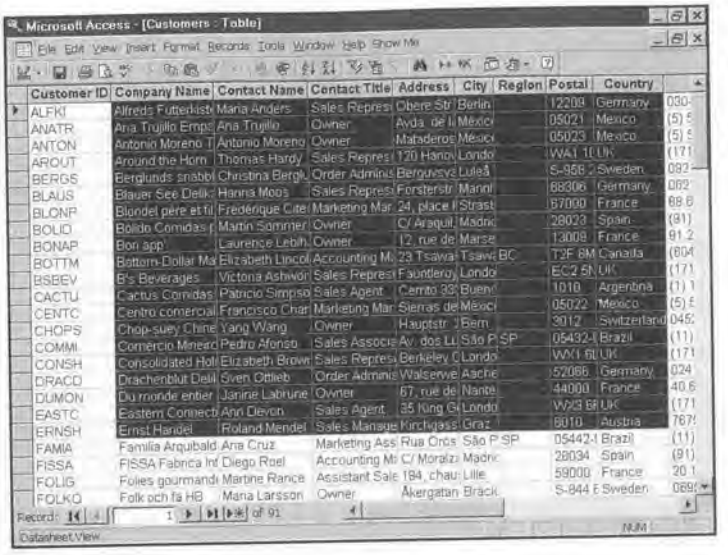


FIG. 7.63 A comma-delimited text file exported from the Customers table.

The records in files created by Access are exported in the order of the primary key. Any other order you may have created is ignored. If you do not assign primary-key fields, the records are exported in the sequence that you entered them into the table.

Exporting Data in Other File Formats

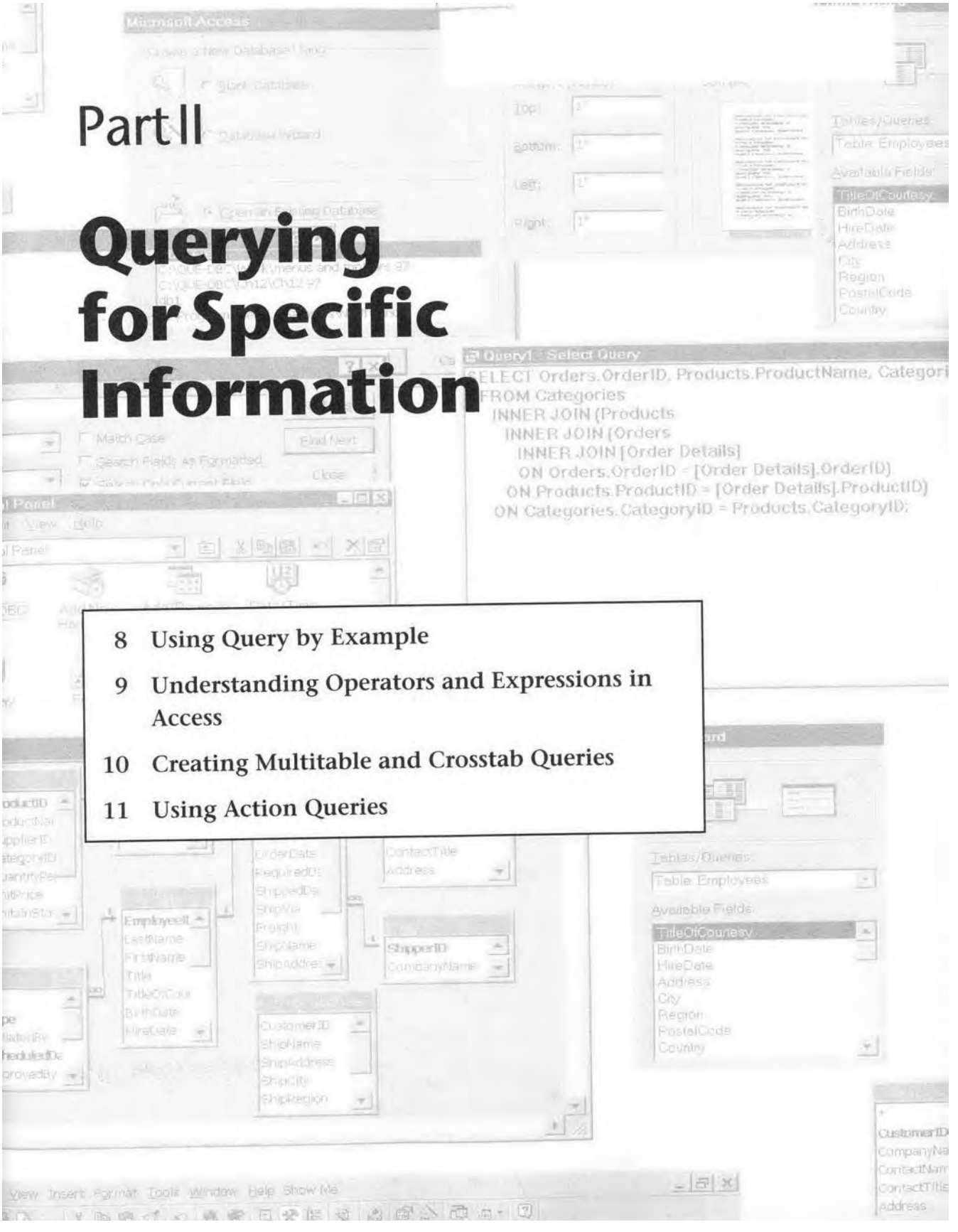
In addition to text files, you can export data to any other file format that Access can import. Access supports export to the following file formats:

- Excel .xls files (versions 3.0 through 97)
- Lotus 1-2-3 and Symphony .WK?
- Rich-text format (.RTF) files for Microsoft Word and other Windows word processing applications
- Paradox for DOS 3.x, 4.x, and 5.0
- FoxPro 2.x and 3.0
- dBASE III/III+, IV, and 5
- Any format supported by an installed ODBC driver

Note

When you export a dBASE file in Access 97, Access creates .ndx files that correspond to the primary key and any additional indexes you may have created in Access. Exporting a Paradox .db file, however, does not create an associated .px file to accompany it. You must use Paradox to re-create any index files required for your application.

Oracle Ex. 1009, p. 283
Oracle v. DeBellis
IPR2024-00221



Part II

Querying for Specific Information

8 Using Query by Example

9 Understanding Operators and Expressions in Access

10 Creating Multitable and Crosstab Queries

11 Using Action Queries

Oracle Ex. 1009, p. 284
Oracle v. DeBellis
IPR2024-00221

Chapter 8

Using Query by Example

Queries are essential tools in any database management system. You use queries to select records, update tables, and add new records to tables. Most often, you use queries to select specific groups of records that meet criteria you specify. You can also use queries to combine information from different tables, providing a unified view of related data items. In this chapter, you learn the basics of creating your own queries. You learn how to specify selection criteria in a query and use the results of your queries.

Introducing Queries

Query by Example, usually abbreviated QBE, was originally developed to enable users of mainframe-computer database applications to find and display pieces of data (or collections of data) without having to know a computer language. Many database management systems eventually came to use QBE in one form or another. (In fact, dBASE, the first commercially successful PC database manager, used a variant of QBE for its dot-prompt commands.)

At a command prompt, for example, QBE users enter such statements as this:

```
LIST ALL lastnames LIKE Lin* WITH state IL IN us_hist
```

These are known as *give me an example of* statements. The QBE application program then searched the `lastnames` field of the `us_hist` table file for all names beginning with `Lin`. The program disregarded the remaining characters in the names and displayed only those records that met the `lastname` and `state (IL)` requirements.

As computer display terminals became more sophisticated, *graphical QBE* developed as the preferred method of creating queries. Graphical QBE displays the field names of one or more database tables as headers of a column. Users can type partial example statements, or *expressions*, in these columns to create a query. Because terminals capable of displaying actual graphics were a rarity when graphical QBE was developed, the standard 80-character by 25-line text mode was used. The term *graphical* is a misnomer, therefore, in today's world

Oracle Ex. 1009, p. 285
Oracle v. DeBellis
IPR2024-00221

of truly graphical user environments like Windows. In the original version of graphical QBE, the preceding example statement might resemble the following display:

```
LASTNAME     FIRSTNAME     ADDRESS          CITY             STATE
Like LIN*                                                    IL
```

Fields corresponding to the columns in which no expressions are entered are not checked for compliance with the query.

After checking the designated fields and finding all matches within the searched file, the application displays those addresses containing matching values. The parts of each address appear in their respective columns, as in the following example:

```
LASTNAME     FIRSTNAME     ADDRESS          CITY             STATE
Lincoln      Abraham       123 Elm St.      Springfield      IL
Lincoln      Mary Todd     123 Elm St.      Springfield      IL
```

Learning to type query expressions in graphical QBE columns proved easier for most computer users than typing QBE expressions at a prompt. The user had to know the syntax of only a few expressions to create relatively complex queries. The use of graphical QBE was one feature that made Ansa Software's original Paradox RDBMS a success in the PC desktop RDBMS market, dominated at the time by dBASE II and dBASE III.

Microsoft Windows introduced a graphical user environment ideally suited to a truly graphical QBE database management system. Microsoft has taken full advantage of the wide range of graphical features incorporated in Windows 95 and Windows NT 4.0, as you see when you create your first true Access query.

Creating Your First Real Query

Chapter 6, "Sorting, Finding, and Filtering Data in Tables," introduced the concepts of creating queries. You applied a sort order to tables and added record selection criteria to filters for data contained in tables. Access's Query Design window, however, gives you greater flexibility than the Filter window because it lets you choose the fields that appear in the query result set. You also can create more complex queries by joining primary and related tables.

This chapter explains the basics of QBE. You learn to create multitable queries in Chapter 10, "Creating Multitable and Crosstab Queries," after you learn the details of how to use operators and create expressions in Chapter 9, "Understanding Operators and Expressions in Access." Expressions often are used as criteria for more complex queries.

To devise a simple query that enables you to customize mailing lists for selected customers of Northwind Traders, for example, follow these steps:

1. Open the Northwind Traders database (filename Northwind.mdb in the \Program Files\Microsoft Office\Office\Samples folder). The Northwind Database window appears.

Oracle Ex. 1009, p. 286
Oracle v. DeBellis
IPR2024-00221



2. Click the arrow adjacent to the New Object button in the toolbar and choose New Query from the drop-down menu; alternatively, click the Query tab in the Database window and then click the New button. The New Query dialog appears (see Figure 8.1). This dialog enables you to select between creating a query yourself (Design view) or using one of the Query Wizards to design specialized queries. Select the Design View item and click OK to display the Query Design window.

   The Show Table dialog is superimposed on the Query Design window, as shown in Figure 8.2. The tabbed lists in the Show Table dialog let you select from all existing tables, all queries, or a combination of all tables and queries. You can base a new query on one or more previously entered tables or queries. (The tables and queries listed in the Show Table list come with Access as samples for the Northwind Traders database.)

> **Note**
>
> If you select a table in the Tables page of the Database window, and then click the New Query toolbar button, Access 97 automatically places the selected table in the Query Design window, without displaying the Add Tables dialog described in step 2 of this procedure.



FIG. 8.1 The New Query dialog, in which you choose how you want to create a new query.



FIG. 8.2 Starting the design of a new query with the Show Table dialog.

Oracle Ex. 1009, p. 287
Oracle v. DeBellis
IPR2024-00221

3. This example uses only tables in the query, so accept the default selection of Tables. Click (or use the down-arrow key to select) Customers in the Show Table list to select the Customers table, and then choose the Add button. Alternatively, double-click Customers to add a table to the query. You can use more than one table in a query by choosing another table name from the list and choosing Add again. This example, however, uses only one table. After selecting the tables that you want to use, choose Close. The Show Table dialog disappears.

The Field list for the Customers table appears at the left in the upper pane of the Query Design window, and a blank Query Design grid appears in the lower pane, as shown in Figure 8.3. The Field list displays all the names of the fields listed in the Customers table.

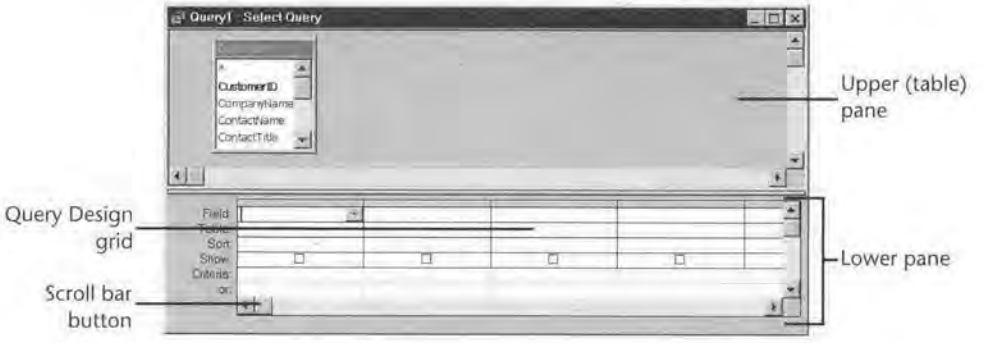


FIG. 8.3 The Field list box of the Customers table in a Query Design window.

By clicking the Query and New buttons in the Database window, you enter Access's Query Design view. By default, you create Select queries that return selected data from tables or the result set of other Select queries. Access assigns a default name, Query1, to the first query that you create (but have not yet saved) in this mode. You assign your own names to queries when you save them. If you create additional queries without saving the first query, Access assigns them the default names Query2, Query3, and so on, in sequence.

Choosing Fields for Your Query

After you choose a table from the Show Table dialog, the next step is to decide which of the table's fields to include in your query. Because you plan to use this query to create a customer mailing list, you must include the fields that make up a personalized mailing address.

As this chapter's introduction explained, the first row of any graphical QBE query contains the names of each field involved in the query. (These field names also are called *field headers*.) The sample query that you are creating, therefore, must include in its first row the names of all the fields that constitute a mailing address.

Oracle Ex. 1009, p. 288
Oracle v. DeBellis
IPR2024-00221

To choose the fields that you want to include in the Query Design grid, follow these steps:

1. When you open the Query Design window, the caret is located in the Field row of the first column. Click the List Box button that appears in the right corner of the first column, or press F4, to open the Field Name list (see Figure 8.4).

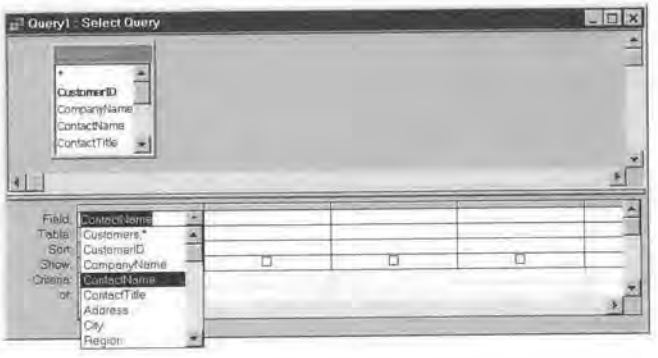


FIG. 8.4 Adding a field name to the Query Design grid.

2. Choose the `ContactName` field as the first field header of the query, or use the down-arrow key to highlight the name and press Enter. The Field list in the lower pane closes.

3. Move the caret to the second column by using the right arrow or Tab key. (Notice that the List Box button moves to the second column along with the caret.) Double- click CompanyName in the Customers Field list in the upper pane to add CompanyName as the second field of your query. Double-clicking entries in the upper pane's list is the second method that Access provides to add fields to a query.

   Access offers a third method of adding fields to your query: the drag-and-drop method. You can add the Address, City, and Region fields to columns 3–5, respectively, in one step by using the drag-and-drop method (see steps 4 and 5).

4. To use the drag-and-drop method of adding fields, you must first select the fields. In the Customers field list of the upper pane's Query Design window, click Address and hold the Shift or Ctrl key as you click City and Region. Alternatively, select Address with the down-arrow key, hold the Shift or Ctrl key, and press the down-arrow key twice more. You have selected the Address, City, and Region fields, as shown in Figure 8.5.

5. Position the mouse pointer over the selected fields and click the left mouse button. Your mouse pointer turns into a symbol representing the three selected field names. Drag the symbol for the three fields to the third column of your query's Field row, as shown in Figure 8.5, and release the left mouse button.

Oracle Ex. 1009, p. 289
Oracle v. DeBellis
IPR2024-00221

Access adds the three fields to your query, in sequence, starting with the column in which you drop the symbol. When the mouse pointer is in an area where you cannot drop the fields, it becomes the international Do Not Enter symbol shown in the upper pane of the Query Design window of Figure 8.5.

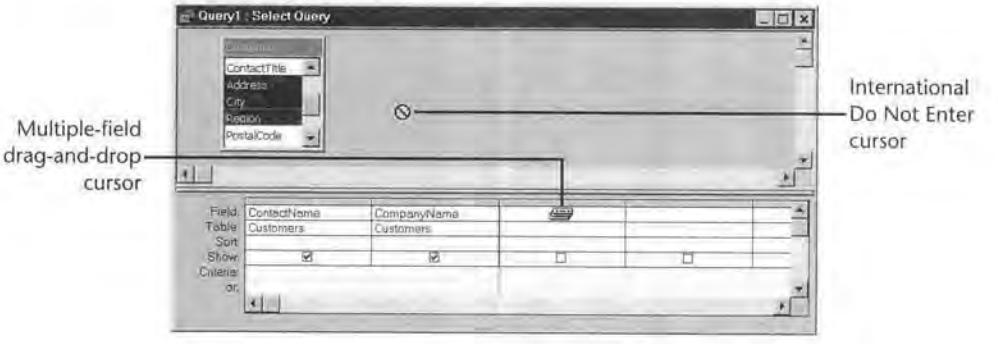
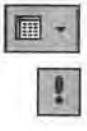


FIG. 8.5 Selecting multiple fields and using the drag-and-drop method to add fields to the Query Design grid.

6. The Query Design grid in the lower pane displays four columns (in the default width) in a normal Query Design window. This query uses seven fields, so you need to drag the edges of the Query Design window to increase the width of the grid's display to expose two additional empty fields. You can reduce the columns' width by dragging the divider of the grid's header bars to the left. Click the scroll-right button (on the horizontal scroll bar at the bottom of the window) twice to display two blank fields, or drag the scroll bar slider button to the right to expose empty fields as necessary.

7. Click the scroll-down button in the Customers field list to display the PostalCode and Country fields. Hold the Shift or Ctrl key and select PostalCode and Country. Drag the symbol for these two fields to the first empty field cell (column 6), and drop the two fields there. Your Query Design window appears similar to the one shown in Figure 8.6. (Notice that the check boxes in the Show row for the columns that contain a field name now are marked.)

> **Note**
>
> Most of the figures in this book are created with windows in *Normal* style, which occupies a portion of the display, rather than *Maximized* style, which occupies all the display. Normal style is used so that figures can be reproduced on a larger scale, which improves legibility.



8. Click the Query View button on the toolbar to enter Run mode. Expect a brief waiting period while Access processes your query on the Customers table. Alternatively, click the Run button on the toolbar to run your query against the Customers table.

Oracle Ex. 1009, p. 290
Oracle v. DeBellis
IPR2024-00221

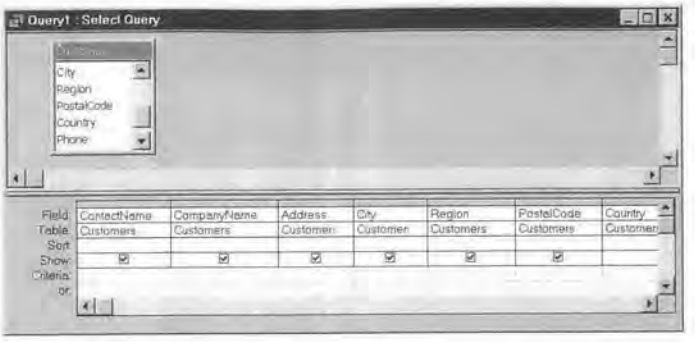


FIG. 8.6 The seven field names included in the new query.

Because you have not yet entered any selection criteria in the Criteria row of the Query Design grid, your query results in the Customers table displaying all records. These records appear in the order of the primary key index on the CustomerID field because you have not specified a sorting order in the Sort row of the Query Design grid. (The values in the CustomerID field are alphabetic codes derived from the Company Name field.) Figure 8.7 shows the result of your first query.

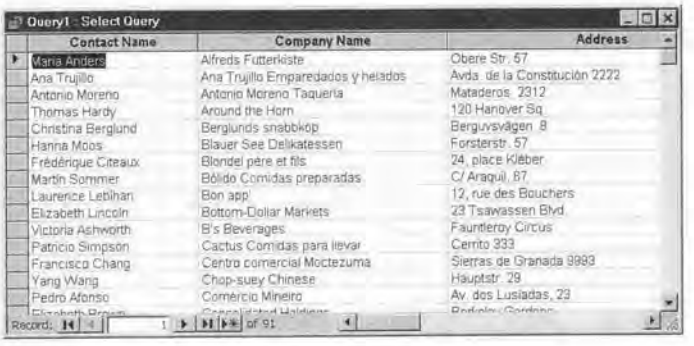


FIG. 8.7 A list of all records contained in the Customers table.

> **Note**
>
> Many field names of tables in Access 2.0's NWIND.MDB sample database contained spaces. Most RDBMSs don't permit spaces in field names or table names. Starting with Access 95, field names in Northwind.mdb no longer include spaces. Access 97's Northwind.mdb uses the Caption property of table fields to alter (alias) the field headers of tables to add spaces where appropriate. For example, to alias the Customers table's CompanyName field, you set the Caption property's value to **Company Name**.

Selecting Records by Criteria and Sorting the Display

The mailing for which you are creating a list with your sample query is to be sent to U.S. customers only, so you want to include in your query only those records that have USA

Oracle Ex. 1009, p. 291
Oracle v. DeBellis
IPR2024-00221

in their Country fields. Selecting records based on the values of fields—that is, establishing the criteria for the records to be returned (displayed) by the query—is the very heart of the query process.

◀◀ See "Working with Relations, Key Fields, and Indexes," p. 146

Perform the following steps to establish criteria for selecting the records to comprise your mailing list:

Design

1. Click the Design View button on the toolbar to return to Design mode. The partially filled Query Design grid replaces the mailing list on-screen.
2. To restrict the result of your query to firms in the United States, type the expression **USA** in the Criteria row of the Country column. Entering a criterion's value without preceding the value with an operator indicates that the value of the field must match the value of the expression USA. You do not need to add quotation marks to the expression; Access adds them for you (see the Country column in Figure 8.8).

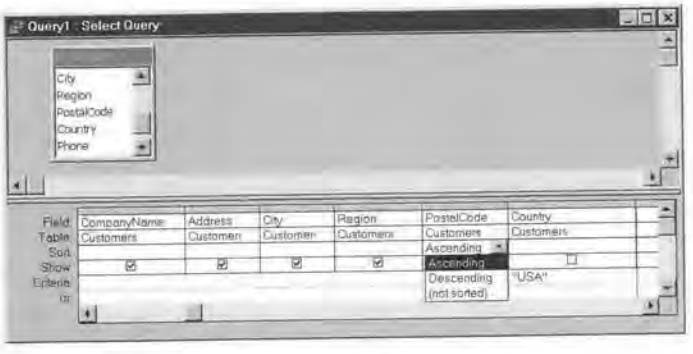


FIG. 8.8 Adding the Ascending sort order to the PostalCode field.

3. Click the Show check box in the Country column to remove the check mark that appeared when you named the column. After you deactivate the Show check box, the Country field does not appear when you run your query. (You do not need to include the Country column in your mailing list address if you are mailing from within the United States to another location within the United States.) If you do not deactivate a Show check box, that field in the query appears in the query's result by default.
4. Move the caret to the PostalCode column's Sort row and press F4 to display the sorting options for that field: Ascending, Descending, and Not Sorted. Choose the Ascending option to sort the query by PostalCode from low codes to high.

   At this point, the Query Design grid appears as shown in Figure 8.8.

Oracle Ex. 1009, p. 292
Oracle v. DeBellis
IPR2024-00221

5. Click the Query View or Run button on the toolbar to display the result of your criterion and sorting order. Use the horizontal scroll bar to display additional fields.

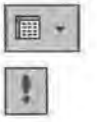


Figure 8.9 displays the query result table (also called a *query result set*) that Access refers to as an updatable `Recordset` (a `Recordset` object of the `Dynaset` type, to be more precise). A `Recordset` object is a temporary table stored in your computer's memory; it is not a permanent component of the database file. A `Recordset` object differs from the conventional view of an object created by the SQL `VIEW` reserved word because you can update the data in a `Recordset` object of the `Dynaset` type. After you save the query, the Northwind.mdb file saves only the design specifications of the query, not the values that the query contains. The query design specification is called a `QueryDef` object.



Query1 : Select Query

Contact Name	Company Name	Address	City	Region	Postal Code
Lu Wong	The Cracker Box	55 Grizzly Peak Rd	Butte	MT	59801
Art Braunschweiger	Split Rail Beer & Ale	P.O. Box 555	Lander	WY	82520
Jose Pavarotti	Save-a-lot Markets	187 Suffolk Ln.	Boise	ID	83720
Paula Wilson	Rattlesnake Canyon Grocery	2817 Milton Dr.	Albuquerque	NM	87110
Jaime Yorres	Let's Stop N Shop	87 Polk St.	San Francisco	CA	94117
Liz Nixon	The Big Cheese	89 Jefferson Way	Portland	OR	97201
Fran Wilson	Lonesome Pine Restaurant	89 Chiaroscuro Rd.	Portland	OR	97219
Howard Snyder	Great Lakes Food Market	2732 Baker Blvd.	Eugene	OR	97403
Yoshi Latimer	Hungry Coyote Import Store	City Center Plaza	Elgin	OR	97827
Helvetius Nagy	Trail's Head Gourmet Provisioners	722 DaVinci Blvd.	Kirkland	WA	98034
Karl Jablonski	White Clover Markets	305 - 14th Ave. S.	Seattle	WA	98128
John Steel	Lazy K Kountry Store	12 Orchestra Terrace	Walla Walla	WA	99362
Rene Phillips	Old World Delicatessen	2743 Bering St.	Anchorage	AK	99508

Record: 14 1 of 13

FIG. 8.9 The query result, sorted in ascending order by PostalCode.

Creating More Complex Queries

To limit your mailing to customers in a particular state or group of states, you can add a Criteria expression to the Region or PostalCode field. To restrict the mailing to customers in California, Oregon, and Washington, for example, you can specify that the value of the PostalCode field must be equal to or greater than 90000. Alternatively, you can specify that Region values must be CA, OR, and WA.

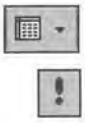
Follow these steps to restrict your mailing to customers in California, Oregon, and Washington:

1. Click the Design View button on the toolbar to return to Design mode.



2. Use the right-arrow or Tab key to move the caret to the Region column. If the Region column is not on-screen, click the scroll-right button until that column appears.
3. Type **CA** in the first criteria row of the Region column. Access adds the quotation marks around CA (as it did when you restricted your mailing to U.S. locations with the USA criterion).
4. Press the down-arrow key to move to the next criteria row in the Region column. Type **OR**, and then move to the third criteria row and type **WA**. Your query design now appears as shown in Figure 8.10. Access adds the required quotation marks to these criteria, also.

Oracle Ex. 1009, p. 293
Oracle v. DeBellis
IPR2024-00221



5. Click the Datasheet View or Run button on the toolbar to return to Run mode. The query result set appears as shown in Figure 8.11.

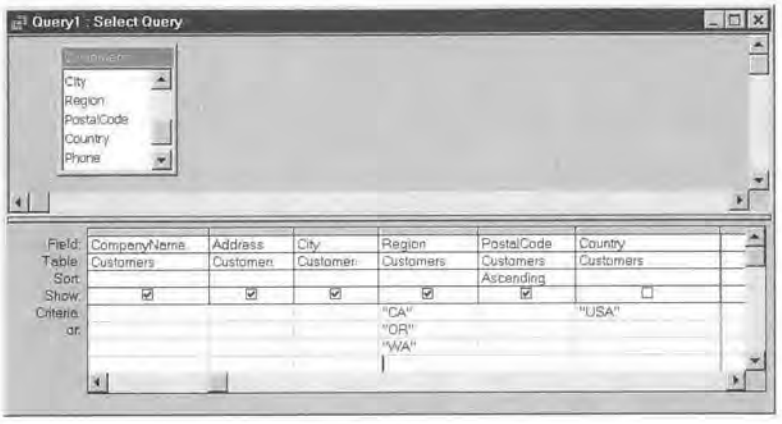


FIG. 8.10 Adding criteria to the Region field of the Query Design grid.

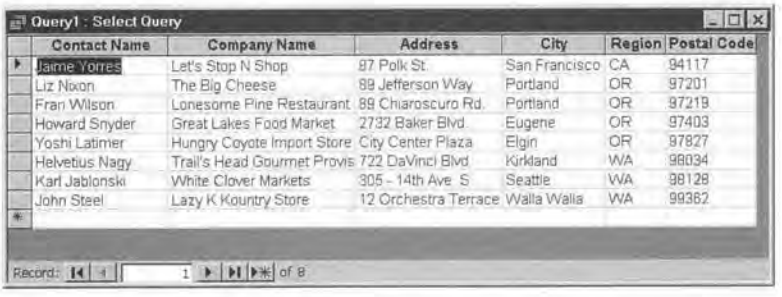


FIG. 8.11 The query result set for customers in California, Oregon, and Washington.

Note

After you type a criterion on the same line as a previously entered criterion in another field, only those records that meet both criteria are selected for display. In the preceding example, therefore, only those records with Region values equal to CA and Country values equal to USA are displayed.

To be displayed, records for Region values OR and WA need not have Country values equal to USA because the USA criterion is missing from the OR and WA rows. This omission does not really affect the selection of records in this case because all OR and WA records are also USA records. To eliminate possible ambiguity, however, USA should appear in each criterion row that contains a state code.

(Note that the remaining criteria rows in the different columns on the Query Design grid enable you to enter additional criteria to qualify further which records to display. In the current example, you need no additional criteria, so Access leaves these cells blank.)

Oracle Ex. 1009, p. 294
Oracle v. DeBellis
IPR2024-00221

Editing Table Data in Query View

You can edit the data in any visible fields of the table in the query display. Any table that underlies the `Recordset` reflects the changes that you make to data cells in an updatable `Recordset` object of the `Dynaset` type. To edit an entry in a query and then verify the change to the corresponding record in the underlying table, perform these steps:

1. Use the right-arrow or Tab key to move the caret to the first row of the Company Name column (Let's Stop N Shop).
2. Press F2 to deselect the field and to enter Edit mode.
3. Use the arrow keys to position the cursor to immediately before the `N` and add an apostrophe to change the Company Name value to **Let's Stop 'N Shop**.
4. Press Enter or move the caret down another line to make the change to Let's Stop 'N Shop permanent. (Data in the underlying table does not change until you press Enter or move to a different record.)

> **Note**
>
> Changes made to sorted fields in Access do not actually move the edited records to their correct locations in the query tables until you press Shift+F9 to rerun the query.

5. Click the Database Window button of the toolbar, or choose Window, 1 Database and then click the Tables tab of the Database window.



6. Double-click the Customers table to display the table in Datasheet view. Scroll down to verify that the underlying Customers table reflects your update to the query data (see Figure 8.12).

Customers : Table

Customer ID	Company Name	Contact Name	Co
LAMAI	La maison d'Asie	Annette Roulet	Sales Manage
LAUGB	Laughing Bacchus Wine Cellars	Yoshi Tannamuri	Marketing Ass
LAZYK	Lazy K Kountry Store	John Steel	Marketing Mar
LEHMS	Lehmanns Marktstand	Renate Messner	Sales Repres
LETSS	Let's Stop 'N Shop	Jaime Yorres	Owner
LILAS	LILA-Supermercado	Carlos González	Accounting M
LINOD	LINO-Delicateses	Felipe Izquierdo	Owner
LONEP	Lonesome Pine Restaurant	Fran Wilson	Sales Manage
MAGAA	Magazzini Alimentari Riuniti	Giovanni Rovelli	Marketing Mar
MAISD	Maison Dewey	Catherine Dewey	Sales Agent
MEREP	Mère Paillarde	Jean Fresnière	Marketing Ass
MORGK	Morgenstern Gesundkost	Alexander Feuer	Marketing Ass
NORTS	North/South	Simon Crowther	Sales Associ
OCEAN	Océano Atlántico Ltda.	Yvonne Moncada	Sales Agent

Record: 45 of 91

FIG. 8.12 Viewing the original table data confirms the alteration made to a record in the query results window.

Changing the Names of Query Column Headers

You can substitute a query's field header names with column header names of your choice—a process called *aliasing*—but only if the header name has not been changed by an entry in the field Caption property of the table. If you are a U.S. firm, for example,

Oracle Ex. 1009, p. 295
Oracle v. DeBellis
IPR2024-00221

you might want to change Region to **State** and PostalCode to **ZIP**. (Canadian firms might want to change only Region to **Province**.)

As demonstrated in the following example, you cannot make the change to the PostalCode field for queries based on the Customers table because the PostalCode field previously has been changed (aliased) to Postal Code by the Caption property for the field. You can, however, make the change to the Region field because this field is not aliased at the table level.

Tip

If you already have a main document for the merge operation, substitute the main merge document's merge field names for the table's field header names in your query.

Note

The inability to alias field names in queries that have been altered by use of the `Caption` property in the source table is a good reason not to use the `Caption` property of table fields. If you want to display different field headers, use a query for this purpose. In a client/server RDBMS, such a query is called an SQL `VIEW`. Aliasing field names in tables, rather than queries, is not considered a generally accepted database design practice.

To attempt to change the query column header names, perform the following steps:



1. Switch to Design mode by clicking the toolbar's Design View button. Then use the right-arrow or Tab key to move the caret to the Field column containing the field header name that you want to change—in this case, the Region column.
2. Press F2 to deselect the field; then press Home to move the caret to the first character position.
3. Type the new name for the column and follow the name with a colon (with no spaces):

   State:

   The colon separates the new column name that you type from the existing table field name, which shifts to the right to make room for your addition. The result, in this example, is `State:Region` (see Figure 8.13).
4. Use the arrow key to move to the PostalCode field and repeat steps 2 and 3, typing **ZIP:** as that header's new name. The result, as shown in Figure 8.13, is `ZIP:PostalCode`.
5. Change the column header for the ContactName field to **Contact**; change the column header for the CompanyName field to **Company** (refer to steps 2, 3, and 4).
6. Delete the CA, OR, and WA criteria from the State:Region column so that all records for the U.S. appear.

Oracle Ex. 1009, p. 296
Oracle v. DeBellis
IPR2024-00221

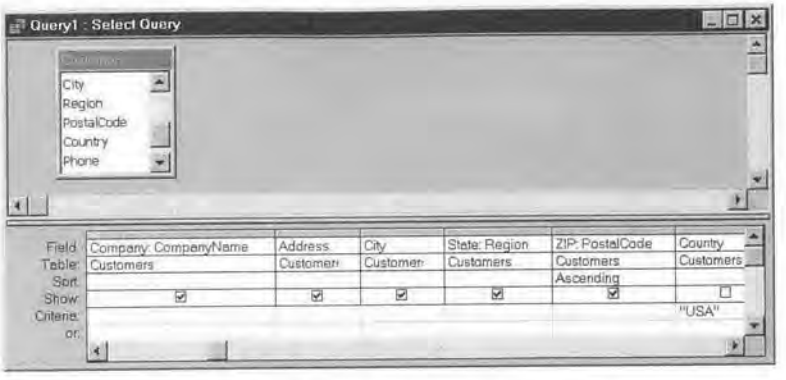


FIG. 8.13 Changing the names of the Region and PostalCode column headers.

7. Click the toolbar's Query View or Run button to execute the query. Observe that only the Region column header is changed to State; the other columns are unaffected by the alias entry (see Figure 8.14).



FIG. 8.14 The query result set, with only one new column header.

Note

To make field aliasing in queries operable, delete the entries in the Caption field for each of the aliased fields of the table. In the sections that follow, the entries in the Caption property of the ContactName, CompanyName, and PostalCode fields of the Customers table have been deleted. Deleting these entries makes the aliases you entered in the preceding example work as expected.

Printing Your Query as a Report

Queries are often used to print quick, *ad hoc* reports. Access 97 lets you print your report to a Word for Windows .rtf (rich-text format) file, Excel worksheet .xls file, or a DOS .txt (text) file, or as an attachment to an Exchange message.

Oracle Ex. 1009, p. 297
Oracle v. DeBellis
IPR2024-00221



▶▶ See "Sending Reports by Microsoft Exchange and Outlook," p. 570

Previewing your query table's appearance to see how the table will appear when printed is usually a good idea. After you determine from the preview that everything in the table is correct, you can print the finished query result set in a variety of formats.

To preview a query result set before printing it, follow these steps:



1. In Query view, click the Print Preview button on the toolbar. A miniature version of the query table appears in Report Preview mode.



2. Position the Zoom mouse pointer (the magnifying glass cursor) at the upper-left corner of the miniature table and click the left mouse button or the Zoom button above the window to view the report at approximately the scale at which it will print.

3. Use the vertical and horizontal scroll bar buttons to position the preview in the window (see Figure 8.15).

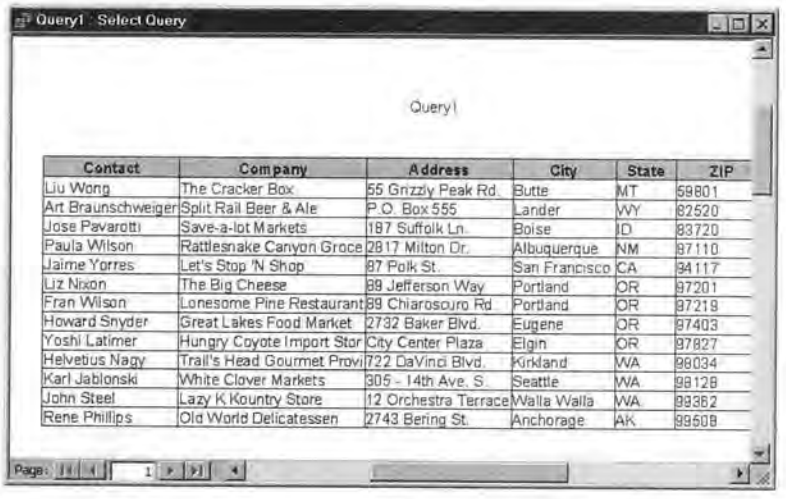


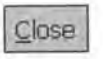
FIG. 8.15 Previewing the Mailing List query table at zoomed scale.

Note

Field width in the query table is based on the column width that you last established in Run mode. You might have to drag the right edge of the field header buttons to the right to increase the columns' width so that the printed report does not truncate the data. If the query data's width exceeds the available printing width (the paper width minus the width of the left and right margins), Access prints two or more sheets for each page of the report.

Oracle Ex. 1009, p. 298
Oracle v. DeBellis
IPR2024-00221

4. Click the Close button and then alter the width of the columns in Run mode so that all seven columns of the query fit on one sheet of paper when you print it. Return to Report Preview mode after you finish this process.



Note

Adjust the displayed width of query fields the same way that you adjusted table fields (as described in Chapters 4, "Working with Access Databases and Tables," and 6, "Sorting, Finding, and Filtering Data in Tables"). Access stores query column widths as properties of the query.

To print to the default printer a query table after previewing it and determining that all the data is correct, simply click the Print button of the toolbar. To change the default printing margins of one inch on all sides of the sheet (or any revised defaults that you might have set earlier by choosing Tools, Options and entering new values in the Print Options text boxes of the General page), or to print only data (but not field header names), follow these steps:



1. Choose File, Page Setup to open the Page Setup dialog shown in Figure 8.16. If necessary, click the Margins tab to display the Margins page.

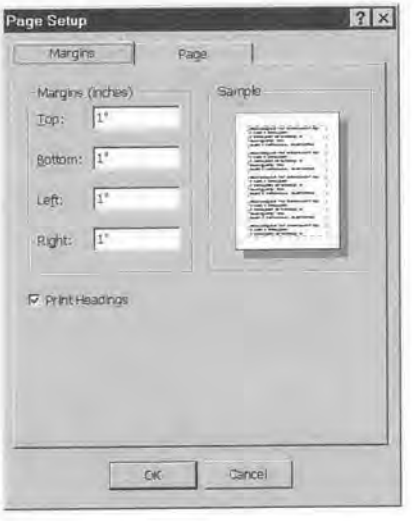


FIG. 8.16 The Margins page of the Page Setup dialog.

2. Enter any changes that you want to make to the margins; mark the Print Headings check box if you want to print the field header names. Click the Page tab to change the print orientation, paper size or source, or printer. Then click OK to return to Print Preview mode.

Oracle Ex. 1009, p. 299
Oracle v. DeBellis
IPR2024-00221



3. Click the Print button of the toolbar to print your query data.

Using the Data from Your Query

Occasionally, you might want to use data from your query as part of a different Windows application without printing the data in a report. The simplest technique for transferring data in your query to another Windows application is to use the Clipboard. Clipboard operations for data in query tables are identical to those operations for tables described in Chapter 7, "Linking, Importing, and Exporting Tables."



◀◀ See "Using the Clipboard to Import Data," p. 245



Access 97 includes a Merge It choice when you click the arrow adjacent to the Office Links button on the toolbar. Merge It automatically creates a Word mail merge document. Performing automated operations manually, however, gives you insight into how Access 97 implements Office Links. To create manually a merge data file for Word directly from a query, for example, follow these steps:

1. In Query view, choose Edit, Select All Records.

Tip

For a partial mailing, such as to firms in California only, you can choose the individual records by dragging the mouse over the records' selection buttons.

2. Press Ctrl+C or choose Edit, Copy to copy the selected records to the Clipboard.
3. Run Word and, if necessary, choose File, New to open a new window.
4. Press Ctrl+V or choose Edit, Paste. Your Access mailing records are added as a table to the Word document (see Figure 8.17).
5. Convert the table to the tab-delimited format necessary for Word merge data files. Place the caret in one of the cells of the table and choose Table, Select Table to select all the rows and columns of the table.
6. Choose Table, Convert Table to Text to display the dialog that lets you choose the field-delimiter character. Accept the default Tab selection in the Convert Table to Text dialog and click OK. Your merge data file appears as shown in Figure 8.18.

Note

Some of the records of this query also contain *newline pairs* (paragraph marks) embedded in fields that result in premature ends of these records. An example of a spurious newline pair in the Address field is highlighted for the Let's Stop 'N Shop record in Figure 8.18. Replace the extra new line pairs with commas and spaces, and delete the quotation marks that enclose fields containing extra headline pairs before using the document in a merge operation.

Oracle Ex. 1009, p. 300
Oracle v. DeBellis
IPR2024-00221

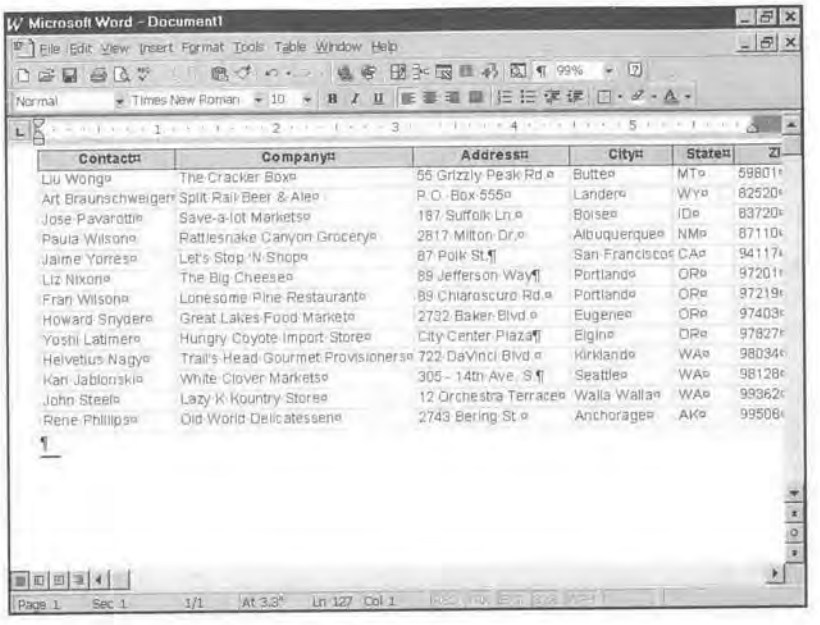


FIG. 8.17 The query result set copied to a Word for Windows document.

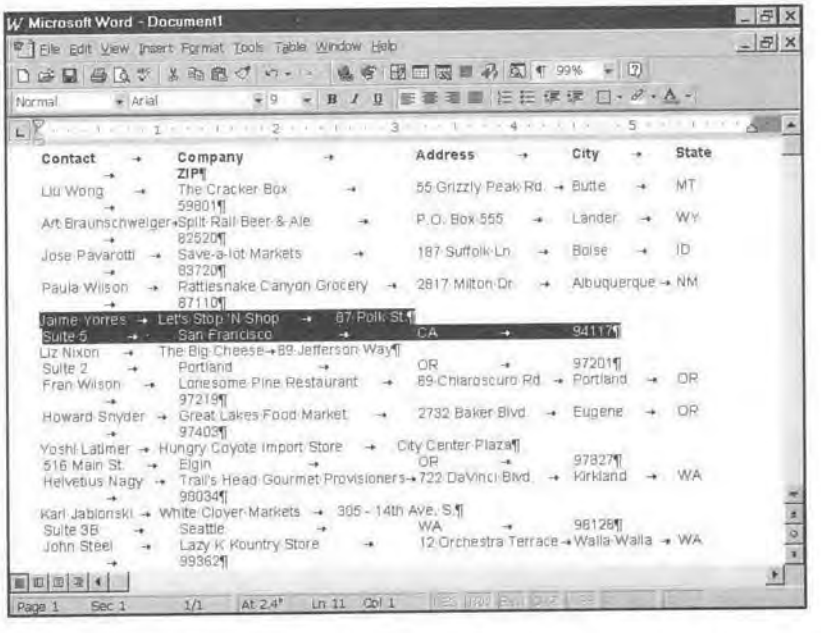


FIG. 8.18 A Microsoft Word merge data file with extra newline pairs embedded.

Oracle Ex. 1009, p. 301
Oracle v. DeBellis
IPR2024-00221

If you use a DOS word processor and want to import query data into a merge data file, save the query in the form of a table and export the table as a file in a format compatible with that of your word processor. This process requires that you create an action query to create a table. Then export the table's data in a compatible format, as described in Chapter 7, "Linking, Importing, and Exporting Tables." Action queries are discussed briefly in the section "Creating and Using a Simple Make-Table Action Query," later in this chapter, and in Chapter 11, "Using Action Queries."

Saving and Naming Your Query

After completing your query, save it as an element of your database file, giving the query its own descriptive name.

Follow these steps to save and name your query:

1. Close your query by clicking the Close Window button. Access prompts you with a message box to save the new or modified query, as shown in Figure 8.19.



FIG. 8.19 The message reminding you to save a newly created or modified query.

2. Click the message box's Yes button to save your query. Because Access has already assigned a default name to your query, the Save As dialog appears (see Figure 8.20). If you click Cancel or press Esc, Access does not save your query.



FIG. 8.20 The Save As dialog, in which you enter the name for your query.

3. Type in the Query Name text box a descriptive name for your query, in this case **qryUSMailList**, and then press Enter or click OK. Your query now is saved under the name that you assigned it rather than under the default name.

Oracle Ex. 1009, p. 302
Oracle v. DeBellis
IPR2024-00221

Note

Prefixes, called tags, are commonly used to identify types of Access objects you create. Tags are one of the subjects of Appendix B, "Naming Conventions for Access Objects and Variables."

Alternatively, you can save your query by choosing File, Save As.

To rename a saved query, follow these steps:

1. Close the query by clicking the close window box. (You cannot rename an open query.)
2. In the Queries page of the Database window, select the query that you want to rename.



3. Choose Edit, Rename. Alternatively, right-click the query you want to rename and choose Rename from the popup menu.
4. Type in the text box a new name to replace the caption adjacent to the query's icon.
5. Press Enter to accept the change. Access saves your query with the new name that you assigned.

Creating Other Types of Queries

Access enables you to create the following four basic types of queries to achieve different objectives:

- *Select*. Select queries extract data from one or more tables and display the data in tabular form.
- *Crosstab*. Crosstab queries summarize data from one or more tables in the form of a spreadsheet. Such queries are useful for analyzing data and creating graphs or charts based on the sum of the numeric field values of many records.
- *Action*. Action queries create new database tables from query tables or make major alterations to a table. Such queries enable you to add or delete records from a table or to make changes to records based on expressions that you enter in a query design.
- *Parameter*. Parameter queries repeatedly use a query and make only simple changes to its criteria. The mailing list query that you created earlier in this chapter is an excellent candidate for a parameter query because you can change the criterion of the Region field for mailings to different groups of customers. When you run a parameter query, Access displays a dialog to prompt you for the new criterion. Parameter queries are not actually a separate query type because you can add the parameter function to select, crosstab, and action queries.

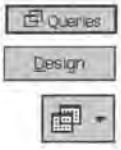
Chapters 10, "Creating Multitable and Crosstab Queries," and 11, "Using Action Queries," explain how to create each of the four query types. Creating a table from the mail-

Oracle Ex. 1009, p. 303
Oracle v. DeBellis
IPR2024-00221

ing list query to export to a mail merge file is an example of an action query. (In fact, this is the simplest example of an action query and also the safest because make-table queries do not modify data in existing tables.)

Creating and Using a Simple Make-Table Action Query

To create a table from your mailing list query, you first must convert the query from a select to an action query. Follow these steps to make this change:



1. Open your mailing list query in Design mode by selecting the name that you gave the query in the Database window and clicking the Design button.
2. Choose Query, Make-Table Query. (You can access the Query menu only in Query Design mode.) Alternatively, click the Query Type button on the toolbar and select Make Query Table. The Make Table dialog appears, as shown in Figure 8.21.



FIG. 8.21 The Make Table dialog, in which you enter a name for the table the query creates and specify the new table's location.

3. In the Table Name text box, type a descriptive table name for your query table, such as **tblUSMailList**.

   The Make Table dialog enables you to define your query table's properties further in several ways. You can add the table to the Northwind Traders database by choosing the Current Database option (the default). You also can choose the Another Database option to add the table to a different database that you specify in the File Name text box.

4. Click OK. Access converts your select query to the make-table type of action query.
5. Close your query by clicking the Close Window button. Access displays a message dialog asking if you want to save changes to your query; click Yes. Your query name in the Database window now is prefixed by an exclamation point, as shown in Figure 8.22. An exclamation point indicates that the query is an action query.

Now that you have converted your query from a select query to an action query, you can create a new U.S. mailing list table. To create the table, follow these steps:

1. Run the newly converted action query table to create your mailing list by double-clicking its name in the Queries page of the Database window (refer to Figure 8.22).

   When you open an action query table, it performs the desired action—in this case, creating the tblUSMailList table—instead of simply displaying a select query result set. Before Access carries out the action, however, a message appears (see Figure

Oracle Ex. 1009, p. 304
Oracle v. DeBellis
IPR2024-00221

8.23) and warns you that the query will modify the data in the tblUSMailList table (despite the fact that you haven't yet created the table).



FIG. 8.22 A highlighted action query in the list of queries.



FIG. 8.23 The message box that appears after you open an action query.

2. Click Yes to dismiss the message box and continue the operation. A second message, shown in Figure 8.24, appears to tell you what happens after you execute the action query.



FIG. 8.24 A second message indicating the effects of running the action query.

3. Click Yes. Because you have not run this action query before, running it now creates the new tblUSMailList table.

Oracle Ex. 1009, p. 305
Oracle v. DeBellis
IPR2024-00221



4. Click the Table tab in the Database window. Access adds the new tblUSMailList table to the list of tables in the Northwind database, as shown in Figure 8.25.



FIG. 8.25 The new table added to the Northwind Database's table list.

5. Double-click the tblUSMailList icon to open the table. Its contents, which are identical to the contents of the Datasheet view of the make-table query, appear as shown in Figure 8.26.



FIG. 8.26 The Datasheet view of the table created by the make-table query.

After you create the new table, you can export its data to any of the other file formats supported by Access. To do so, use the methods described in Chapter 7, "Linking, Importing, and Exporting Tables."

Adding a Parameter to Your Make-Table Query

A simple modification to your mailing list query enables you to enter a selection criterion, called a *parameter*, from a prompt created by Access. Follow these steps:



◀◀ See "Setting Default Options," p. 81

Oracle Ex. 1009, p. 306
Oracle v. DeBellis
IPR2024-00221

1. Close the tblUSMailList table by clicking the Close Window button; then click the Queries tab in the Database window.

Queries

2. Choose the qryUSMailList query that you created earlier in the chapter, and then choose the Design button to display your make-table action query in Design mode.

Design

3. Type **[Enter the state code:]** in the first criteria row of the State:Region column, as shown in Figure 8.27. The enclosing square brackets indicate that the entry is a prompt for a parameter when you run the action query.

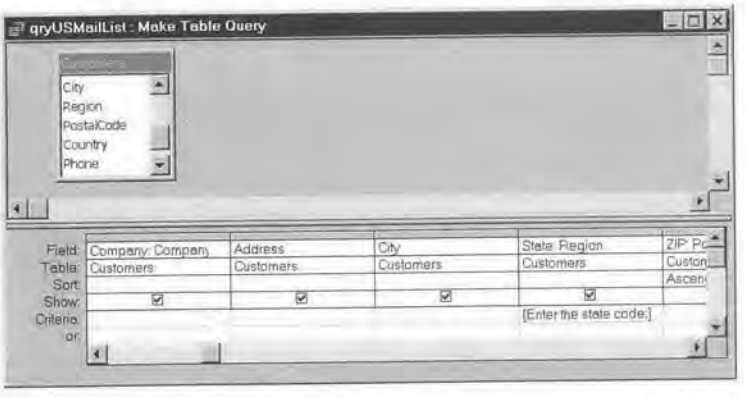


FIG. 8.27 Entering a parameter prompt as a criterion in Query Design view.

4. Close and save changes to the action query, select the qryUSMailList query, and choose File, Edit, Rename.

5. Rename your query by typing **qryStateMailList** in the text box and pressing Enter.

6. Select the renamed qryStateMailList query from the list in the Database window, and then click the Open button. Alternatively, double-click the qryStateMailLis icon. You see the message indicating that data will be modified (refer to Figur 8.23)

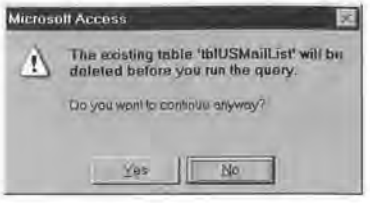
Open

7 Choose Yes. Another message appears, warning that you are about to overwrite th data in the table created by the last execution of your query (see Figure 8.28)

Note

Each time you execute a make-table query, it creates a new table with whatever name you have specified in the Make Table dialog. If a table with that name already exists, its contents are lost and replaced by the result set of the make-table query. If you want a make-table query to create a table with a different name, you must choose Query, MakeTtable Query,

Oracle Ex. 1009, p. 307
Oracle v. DeBellis
IPR2024-00221



and enter a new name in the Make Table dialog.

FIG. 8.28 The message warning that you are about to overwrite data.

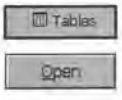


8. Choose Yes. Access displays the Enter Parameter Value dialog. This dialog contains a prompt for you to enter the state criterion, as shown in Figure 8.29.

FIG. 8.29 The text box for entering a parameter to be used as a select criterion.

9. Type **WA**, and press Enter or choose OK. (You do not need to type an equal sign before the state code because Access enters the equal sign for you.)

   Another message appears, similar to the one in Figure 8.24, indicating the number of records in the new version of the tblUSMailList table that have a value in the Regions field that matches your state parameter entry.



10. Choose Yes to close the message box and execute the make-table query.

11. Click the Tables tab in the Database window, and select tblUSMailList. Choose the Open button. Records only for customers in Washington appear in the table.

You can delete the new table from the Northwind Traders database by closing the table, selecting the tblUSMailList table in the Database window, and then pressing the Delete key. (Access requests that you confirm your deletion. Click OK, and Access removes the table from the database.)

Translating Graphical QBE to Structured Query Language

Structured Query Language, or *SQL*, is a standard set of English words used to describe a query. Access translates the QBE expressions that you type in the Query Design grid into a series of statements in its own dialect of SQL, called *Jet SQL* in this book. Access then carries out these instructions on any tables that contain fields matching those specified in your query.

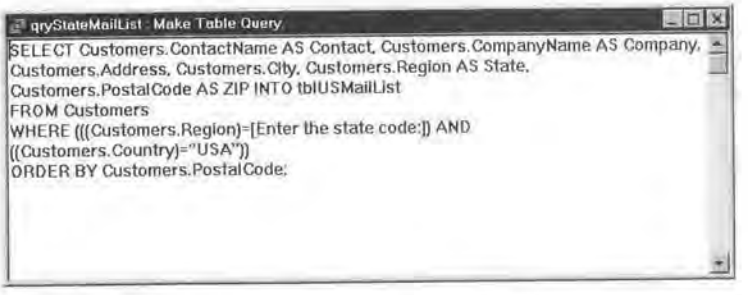
Oracle Ex. 1009, p. 308
Oracle v. DeBellis
IPR2024-00221

Access's use of SQL is important when you are dealing with client/server databases that process SQL statements on the server's computer. After processing the query, the server sends the data for the query result table to your Access client application for further processing.

To display the SQL statements created from your mailing list query, perform the following steps:



1. Open the mailing list query in Design mode by selecting the name you gave the query in the Database window and clicking the Design button.
2. Click the Query View button of the toolbar and choose SQL from the menu, or



choose View, SQL View. The SQL window, which acts as a multiline text box, appears, as shown in Figure 8.30.

FIG. 8.30 Graphical QBE expressions translated by Access into Jet SQL.

SQL *reserved words* are displayed in uppercase letters. SQL reserved words are the instructions or the actions that the query is to perform. The names of the objects in your query appear in uppercase and lowercase letters. The following list explains the meanings of the reserved words used in your query:

- `SELECT` is usually the first reserved word in a SQL statement that returns records to a query result set. The expressions that follow specify the fields involved in the query. Fields are identified by the table's name followed by a period and the field's name.
- `AS` establishes the alias for the field name preceding the alias. ZIP, for example, is the alias for the PostalCode field. The alias is the caption that appears in the field header for the PostalCode field.
- `INTO` specifies the name of the table into which the results of the query are to be placed. `INTO` applies only to action queries.
- `FROM` is the name of the table in which the fields are located.
- `WHERE` identifies the expressions that follow as the query's selection criteria.
- `AND` is a logical operator that results in records that meet both the criterion that

Oracle Ex. 1009, p. 309
Oracle v. DeBellis
IPR2024-00221

precedes the AND and the criterion that follows the AND.

- ORDER BY specifies the field or fields by which the query result table is to be sorted.

When you finish viewing the SQL statements, close the SQL window.

This chapter and Chapter 6, "Sorting, Finding, and Filtering Data in Tables," provide only a brief glimpse of SQL, and this example uses only a small cross-section of SQL's reserved words. Chapter 24, "Working with Structured Query Language," explains in detail the syntax that you use to edit and create SQL statements.



▶▶ See "What Is Structured Query Language?" p. 836

Oracle Ex. 1009, p. 310
Oracle v. DeBellis
IPR2024-00221

Chapter 9

Understanding Operators and Expressions in Access

Chapter 5, "Entering, Editing, and Validating Data in Tables," briefly introduced you to operators and the expressions that use them when you added validation rules to table fields. Chapter 8, "Using Query by Example," touched on expressions again when you devised selection criteria for the query that you created. You must use expressions with the forms (Chapters 12 and 13), reports (Chapters 14 and 15), and queries (Chapters 8 and 10) that you combine when creating custom Access applications; furthermore, you use expressions extensively when programming with Access VBA (Chapters 28 through 31). To work effectively with Access, therefore, you must know how to create simple expressions that use Access's group of functions and operators.

If you use spreadsheet applications, such as Microsoft Excel or Lotus 1-2-3, you might be familiar with using operators to create expressions. In spreadsheet applications, expressions are called *formulas*. As discussed in Chapter 4, "Working with Access Databases and Tables," the syntax for expressions that create default values, such as `=Date + 28`, is similar to formula entries in Excel. Conditional expressions that use the `=IIF` function in Excel use the `IIf` function in Access.

Much of this chapter is devoted to describing the functions available in Access for manipulating data of the Numeric and Text field data type. Functions play important roles in every element of Access, from validation rules for tables and fields of tables, to the control of program flow in Access VBA. You use functions when creating queries, forms, reports, and when writing Access VBA code. To use Access effectively, you must know what functions are available to you.

Oracle Ex. 1009, p. 311
Oracle v. DeBellis
IPR2024-00221

Understanding the Elements in Expressions

An *expression* is a statement of intent. If you want an action to occur after meeting a specific condition, your expression must specify that condition. To select records in a query that contains ZIP field values of 90000 or higher, for example, you type the expression

ZIP>=90000

You can use expressions in arithmetic calculations also. If you need an Extended Amount field in a query, for example, you type

[Extended Amount]: Quantity * [Unit Price]

as the expression to create calculated values in the data cells of the Extended Amount column.

> **Note**
>
> Square bracket pairs ([ ]) must surround object names, such as field names that contain spaces or punctuation other than the underscore character. You also use square bracket pairs to identify field names in expressions for validating data. The Northwind Traders sample database in prior versions of Access used field names that included spaces; in Access 97, the field names of tables in Northwind.mdb do not contain spaces, but table names include spaces. It is a Good Database Design Practice (GDBDP) never to use spaces in object names of any type. If you want to make object names more readable, separate words with the underscore character (_); for example, a field name of `Postal_Code` is preferable to the name `Postal Code`. In most cases, using upper- and lowercase object names provides adequate readability.

To qualify as an expression, a statement must have at least one operator and at least one literal, identifier, or function. The following list describes these elements:

- *Operators* include the familiar arithmetic symbols +, −, * (multiply), and / (divide), as well as many other symbols and abbreviations. Most other operators available in Access are equivalent to those operators found in traditional programming languages, such as BASIC, but some are specific to Access or SQL, such as the `Between`, `In`, `Is`, and `Like` operators.
- *Literals* consist of values that you type, such as **12345** or **ABCDE**. Literals are used most often to create default values and, in combination with field identifiers, to compare values in table fields.
- *Identifiers* are the names of objects in Access (such as fields in tables) that return distinct numeric or text values. The term *return*, when used with expressions, means that the present value of the identifier is substituted for its name in the expression. For example, the field name identifier `[CompanyName]` in an expression returns the value (a name) of the CompanyName field for the currently selected record. Access has five predefined named constants that also serve as identifiers: **`True`**, **`False`**, **`Yes`**, **`No`**, and **`Null`**. Named constants and variables that you create in Access VBA also are identifiers.

Oracle Ex. 1009, p. 312
Oracle v. DeBellis
IPR2024-00221

- *Functions* return a value in place of the function name in the expression, such as the **Date** and **Format$** functions, which are used in the examples in Chapter 5, "Entering, Editing, and Validating Data in Tables." Unlike identifiers, most functions require that you supply with parentheses an identifier or value as an argument. Later in this chapter, the section "Functions" explains functions and their arguments.

When literals, identifiers, or functions are used with operators, these combinations are called *operands*. The following sections explain these four elements of expressions more thoroughly.

Note

Expressions in this book appear in monospace type to distinguish expressions from the explanatory text of the book. Operators, including symbolic operators, built-in functions, and other reserved words and symbols of VBA, are set in **monospace bold** type. (VBA reserved words appear in blue color in the code-editing window of modules.) SQL operators and names of Access objects are set in monospace type.

Operators

Access provides six categories of operators that you can use to create expressions:

- *Arithmetic* operators perform addition, subtraction, multiplication, and division.
- *Assignment* and *comparison* operators set values and compare values.
- *Logical* operators deal with values that can only be true or false.
- *Concatenation* operators combine strings of characters.
- *Identifier* operators create unambiguous names for database objects so that you can assign the same field name, for example, in several tables and queries.
- Other operators, such as the `Like`, `Is`, and `Between` operators, simplify the creation of expressions for selecting records with queries.

Operators in the first four categories are available in almost all programming languages, including xBase and PAL (Paradox Application Language). Identifier operators are specific to Access; the other operators of the last category are provided only in relational database management systems (RDBMSs) that create queries using Query by Example (QBE) or Structured Query Language (SQL). The following sections explain how to use each of the operators in these categories.

Arithmetic Operators. Arithmetic operators operate only on numeric values and must have two numeric operands, with the following exceptions:

- When the minus sign (-) changes the sign (negates the value) of an operand. In this case, the minus sign is called the *unary minus*.
- When the equal sign (=) assigns a value to an Access object or an Access VBA variable identifier.

Oracle Ex. 1009, p. 313
Oracle v. DeBellis
IPR2024-00221

Table 9.1 lists the arithmetic operators that you can use in Access expressions.

Table 9.1 Arithmetic Operators

Operator	Description	Example
+	Adds two operands	`[Subtotal] + [Tax]`
–	Subtracts two operands	`Date – 30`
– (unary)	Changes the sign of an operand	`–12345`
*	Multiplies two operands	`[Units] * [UnitPrice]`
/	Divides one operand by another	`[Quantity] / 12.55`
\	Divides one integer operand by another	`[Units] \ 2`
Mod	Returns the remainder of division by an integer	`[Units] Mod 12`
^	Raises an operand to a power (exponent)	`[Value] ^ [Exponent]`

Access operators are identical to operators used in Microsoft QuickBASIC, QBasic (supplied with DOS 5.0 and later), and Visual Basic. If you aren't familiar with BASIC programming, the following operators need further explanation:

\ The integer division symbol is the equivalent of "goes into," as used in the litany of elementary school arithmetic: *Three goes into 13 four times, with one left over.* When you use integer division, operators with decimal fractions are rounded to integers, but any decimal fraction in the result is truncated.

Mod An abbreviation for *modulus*, this operator returns the *left over* value of integer division. Therefore, `13 Mod 4`, for example, returns `1`.

^ The exponentiation operator raises the first operand to the power of the second. For example, `2 ^ 4`, or two to the fourth power, returns 16 (2*2*2*2).

These three operators seldom are used in business applications but often are used in Access VBA program code.

Assignment and Comparison Operators. Table 9.1 omits the equal sign associated with arithmetic expressions because in Access you use it in two ways—neither of which falls under the arithmetic category. The most common use of the equal sign is as an assignment operator; = assigns the value of a single operand to an Access object or to a variable or constant. When you use the expression `= "Q"` to assign a default value to a field, the equal sign acts as an assignment value. Otherwise, = is a comparison operator that determines whether one of two operands is equal to the other.

Comparison operators compare the values of two operands and return logical values (**True** or **False**) depending on the relationship between the two operands and the operator. An exception is when one of the operands has the **Null** value. In this case, any comparison returns a value of **Null**. Because **Null** represents an unknown value, you cannot compare an unknown value with a known value and come to a valid **True** or **False** conclusion. The **Null** value is an important concept in database application, but few desktop RDBMSs support **Null** values.

Oracle Ex. 1009, p. 314
Oracle v. DeBellis
IPR2024-00221

Table 9.2 lists the comparison operators available in Access.

Table 9.2 Comparison Operators

Operator	Description	Example	Result
<	Less than	123 < 1000	True
<=	Less than or equal to	15 <= 15	True
=	Equal to	2 = 4	False
>=	Greater than or equal to	1234 >= 456	True
>	Greater than	123 > 123	False
<>	Not equal	123 <> 456	True

The principal uses of comparison operators are to create validation rules, to establish criteria for selecting records in queries, to determine actions taken by macros, and to control program flow in Access VBA.

Logical Operators. *Logical operators* (also called *Boolean operators*) are used most often to combine the results of two or more comparison expressions into a single result. Logical operators can combine only expressions that return the logical values `True`, `False`, or `Null`. With the exception of `Not`, which is the logical equivalent of the unary minus, logical operators always require two operands.

Table 9.3 lists the Access logical operators.

Table 9.3 Logical Operators

Operator	Description	Example 1 Example 2	Result 1 Result 2
`And`	Logical and	`True And True` `True And False`	`True` `False`
`Or`	Inclusive or	`True Or False` `False Or False`	`True` `False`
`Not`	Logical not	`Not True` `Not False`	`False` `True`
`Xor`	Exclusive or	`True Xor False` `True Xor True`	`True` `False`

The logical operators `And`, `Or`, and `Not` are used extensively in Access expressions. `Xor` is seldom used in Access. `Eqv` (equivalent) and `Imp` (implication) are rarely seen, even in programming code, so Table 9.3 omits these two operators.

Concatenation Operators. *Concatenation operators* combine two text values into a single string of characters. If you concatenate ABC with DEF, for example, the result is ABCDEF. The ampersand (`&`) is the preferred concatenation operator in Access because the character is SQL's standard concatenation symbol. Although you can use the + operator to link two strings of characters, using the + operator for concatenation can lead to ambiguities because the operator's primary purpose is to add two numeric operands.

Oracle Ex. 1009, p. 315
Oracle v. DeBellis
IPR2024-00221

Identifier Operators. The *identifier operators*, ! (the exclamation point, often called the *bang operator*) and . (the period, called the *dot operator* in Access), are separators and perform the following operations:

- Combine the names of object classes and object names to select a specific object or property of an object. For example, the following expression identifies the Personnel Actions form:

  ```
  Forms![Personnel Actions]
  ```

  This identification is necessary because you might also have a table called Personnel Actions.

- Distinguish object names from property names. Consider the following expression:

  ```
  TextBox1.FontSize = 8
  ```

  `TextBox1` is a control object, and `FontSize` is a property.

- Identify specific fields in tables, as in the following expression, which specifies the Company Name field of the Customers table:

  ```
  Customers![Company Name]
  ```

You use the ! character to separate object references; the general syntax is *`ObjectClass!ObjectName`*. The . character separates objects and their properties or methods, as in *`ObjectClass!Object.Property`* or *`ObjectClass!ObjectName.Method`*.

Other Operators. The remaining Access operators are related to the comparison operators. These operators return **`True`** or **`False`**, depending on whether the value in a field meets the chosen operator's specification. A **`True`** value causes a record to be included in a query; a **`False`** value rejects the record. When you use these operators in validation rules, entries are accepted or rejected based on the logical value returned by the expression.

Table 9.4 lists the four other operators used in Access.

Table 9.4 Other Operators

Operator	Description	Example
`Is`	Used with **`Null`** to determine whether a value is **`Null`** or **`Not Null`**.	`Is Null` `Is Not Null`
`Like`	Determines whether a string value begins with one or more characters (for `Like` to work properly, you must add a wild card, *, or one or more ?s).	`Like "Jon*"` `Like "FILE????"`
`In`	Determines whether a string value is a member of a list of values.	`In("CA", "OR", "WA")`
`Between`	Determines whether a numeric value lies within a specified range of values.	`Between 1 And 5`

Oracle Ex. 1009, p. 316
Oracle v. DeBellis
IPR2024-00221

You use the wildcard characters * and ? with the Like operator the same way that you use them in DOS. The * (often called *star* or *splat*) takes the place of any number of characters. The ? takes the place of a single character. For example, `Like "Jon*"` returns **True** for values such as *Jones* or *Jonathan*. `Like "*on*"` returns **True** for any value that contains *on*. `Like "FILE????"` returns **True** for *FILENAME*, but not *FILE000* or *FILENUMBER*. Wildcard characters can precede the characters that you want to match, as in `Like "*son"` or `Like "????NAME"`.

Except for Is, the operators in this other category are equivalent to the SQL reserved words LIKE, IN, and BETWEEN. Access includes these operators to promote compatibility with SQL. You can create each of these operators by combining other Access operators or functions. `Like "Jon*"` is the equivalent of Access VBA's **InStr**(**Left$**(*FieldName*, 3), "Jon"); `In("CA", "OR", "WA")` is similar to **InStr**("CAORWA", FieldName), except that no matches occur for the ambiguous AO and RW. `Between 1 And 5` is the equivalent of `>= 1 And <= 5`.

Literals

Access provides three types of literals that you can combine with operators to create expressions. The following list describes these types of literals:

- *Numeric* literals are typed as a series of digits, including the arithmetic sign and decimal point, if applicable. You don't have to prefix positive numbers with the plus sign; Access assumes positive values unless the minus sign is present. Numeric literals can include E or e and the sign of the exponent to indicate an exponent in scientific notation—for example, `1.23E 02`.

- *Text* (or *string*) literals can include any printable character, plus unprintable characters returned by the **Chr$** function. The **Chr$** function returns the characters specified by a numeric value from the ANSI character table (similar to the ASCII character table) that Windows uses. For example, **Chr$**(9) returns the Tab character. Printable characters include the letters A–Z, numbers 1–0, punctuation symbols, and other special keyboard symbols such as the tilde (~). Access expressions require that you enclose string literals within double quotation marks (""). Combinations of printable and unprintable characters are concatenated with the ampersand. For example, the following expression separates two strings with a newline pair.

  ```
  "First line" & Chr$(13) & Chr$(10) & "Second line"
  ```

  Chr$**(13) is the carriage return (CR), and **Chr$(10) is the line-feed (LF) character; together they form the newline pair.

  When you enter string literals in the cells of tables and Query Design grids, Access adds the quotation marks for you. In other places, you must enter the quotation marks yourself.

- *Date/time* literals are enclosed within number or pound signs (#), as in the expressions `#1-Jan-80#` or `#10:20:30#`. Access adds the enclosing pound signs if the program detects that you are typing into a Design grid a date or time in one of the standard Access date/time formats.

Oracle Ex. 1009, p. 317
Oracle v. DeBellis
IPR2024-00221

Identifiers

An *identifier* is usually the name of an object; databases, tables, fields, queries, forms, and reports are objects in Access. Each object has a name that uniquely identifies that object. Sometimes, to identify a subobject, an identifier name consists of a *family name* (object class) separated from a *given name* (object name) by a bang symbol or a period (an identifier operator). The family name of the identifier comes first, followed by the separator and then the given name. SQL uses the period as an object separator. An example of an identifier in an SQL statement is as follows:

```
Customers.Address
```

In this example, the identifier for the Address field object is contained in the Customers table object. *Customers* is the family name of the object (the table), and *Address* is the given name of the subobject (the field). In Access, however, you use the `!` symbol to separate table names and field names. (The period separates objects and their properties.) If an identifier contains a space or other punctuation, enclose the identifier within square brackets, as in this example:

```
[Personnel Actions]![paID]
```

You cannot include periods or exclamation points within the names of identifiers; `[PA!ID]`, for example, is not allowed.

In simple queries that use only one table, identifiers are usually the name of a field. You use identifiers to return the values of fields in form and report objects. Chapters 12 through 15 cover the specific method of identifying objects within forms and reports.

Functions

Functions return values to their names; functions can take the place of identifiers in expressions. One of the most common functions used in Access expressions is **`Now`**, which returns to its name the date and time from your computer's internal clock. (The empty parentheses are optional for the **`Now`** function.) If you type **`Now`** as the `DefaultValue` property of a table's Date/Time field, for example, `5/15/97 9:00` appears in the field when you change to Datasheet view (at 9:00 A.M. on March 15, 1997).



▶▶ See "Modules, Functions, and Subprocedures," p. 981

Access and VBA define about 140 individual functions. The following list groups functions by purpose:

- *Date and time* functions manipulate date/time values in fields or date/time values that you enter as literals. You can extract parts of dates (such as the year or day of the month) and parts of times (such as hours and minutes) with date and time functions.
- *Text-manipulation* functions are used for working with strings of characters.
- *Data-type conversion* functions enable you to specify the data type of values in Numeric fields instead of depending on Access to pick the most appropriate data type.

Oracle Ex. 1009, p. 318
Oracle v. DeBellis
IPR2024-00221

- *Mathematic and trigonometric* functions perform on numeric values operations that are beyond the capability of the standard Access arithmetic operators. You can use simple trigonometric functions, for example, to calculate the length of the sides of a right triangle (if you know the length of one side and the included angle).
- *Financial* functions are similar to functions provided by Lotus 1-2-3 and Microsoft Excel. They calculate depreciation, values of annuities, and rates of return on investments. To determine the present value of a lottery prize paid out in 25 equal yearly installments, for example, you can use the `PV` function.
- *General-purpose* functions don't fit any of the preceding classifications; you use these functions to create Access queries, forms, and reports.
- Other functions include those that perform dynamic data exchange (DDE) with other Windows applications, domain aggregate functions, SQL aggregate functions, and functions used primarily in Access VBA programming.

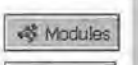
The following sections describe these functions more fully.

You can create user-defined functions by defining them with Access VBA programming code. Chapter 28, "Writing Visual Basic for Applications Code," describes how to create user-defined functions.

Using the Debug Window. When you write Access VBA programming code in a module, the Debug window is available to assist you in debugging your code. You also can use the module's Debug window to demonstrate the use and syntax of functions.

To experiment with some of the functions described in the following sections, perform these steps:

1. Click the Modules tab in the Database window.



2. Choose the New button to create a temporary module. Access assigns the default name `Module1` to the temporary module.



3. Click the Debug Window button of the toolbar or choose View, Debug Window. The Debug window appears, as shown in Figure 9.1. The entries shown in Figure 9.1 aren't visible at this point. You can create similar entries by using the functions for date and time discussed in the section "Functions for Date and Time" later in this chapter, and listed in Table 9.6.



4. Type **?Now** in the bottom portion of the Debug window (see Figure 9.1) and press Enter. The date and time from your computer's clock appear on the next line. The `?` is shorthand for the VBA **`Print`** statement (which displays the value of a function or variable) and must be added to the **`Now`** function to display the function's value.

   If you neglected to precede the function entry with `?` or **`Print`**, an error message appears, indicating that Access expected you to type a statement or an equal sign. Click OK and type **?** before the function name in the Debug window. Press End to return the caret to the end of the line and then Enter to retry the test.

Oracle Ex. 1009, p. 319
Oracle v. DeBellis
IPR2024-00221

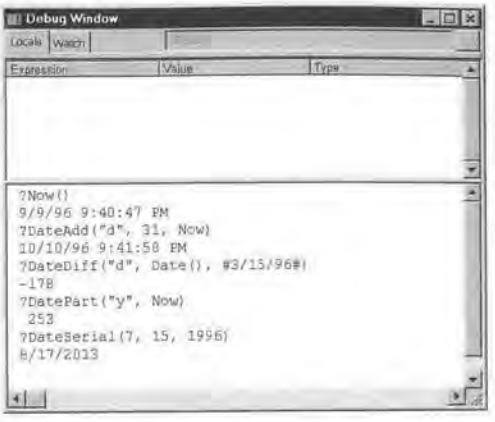


FIG. 9.1 Using the Debug window of a temporary module to experiment with functions.

The following sections describe and provide the correct syntax for the various functions available to Access users. This information should help you get acquainted with using functions with queries, forms, and reports. These descriptions and syntax examples are brief compared to the information available from the Access online Help system and in the Microsoft Access Language Reference.



As you type in your functions in the Debug window, Access displays a popup help window, showing the function's name and its complete argument list. An *argument list* is the list of information that you specify for the function to work on—for example, if you use the **Sqr** function to compute the square root of a number, you must supply a number inside the function's parentheses for the function to work on. Figure 9.2 shows the popup help window for the **Sqr** function. You can turn this feature on and off by choosing Tools, Options, and selecting (on) or clearing (off) the Auto Quick Info option on the Module page.

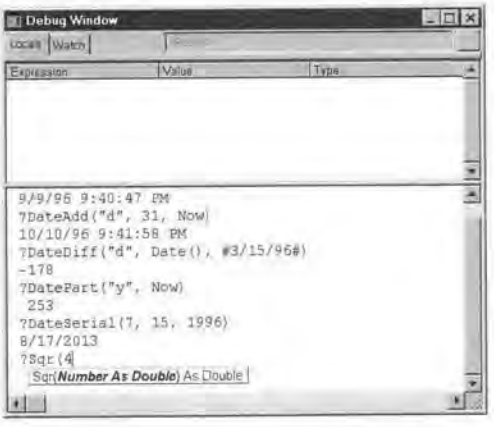


FIG. 9.2 A popup help window appears as you type functions in the Debug window.

Oracle Ex. 1009, p. 320
Oracle v. DeBellis
IPR2024-00221

Another way to learn more about functions is to choose Help, Contents and Index, click the Index tab, and type in the Index dialog's text box the name of the function on which you want more information. For a faster method to learn more about a particular function, however, follow these steps:

1. In the Debug window, press Enter to move the caret to the beginning of a new line.
2. Type the name of the function; select the name by double-clicking it or by pressing Shift+left arrow several times (as shown for the `Format` function in Figure 9.3). Alternatively, you can press Home to place the cursor in front of the function name's first character.
3. Press F1. The Help window opens. If a function and a property or event share the same name, an intermediate Context Help On window appears that enables you to choose the function, the event, or the property.
4. Click the See Also hotspot. Related Help topics appear in a Topics Found dialog (see Figure 9.4). Select a related topic and choose Display to show the associated Help window, or click Cancel to close the dialog.

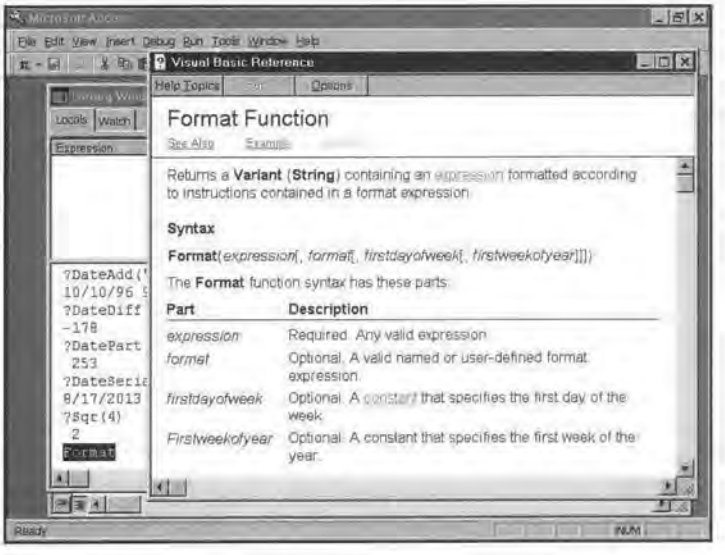


FIG. 9.3 The `Format` function help window.

5. After reviewing the syntax and other information concerning the function, close the Help window by clicking its Close Window button.

Help windows for functions have a standard format, as shown in Figure 9.3. If you click the Example hotspot in any function Help window, another window displays an example of the function used in Access VBA code. These examples show the syntax of the functions and appropriate arguments.

Oracle Ex. 1009, p. 321
Oracle v. DeBellis
IPR2024-00221

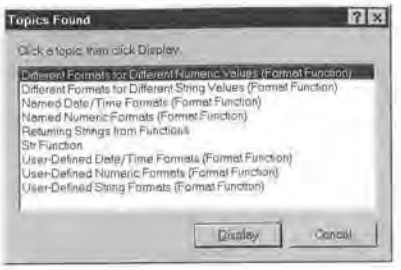


FIG. 9.4 The Topics Found dialog displayed by clicking the See Also hotspot.

The *Variant* Data Type in Access. `Variant` is a special data type unique to Microsoft Visual Basic dialects. The `Variant` data type enables you to concatenate values that ordinarily have different data types, such as an integer and a character string. The capability to concatenate different data types is called *turning off data-type checking*. `Variant` values are related to the `As Any` data type that Access VBA uses to turn off data-type checking when declaring external functions contained in Windows Dynamic Link Libraries (DLLs).

The `Variant` data type enables you to concatenate field values of tables and queries that have dissimilar data types without using VBA's data-type conversion functions such as `Str$`. (`Str$` converts numeric values to the `String` data type.) The `Variant` data type also simplifies expressions that combine field values to create concatenated indexes. The `Variant` data type also enables you to use the `&` symbol to concatenate values of different data types. SQL requires you to use the ampersand for such concatenation.

▶▶ See "Access Intrinsic Constants," p. 996

Table 9.5 lists the 16 subtypes of the `Variant` data type of Access VBA, along with the names of the intrinsic Visual Basic constants, `vbConstant`, corresponding to the `Variant` subtype value. Access 2.0 defined only `Variant` subtypes 0–8. In addition to the Access intrinsic constants that Access 2.0 originally defined, VBA provides its own set of intrinsic constants, which are prefixed with `vb`. Access intrinsic constants are prefixed with `ac`. Intrinsic constants, which you use primarily when writing Access VBA code, are one of the subjects of Chapter 29, "Understanding the Data Access Object Class."

Table 9.5 Subtypes of the `Variant` Data Type

Subtype	Constant	Corresponds To	Stored As
0	(None)	`Empty` (uninitialized)	Not applicable
1	`vbNull`	`Null` (no valid data)	Not applicable

Oracle Ex. 1009, p. 322
Oracle v. DeBellis
IPR2024-00221

Subtype	Constant	Corresponds To	Stored As
2	vbInteger	**Integer**	2-byte integer
3	vbLong	**Long**	4-byte long integer
4	vbSingle	**Single**	4-byte single-precision floating point
5	vbDouble	**Double**	8-byte double-precision floating point
6	vbCurrency	**Currency**	4-byte fixed point
7	vbDate	Date/Time	8-byte double-precision floating point
8	vbString	**String**	Conventional string variable
9	vbObject	**Object**	Automation object
10	vbError	**Error**	**Error** data type (error number)
11	vbBoolean	**Boolean**	**True** or **False** values only
12	vbVariant	**Variant**	Used with **Variant** arrays
13	vbDataObject	Special	Non-Automation object
17	vbByte	**Byte**	Numeric value from 0–255
8192	vbArray	**Array**	Used with **Variant** arrays

You can concatenate **Variant** values with **Variant** subtypes 1–8 listed in Table 9.5. You can concatenate a subtype 8 **Variant** (**String**) with a subtype 5 **Variant** (**Double**), for example, without receiving from Access the `Type Mismatch` error message displayed when you attempt this concatenation with conventional **String** (text) and **Double** data types. Access returns a value with the **Variant** subtype corresponding to the highest subtype number of the concatenated values. This example, therefore, returns a subtype 8 (**String**) **Variant** because 8 is greater than 5, the subtype number for the **Double** value. If you concatenate a subtype 2 (**Integer**) value with a subtype 3 (**Long**) value, Access returns subtype 3 **Variant** data.

Distinguishing between the **Empty** and **Null** **Variant** subtypes is important. **Empty** indicates that a variable that you created with Access VBA code has a name but doesn't have an initial value. **Empty** applies only to Access VBA variables (see Chapter 28, "Writing Visual Basic for Applications Code"). **Null** indicates that a data cell doesn't contain an entry. You can assign the **Null** value to a variable, in which case the variable is initialized to the **Null** value, **Variant** subtype 1.

▶▶ See "Variables and Naming Conventions," p. 988



You can experiment with **Variant** subtypes in the Debug window to become more familiar with using the **Variant** data type. VBA provides a function, **VarType**, that returns the integer value of its argument's subtype. Figure 9.5 shows the data subtype values that **VarType** returns for four variables (`A` to `D`) and the result of the concatenation of these variables (`E`).

Oracle Ex. 1009, p. 323
Oracle v. DeBellis
IPR2024-00221

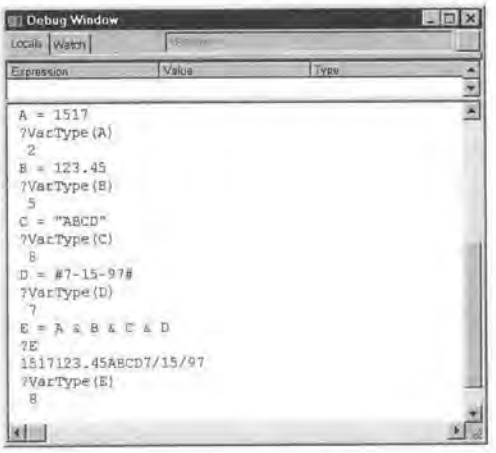


FIG. 9.5 The **Variant** subtype return values for four variables and the result of their concatenation.

◀◀ See "Choosing Field Data Types, Sizes, and Formats," p. 119
◀◀ See "Adding Indexes to Tables," p. 154

Functions for Date and Time. Access offers a variety of functions for dealing with dates and times. If you have used Visual Basic, you probably recognize most of the functions applicable to the Date/Time field data types shown in Table 9.6. VBA has several Date/Time functions, such as **DateAdd** and **DateDiff**, to simplify the calculation of date values.

All Date/Time values are stored as double-precision values but are returned as **Variant** subtype 7 unless you use the function's **String** form. The **String** form is identified by the **String** data type identification character, **$**, appended to the end of the function name. Both ? **VarType(Date$)** and ? **VarType(Time$)** return a subtype 8 (**String**) **Variant**.

Note

The Debug window in Figure 9.1 shows a few of the entries used to test the syntax examples of Table 9.6.

Table 9.6 Access Functions for Date and Time

Function	Description	Example	Returns
Date, Date$	Returns the current system date and time as a subtype 7 date **Variant** or a standard date **String** subtype 8.	**Date**	7/15/97 07-15-97
DateAdd	Returns a subtype 7 date with a specified number of days, weeks ("ww"), months ("m"), or years ("y") added to the date.	**DateAdd** ("d",31, #7/15/97#)	8/15/97

Oracle Ex. 1009, p. 324
Oracle v. DeBellis
IPR2024-00221

Function	Description	Example	Returns
DateDiff	Returns an **Integer** representing the difference between two dates using the d/w/m/y specification.	**DateDiff** ("d",**Date**, #4/15/95#)	-91 (assuming **Date** = 7/15/95) 7 (Saturday)
DatePart	Returns the specified part of a date such as day, month, year, day of week ("w"), and so on, as an **Integer**.	**DatePart** ("w", #7/15/95)	
DateSerial	Returns a subtype 7 **Variant** from year, month, and day arguments.	**DateSerial** (95,7,15)	7/15/95
DateValue	Returns a subtype 7 **Variant** that corresponds to a date argument in a character format.	**DateValue** ("15-Jul-95")	7/15/95
Day	Returns an **Integer** between 1 and 31 (inclusive) that represents a day of the month from a Date/Time value.	**Day**(**Date**)	15 (assuming that the date is the 15th of the month)
Hour	Returns an **Integer** between 0 and 23 (inclusive) that represents the hour of the Date/Time value.	**Hour** (#2:30 PM#)	14
Minute	Returns an **Integer** between 0 and 59 (inclusive) that represents the minute of a Date/Time value.	**Minute** (#2:30 PM#)	30
Month	Returns an **Integer** between 1 and 12 (inclusive) that represents the month of a Date/Time value.	**Month**(#15-Jul-95#)	7
Now	Returns the date and time of a computer's system clock as a **Variant** of subtype 7.	**Now**	7/15/95 11:57:28 AM
Second	Returns an **Integer** between 0 and 59 (inclusive) that represents the second of a Date/Time value.	**Second** (**Now**)	28
Time, **Time$**	Returns the Time portion of a Date/Time value from the system clock.	**Time** (returns subtype 7) **Time$** (returns String)	11:57:20 AM
TimeSerial	Returns the time serial value of the time expressed in integer hours, minutes, and seconds.	**TimeSerial** (11,57,20)	11:57:20 AM
TimeValue	Returns the time serial value of the time (entered as the **String** value) as a subtype 7 **Variant**.	**TimeValue** ("11:57")	11:57
Weekday	Returns day of the week (Sunday = 1) corresponding to the date as an **Integer**.	**Weekday** (#7/15/95#)	7
Year	Returns the year of a Date/Time value as an **Integer**.	**Year** (#7/15/95#)	1994

Oracle Ex. 1009, p. 325
Oracle v. DeBellis
IPR2024-00221

Text-Manipulation Functions. Table 9.7 lists the functions that deal with the Text field data type, corresponding to the `String` data type or `Variant` subtype 8. Most of these functions are modeled on BASIC string functions.

Table 9.7 Functions for `String` and Subtype 8 `Variant` Data Types

Function	Description	Example	Returns
`Asc`	Returns ANSI numeric value of character as an `Integer`.	`Asc("C")`	67
`Chr`, `Chr$`	Returns character corresponding to the numeric ANSI value as a `String`.	`Chr(67)` `Chr$(10)`	C (line feed)
`Format`, `Format$`	Formats an expression in accordance with appropriate format strings.	`Format (Date, "dd-mmm-yy")`	15-Jul-95
`InStr`	Returns the position of one string within another.	`InStr ("ABCD","C")`	3
`LCase`, `LCase$`	Returns the lowercase version of a string.	`LCase ("ABCD")`	abcd
`Left`, `Left$`	Returns the leftmost characters of a string.	`Left ("ABCDEF",3)`	ABC
`Len`	Returns the number of characters in a string as a `Long`.	`Len("ABCDE")`	5
`LTrim`, `LTrim$`	Removes leading spaces from string.	`LTrim (" ABC")`	ABC
`Mid`, `Mid$`	Returns a portion of a string.	`Mid ("ABCDE",2,3)`	BCD
`Right`, `Right$`	Returns the rightmost characters of a string.	`Right ("ABCDEF",3)`	DEF
`RTrim`, `RTrim$`	Removes trailing spaces from a string.	`RTrim ("ABC ")`	ABC
`Space`, `Space$`	Returns a string consisting of a specified number of spaces.	`Space(5)`	
`Str`, `Str$`	Converts the numeric value of any data type to a string.	`Str(123.45)`	123.45
`StrComp`	Compares two strings for equivalence and returns the integer result of the comparison.	`StrComp ("ABC", "abc")`	0
`String`, `String$`	Returns a string consisting of specified repeated characters.	`String (5, "A")`	AAAAA
`Trim`, `Trim$`	Removes leading and trailing spaces from a string.	`Trim (" ABC ")`	ABC
`UCase`, `UCase$`	Returns the uppercase version of a string.	`UCase ("abc")`	ABC
`Val`	Returns the numeric value of a string in a data type appropriate to the argument's format.	`Val ("123.45")`	123.45

Oracle Ex. 1009, p. 326
Oracle v. DeBellis
IPR2024-00221

Note

There are two different types of **String** variables: variable-length and fixed-length. Access 97 table fields of the Text data type are variable-length (**Variant** subtype 8) and can range from 0–255 characters. In Access VBA, variable-length strings can contain up to about two billion (2 ^ 31) characters. (Access 2.0 Basic limited variable-length strings to about 64K characters.) Fixed-length strings, which you declare in VBA with the syntax **Dim** *strVariable* **As String** * *lngLength*, can range from 1 to about 64K (2 ^ 16) characters. Access 97 Memo fields remain limited to a maximum of 64K characters for data entered at the keyboard, but can contain up to approximately 2 billion characters when text is entered via Access VBA code.

Table 9.7 shows two versions of most of the functions: one with, and one without a string-identifier character (**$**). In these cases, the function without **$** returns a **Variant** subtype 8 (**String**); the function with **$** returns the Text field data type (**String** in VBA). Figure 9.6 shows tests of a few of the string functions in the Debug window.

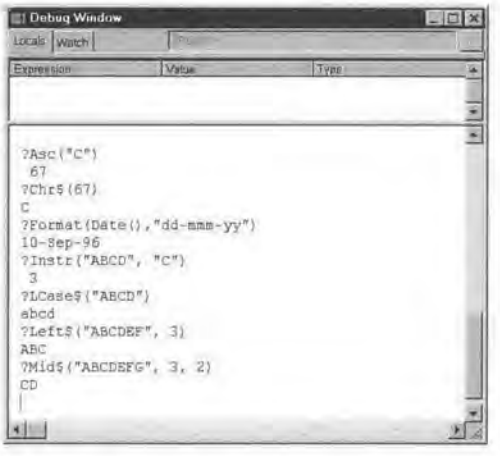


FIG. 9.6 Testing the string-manipulation functions in the Debug window.

Access's **Format** and **Format$** functions are identical to the **Format** and **Format$** functions of Visual Basic 4.0 and VBA (see Chapter 4, "Working with Access Databases and Tables," for the arguments of these functions). Table 9.7 doesn't include the **Tab** and **Spc** functions because these functions are used primarily to format the printing of text strings, which Access's built-in report-generation feature handles.

◀◀ See "Custom Display Formats," p. 125

Oracle Ex. 1009, p. 327
Oracle v. DeBellis
IPR2024-00221

Note

Access 97 stores **String** and **Variant** subtype 8 values in 32-bit Windows' Unicode format, which requires 2 bytes to define a single character. Unicode has the advantage of providing 2-byte codes for both alphabetic characters and the pictographic characters used in Asian languages. Access 97's string-manipulation functions count characters, not bytes. The `FieldSize` property of fields of Access 97 `Recordset` objects, however, is measured in bytes. Thus, values returned by the `FieldSize` property of a `Recordset` are twice those returned by the **Len** function. The difference between characters and bytes is significant in VBA programming that uses the `GetChunk` and `AppendChunk` methods, but not in the ordinary use of Access 97.

Numeric, Logical, Date/Time, and *String* Data-Type Conversion Functions. You can assign a particular data type to a numeric value with any of the data-type conversion functions. After you *freeze* (or *coerce*) a data type with one of the numeric data-type conversion functions, you cannot concatenate that data type with the **String** data type or data **Variant** subtype 7.

Table 9.8 lists the 11 numeric data-type conversion functions of Access 97. Access 97 introduces VBA's **Boolean**, **Byte**, and **Error** data types. The *NumValue* argument in the Syntax column can be any numeric or **String** value. However, if you use a **String** value as the argument of a numeric-type conversion function, the first character of the argument's value must be a digit, a dollar sign, a plus symbol, or a minus symbol.

Table 9.8 Data-Type Conversion Functions for Numeric, Time/Date, and `String` Values

Function	Description	Syntax
CBool	Converts a numeric value to a **Boolean** (**True** or **False**) data type.	**CBool**(*NumValue*)
Cbyte	Converts a numeric value to a **Byte** (0–255) data type.	**CByte**(*NumValue*)
CCur	Converts a numeric value to a **Currency** data type.	**CCur**(*NumValue*)
CDbl	Converts a numeric value to a **Double** data type.	**CDbl**(*NumValue*)
CInt	Converts a numeric value to an **Integer** data type.	**CInt**(*NumValue*)
CLng	Converts a numeric value to a **Long** integer data type.	**CLng**(*NumValue*)
CSng	Converts a numeric value to a **Single** data type.	**CSng**(*NumValue*)
CStr	Converts a numeric value to a **String** data type.	**CStr**(*NumValue*)
CVar	Converts a numeric value to a **Variant** data type.	**CVar**(*NumValue*)
CVDate	Converts a numeric value to a **Variant** subtype 7.	**CVDate**(*NumValue*)
CVErr	Converts a valid error number to create user-defined errors.	**CVErr**(*NumValue*)

Mathematic and Trigonometric Functions. Access provides a sufficient number of mathematic and trigonometric functions to meet most scientific and engineering requirements. You can create additional trigonometric functions with more complex expressions. If you are interested in more obscure transcendental functions, such as cosecants or hyperbolic functions, choose Help, Contents and Index, click the Index tab,

Oracle Ex. 1009, p. 328
Oracle v. DeBellis
IPR2024-00221

enter **math functions** in the text box, and then click Derived Math Functions in the Topics Found dialog. Table 9.9 lists the mathematic and trigonometric functions available directly in Access.

Table 9.9 Mathematic and Trigonometric Functions

Function	Description	Example	Returns
`Abs`	Returns the absolute value of a numeric value.	`Abs(--1234.5)`	1234.5
`Atn`	Returns the arctangent of a numeric value, in radians.	`Atn(1)`	.7853982
`Cos`	Returns the cosine of the angle represented by a numeric value, in radians.	`Cos(pi/4)`	.707106719949
`Exp`	Returns the exponential (antilog) of the numeric value.	`Exp(2.302585)`	9.9999990700 (rounding errors)
`Fix`	Identical to `Int`, except for negative numbers.	`Fix(--13.5)`	13
`Int`	Returns a numeric value with the decimal fraction truncated; the data type isn't changed unless the argument is a string.	`Int(13.5)` `Int(--13.5)`	13 14
`Log`	Returns natural (Napierian) logarithm of a numeric value.	`Log(10)`	2.302585
`Rnd`	Creates a random single-precision number between 0 and 1 when no argument is supplied. Must be initialized by first executing the `Randomize` statement.	`Rnd` (varies)	.533424
`Sgn`	Returns the sign of a numeric value: 0 if positive, –1 if negative.	`Sgn(--13.5)`	--1
`Sin`	Returns the sine of a numeric value, in radians.	`Sin(pi/4)`	.707106842423
`Sqr`	Returns the square root of a numeric value.	`Sqr(144)`	12
`Tan`	Returns the tangent of a numeric value, in radians.	`Tan(pi/4)`	1.0000001732 (fraction because of rounding)

The angles returned by the trigonometric functions are expressed in radians, as is the argument of the arctangent (`Atn`) function shown in the table. To obtain the values shown in Table 9.9's examples, type the expression `Pi = 3.141593` in the Debug window before entering the syntax example expressions. The returned values of the trigonometric functions are for an angle of approximately 45 degrees, corresponding to /4 radians.

Note

Because a 360-degree circle contains 2 radians, you convert radians to degrees with the expression `radians * 360/2` . Because `Pi` is a rounded value of and the trigonometric functions round results, you usually obtain values with rounding errors. The cosine of 45 degrees, for example, is 0.7070707... but Access returns 0.7071067.... These rounding errors are not significant in most applications.

`Int` and `Fix` differ in the following way: `Fix` returns the first negative integer less than or

Oracle Ex. 1009, p. 329
Oracle v. DeBellis
IPR2024-00221

equal to the argument; `Int` returns the first negative integer greater than or equal to the argument. `Int` and `Fix`, unlike other mathematic and trigonometric functions, return the integer value of a string variable but not the value of a literal string argument. Entering **? Int("13.5")**, for example, returns a data-type error message; however, if you type **A = 13.5** and then **? Int(A)**, Access returns 13.

Financial Functions. You might be interested in the financial functions of Access because you have used similar functions in Microsoft Excel or Lotus 1-2-3 spreadsheets. The financial functions of Access (supplied by VBA and listed in Table 9.10) are identical to their capitalized counterparts in Excel and employ the same arguments. If you have a table of fixed asset records, for example, you can use the depreciation functions to calculate monthly, quarterly, or yearly depreciation for each asset and then summarize the depreciation schedule in an Access report.

> **Note**
>
> A full description of the use and syntax of these functions is beyond the scope of this book. If you are interested in more details about these functions, choose Help, Contents and Index, click the Index tab, and enter the function name in the text box.

Table 9.10 Financial Functions for Calculating Depreciation and Annuities

Function	Description
`DDB`	Returns the depreciation of a fixed asset over a specified period by using the double-declining balance method.
`FV`	Returns the future value of an investment based on a series of constant periodic payments and a fixed rate of interest.
`IPmt`	Returns the amount of interest for an installment payment on a fixed-rate loan or annuity.
`IRR`	Returns the internal rate of return for an investment consisting of a series of periodic incomes and expenses.
`MIRR`	Returns the modified internal rate of return for an investment consisting of a series of periodic incomes and expenses.
`NPer`	Returns the number of payments of a given amount required to create an annuity or to retire a loan.
`NPV`	Returns the net present value of an annuity paid in equal periodic installments.
`Pmt`	Returns the amount of the periodic payment that must be made to create an annuity or to retire a loan.
`PPmt`	Returns the amount of principal in an installment payment on a fixed-rate loan or annuity.
`PV`	Returns the present value of an annuity paid in equal periodic installments.
`Rate`	Returns the interest rate of a loan or annuity based on a constant interest rate and equal periodic payments.
`SLN`	Returns the depreciation of a fixed asset over a specified period by using the straight-line method.
`SYD`	Returns the depreciation of a fixed asset over a specified period by the

Oracle Ex. 1009, p. 330
Oracle v. DeBellis
IPR2024-00221

Function	Description
	sum-of-the-years'-digits method.

Miscellaneous Functions. Access provides several functions that don't fit in any of the preceding categories but that you can use for creating queries and validating data entries, or with forms, reports, and macros. Table 9.11 lists these general-purpose functions.

Table 9.11 Miscellaneous Access Functions

Function	Description	Syntax
`Choose`	Returns a value from a list of values, based on the value's position in the list.	`Choose([Unit of Measure], "Each", "Dozen", "Gross")`
`IIf`	Returns one value if the result of an expression is **True**, another if the result is **False**.	`IIf([Order Quantity]Mod 12 = 0, "Dozen", "Each")`
`IsArray`	Returns **True** if the argument is an array; otherwise, returns **False**.	`IsArray(varName)`
`IsDate`	Returns **True** if the argument is the Date/Time field data type; otherwise, returns **False**.	`IsDate(FieldName)`
`IsEmpty`	Returns **True** if the argument is a noninitialized variable; otherwise, returns **False**.	`IsEmpty(varName)`
`IsError`	Returns **True** if the argument is an `Error` object; otherwise, returns **False**.	`IsError(varName)`
`IsMissing`	Returns **True** if an optional argument has not been passed to a procedure; otherwise, returns **False**.	`IsMissing(argOptional)`
`IsNull`	Returns **True** if the argument is **Null**; otherwise, returns **False**.	`IsNull(FieldName)`
`IsNumeric`	Returns **True** if the argument is one of the Number field data types; otherwise, returns **False**.	`IsNumeric(FieldName)`
`IsObject`	Returns **True** if the argument refers to a valid Automation object; otherwise, returns **False**.	`IsObject(objName)`
`Partition`	Returns a **String** value indicating the number of occurrences of a value within a range of values.	`Partition (Number, Start, Stop, Interval)`
`Switch`	Returns the value associated with the first of a series of expressions evaluating to **True**.	`Switch([Unit of Measure], "Each", 1, "Dozen", 12, "Gross", 144)`
`TypeName`	Returns the **String** value representing the name of the **Variant** data type; for instance, "Byte."	`TypeName(varName)`

Choose creates a lookup table that returns a value corresponding to a position from a list of values that you create. **Choose** is related closely to the **Switch** function, which returns a value associated with the first of a series of expressions evaluating to **True**. In Table 9.11's **Choose** example, if the value of the `Unit of Measure` field is 1, the function returns `Each`; if the value is 2, **Choose** returns `Dozen`; and if the value is 3, the function returns `Gross`. Otherwise, the function returns **Null**.

The **Switch** example returns a divisor value for an order. If the value of the Unit of Mea-

Oracle Ex. 1009, p. 331
Oracle v. DeBellis
IPR2024-00221

sure field is `Each`, **`Switch`** returns 1. If Unit of Measure is `Dozen`, the function returns 12; if Unit of Measure is `Gross`, **`Switch`** returns 144. The function returns **`Null`** for no matching value. **`Choose`** and **`Switch`** have similarities to the **`Select Case`** statement in VBA and other BASIC dialects (**`Null`** is the **`Case Else`** value).

The **`IIf`** function is called *in-line If*, or *immediate If*, because it substitutes for the multiline `If...Then...Else...End If` structure of the VBA conditional expression. In the example shown in Table 9.11, the **`IIf`** function returns `Dozen` if the quantity ordered is evenly divisible by 12; otherwise, the function returns `Each`.

The **`Partition`** function creates histograms. A *histogram* is a series of values (usually displayed as a bar chart) representing the frequency of events that can be grouped within specific ranges. A familiar histogram is a distribution of school examination grades, indicating the number of students who received grades A, B, C, D, and F. The grades might be based on a range of test scores from 90 to 100, 80 to 89, 70 to 79, 60 to 69, and less than 60. You establish the upper and lower limits of the data and then add the partition value.

Effective use of the **`Partition`** function requires typing or editing a SQL statement; the result of the query is most useful if presented in graphical form. Adding a histogram chart to a form is much easier than using the **`Partition`** function.

The eight **`Is`*`DataType`*** functions, four of which were new in Access 95, determine the type or value of data. You can use **`IsNull`** in validation rules and query criteria of one field to determine whether another field—whose field name is used as the argument—contains a valid entry. Although *`FieldName`* is the argument in the example syntax in Table 9.11, you can substitute an Access VBA variable name.

Other Functions. The chapters that cover other aspects of Access 97 describe and provide the syntax for the following special-purpose functions:

- *SQL aggregate functions* are described in Chapter 10, "Creating Multitable and Crosstab Queries." You use these functions most often with multiple-table queries that provide the data source forms. SQL aggregate functions return statistical data on the records selected by a query. You cannot use these functions in macros or call them from Access VBA modules, except within quoted strings used to create SQL statements to populate `Recordset` objects.
- *Domain aggregate functions* perform the same functions as the SQL aggregate functions, but on calculated values rather than the values contained in query fields. Chapter 12, "Creating and Using Forms," covers these functions. One domain aggregate function, **`DCount`**, is useful in validating entries in tables; this function is explained later in this chapter in the section "Expressions for Validating Data."
- The two dynamic data exchange functions, **`DDE`** and **`DDESend`**, are used to transfer data from and to other applications, respectively. In most cases, you use DDE functions with applications that don't support ActiveX Automation (formerly known as OLE Automation), the preferred method of interapplication data exchange.

Oracle Ex. 1009, p. 332
Oracle v. DeBellis
IPR2024-00221

- The remaining Access functions are used exclusively or almost exclusively in Access VBA modules, and are described in Part VII of this book, "Programming, Distributing, and Converting Access Applications."

Intrinsic and Named Constants

As noted earlier in this chapter, VBA and Access have many predefined intrinsic constants. The names of these constants are considered *keywords* because you cannot use these names for any purpose other than returning the value represented by the names, such as `-1` for **`True`** and `Yes`, `0` for **`False`** and `No`. (**`True`** and `Yes` are synonyms, as are **`False`** and `No`, so you can use these pairs of values interchangeably.) As mentioned earlier in the chapter, **`Null`** indicates a field with no valid entry. **`True`**, **`False`**, and **`Null`** are the most commonly used VBA intrinsic constants.

Named constants, which you define, return a single, predetermined value for the entire Access session. You can create named constants for use with forms and reports by defining them in the declarations section of an Access VBA module. Chapter 28, "Writing Visual Basic for Applications Code," describes how to create and use named constants such as `Pi` (used in the examples of trigonometric functions).

▶▶ See "Access Intrinsic Constants," p. 996



Creating Access Expressions

Chapter 5, "Entering, Editing, and Validating Data in Tables," uses several functions to validate data entry for most fields in the Personnel Actions table. Chapter 8, "Using Query By Example," uses an expression to select the states to be included in a mailing list query. These examples provide the foundation on which to build more complex expressions that can define more precisely the validation rules and query criteria for real-life database applications.

◀◀ See "Validating Data Entry," p. 170
◀◀ See "Creating Your First Real Query," p. 262



The topics that follow provide a few examples of typical expressions for creating default values for fields, validating data entry, creating query criteria, and calculating field values. The examples demonstrate the similarity of syntax for expressions with different purposes. Part III of this book, "Creating Forms and Reports," provides additional examples of expressions designed for use in forms and reports; Part VII, "Programming, Distributing, and Converting Access Applications," explains the use of expressions with Access VBA code.

Expressions for Creating Default Values

Expressions that create default field values can speed the entry of new records. Assigning values ordinarily requires you to use the assignment operator (=). When entering a default value in the Properties pane for a table in Design mode, however, you can enter a

Oracle Ex. 1009, p. 333
Oracle v. DeBellis
IPR2024-00221

simple literal. An example is the Q default value assigned to the paType field in Chapter 4, "Working with Access Databases and Tables." In this case, Access infers the = assignment operator and the quotation marks surrounding the Q. To adhere to the rules of creating expressions, the default value entry must be = "Q". You often can use shorthand techniques when typing expressions because Access infers the missing characters. If you enter = **"Q"**, you achieve the same result; Access doesn't infer the extra characters.

You can use complex expressions for default values if the result of the expression conforms to or can be converted by Access to the proper field data type. You can enter = **1** as the default value for the paType field, for example, although 1 is a Numeric field data type and paType is a Text type field.

Expressions for Validating Data

The Personnel Actions table uses several expressions to validate data entry. The validation rule for the paID field is > 0; the rule for the paType field is "S" **Or** "Q" **Or** "Y" **Or** "B" **Or** "C"; the rule for the paApprovedBy field is > 0 **Or Is Null**. The validation rule for the paID field is equivalent to the following imaginary in-line **IIf** function:

```
IIf(DataEntry > 0, paID = DataEntry,
    MsgBox("Please enter a valid employee ID number."))
```

Access tests *DataEntry* in the validation rule expression. If the validation expression returns **True**, the value of *DataEntry* replaces the value in the current record's field. If the expression returns **False**, a message box displays the validation text that you typed. **MsgBox** is a function used in VBA programming to display a message box on-screen. You cannot type the imaginary validation rule just described; Access infers the equivalent of the imaginary **IIf** expression after you add the ValidationRule and ValidationText properties with entries in the two text boxes for the paID field.

You might change the expression "S" **Or** "Q" **Or** "Y" **Or** "B" **Or** "C", which you use to test the paType field, to a function. The **In** function provides a simpler expression that accomplishes the same objective:

```
In("S", "Q", "Y", "B", "C")
```

Alternatively, you can use the following table-level validation expression:

```
InStr("SQYBC",[paID]) > 0
```

Both expressions give the same result, but you can use **InStr** only for table-level validation because one of its arguments refers to a field name. Thus, the **In** function provides the better solution.

Expressions for Query Criteria

When creating Chapter 8's qryUSMailingList query to select records from the states of California, Oregon, and Washington, you enter ="CA", ="OR", and ="WA" on separate lines. A better expression is **In**("CA", "OR", "WA"), entered on the same line as the ="USA" criterion for the Country field. This expression corrects the failure to test the Country field for a value equal to USA for the OR and WA entries.

Oracle Ex. 1009, p. 334
Oracle v. DeBellis
IPR2024-00221

◀◀ See "Creating More Complex Queries," p. 269



You can use a wide range of other functions to select specific records to be returned to a query table. Table 9.12 shows some typical functions used as query criteria applicable to the Northwind Traders tables.

Table 9.12 Typical Expressions Used as Query Criteria

Table	Field	Expression	Records Returned
Customers	Country	`Not "USA" And Not "Canada"`	Firms other than those in the U.S. and Canada.
Customers	Country	`Not ("USA" Or "Canada")`	Firms other than those in the U.S. and Canada; the parentheses apply the condition to both literals.
Customers	CompanyName	`Like "[N-Z]*"`	Firms with names beginning with N through Z, outside the U.S.
Customers	CompanyName	`Like S* Or Like V*`	Firms with names beginning with S or V (Access adds Like and quotation marks).
Customers	CompanyName	`Like "*shop*"`	Firms with *shop*, *Shop*, *Shoppe*, or *SHOPPING* in the firm name.
Customers	PostalCode	`>=90000`	Firms with postal codes greater than or equal to 90000.
Orders	OrderDate	`Year([OrderDate]) = 1995`	Orders received to date, beginning with 1/1/1995.
Orders	OrderDate	`Like "*/*/95"`	Orders received to date, beginning with 1/1/1995; using wild cards simplifies expressions.
Orders	OrderDate	`Like "1/*/95"`	Orders received in the month of January 1995.
Orders	OrderDate	`Like "1/?/95"`	Orders received from the 1st to the 9th of January 1995.
Orders	OrderDate	`Year([OrderDate] = 1995 And DatePart("q", [OrderDate]) = 1`	Orders received in the first quarter of 1995.
Orders	OrderDate	`Between #1/1/95# And #3/31/95#`	Orders received in the first quarter of 1995.
Orders	OrderDate	`Year([OrderDate] = 1995 And DatePart("ww", [OrderDate]) = 10`	Orders received in the 10th week of 1995.
Orders	OrderDate	`>= DateValue ("1/15/95")`	Orders received on or after 1/15/95.
Orders	Shipped Date	`Is Null`	Orders not yet shipped.
Orders	Order Amount	`>= 5000`	Orders with values greater than or equal to $5,000.

(continues)

Oracle Ex. 1009, p. 335
Oracle v. DeBellis
IPR2024-00221

Table 9.12 Continued

Table	Field	Expression	Records Returned
Orders	Order Amount	`Between 5000 And 10000`	Orders with values greater than or equal to $5,000 and less than or equal to $10,000.
Orders	Order Amount	`< 1000`	Orders less than $1,000.

The wildcard characters used in `Like` expressions simplify the creation of criteria for selecting names and dates. As in DOS, the asterisk (*) substitutes for any legal number of characters, and the question mark (?) substitutes for a single character. When a wildcard character prefixes or appends a string, the matching process loses its default case-sensitivity. If you want to match a string without regard to case, use the following expression:

```
UCase$(FieldName) = "MATCH STRING"
```

Note

The Orders table of the Northwind Traders sample database supplied with Access 1.0 and 1.1 included an Order Amount field. The value in the Order Amount field was set equal to the sum of the products of the Quantity and Unit Price fields, less the Discount percentage, of the records in the Order Details table. Including the Order Amount field violates the rule of relational databases that requires that all data in the fields of primary tables (Orders is a primary table) be dependent on the primary key and independent of other fields, either in the table or in other tables. The Order Amount field's value depends on the entries in the Order Details table.

Entering a Query Criterion. To experiment with query criteria expressions, follow these steps:



1. Click the Queries tab of the Database window, click the New button to open a new query, then click OK when the New Query dialog appears.
2. Select the Customers table from the Tables list of the Show Table dialog, and click the Add button. Repeat this process for the Orders and Order Details table. (Alternatively, you can add a table from the Show Table dialog by double-clicking the table.) Click the Close button to close the Add Table dialog. The CustomerID fields of the Customers and Orders tables and the OrderID fields of the Orders and Order Details tables are joined; *joins* are indicated by a line between the fields of the two tables. (The next chapter covers joining multiple tables.)
3. Add the CompanyName, PostalCode, and Country fields of the Customers table to the query. You can add fields by selecting them from the Field drop-down list in the Query Design grid, by clicking a field in the Customers field list above the grid and dragging the field to the desired Field cell in the grid, or by double-clicking a field in the Customers field list above the grid.

Oracle Ex. 1009, p. 336
Oracle v. DeBellis
IPR2024-00221

4. Add to the query the OrderID, OrderDate, ShippedDate, and Freight fields of the Orders table. Use the horizontal scroll bar slider under the Query Design grid to expose additional field columns as necessary. Choose View, Table Names to display the table names in the Tables row of the Query Design grid, if necessary. Place the caret in the Sort row of the OrderID field, press F4 to open the Sort list box, and select Ascending Sort.

5. Click the Totals button of the toolbar or choose View, Totals to add the Total row to the Query Design grid. The default value, `Group By`, is added to the Total cell for each field of your query.



6. Scroll the grid so that the Freight column appears. Click the selection bar above the Field row to select the Freight column and press the Insert key to add a new column.

7. Type **Amount: CCur([UnitPrice]*[Quantity]*(1—[Discount]))** in the new column's Field cell (see Figure 9.7). This expression calculates the net amount of each line item in the Order Details table and formats the column as if the field data type were Currency. The next section discusses how to use expressions to create calculated fields.

8. Move the caret to the Total row of the new column and press F4 to open the drop-down list. Choose **Sum** from the list (refer to Figure 9.7). The Sum option totals the net amount for all the line items of each order in the Orders table. In the next chapter, you learn the details of how to create queries that group data.

   The Total row for all the other columns of the query shows Group By. Make sure that you mark the Show check box so that your new query column appears when you run the query. (Do not make an entry in the Table row of your new query column; if you do, you receive an error message when you run the query.)

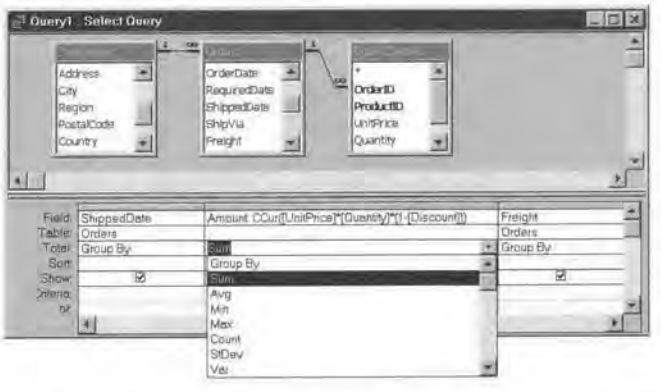
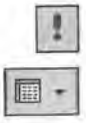


FIG. 9.7 The query design for testing the use of expressions to select records with values that meet criteria.

Oracle Ex. 1009, p. 337
Oracle v. DeBellis
IPR2024-00221



9. Click the Run or Datasheet View button of the toolbar to run your new query. Your query appears as shown in Figure 9.8. The Amount column contains the total amount of the order and the net of discounts, if any.



Query1 : Select Query

CompanyName	PostalCode	Country	Order ID	Order Date	Shipped Da	Amount	Freight
Vins et alcools Cheval	51100	France	10248	04-Aug-94	16-Aug-94	$440.00	$32.38
Toms Spezialitäten	44087	Germany	10249	05-Aug-94	10-Aug-94	$1,863.40	$11.61
Hanari Carnes	05454-876	Brazil	10250	08-Aug-94	12-Aug-94	$1,552.60	$65.83
Victuailles en stock	69004	France	10251	08-Aug-94	15-Aug-94	$654.06	$41.34
Suprêmes délices	B-6000	Belgium	10252	09-Aug-94	11-Aug-94	$3,597.90	$51.30
Hanari Carnes	05454-876	Brazil	10253	10-Aug-94	16-Aug-94	$1,444.80	$58.17
Chop-suey Chinese	3012	Switzerlan	10254	11-Aug-94	23-Aug-94	$556.62	$22.98
Richter Supermarkt	1203	Switzerlan	10255	12-Aug-94	15-Aug-94	$2,490.50	$148.33
Wellington Importador	08737-363	Brazil	10256	15-Aug-94	17-Aug-94	$517.80	$13.97
HILARIÓN-Abastos	5022	Venezuela	10257	16-Aug-94	22-Aug-94	$1,119.90	$81.91
Ernst Handel	8010	Austria	10258	17-Aug-94	23-Aug-94	$1,614.88	$140.51
Centro comercial Moc	05022	Mexico	10259	18-Aug-94	25-Aug-94	$100.80	$3.25
Ottilies Käseladen	50739	Germany	10260	19-Aug-94	29-Aug-94	$1,504.65	$55.09
Que Delícia	02998-873	Brazil	10261	19-Aug-94	30-Aug-94	$448.00	$3.05
Rattlesnake Canyon G	87110	USA	10262	22-Aug-94	25-Aug-94	$584.00	$48.29

Record: 1 of 831

FIG. 9.8 The query result set for the query design shown in Figure 9.7.

Using the Expression Builder to Add Query Criteria. After creating and testing your query, you can apply criteria to limit the number of records that the query returns. You can use Access's Expression Builder to simplify the process of adding record-selection criteria to your query. To test some of the expressions listed in Table 9.12, follow these steps:



1. Click the Design View button of the toolbar to change to Query Design mode.
2. Place the caret in the Criteria row of the field for which you want to establish a record-selection criterion.



3. Click the Build button of the toolbar to display the Expression Builder's window. Alternatively, you can right-click the Criteria row, then choose Build from the popup menu.
4. In the expression text box at the top of Expression Builder's window, type one of the expressions from Table 9.12. Figure 9.9 shows the sample expression `Like "*shop*"` that applies to the Criteria row of the Company Name column. You can use the Like button under the expression text box as a shortcut for entering `Like`.
5. Click OK to return to the Query Design grid. In the field where the caret is located, the Expression Builder places the expression that you built (see Figure 9.10).

Oracle Ex. 1009, p. 338
Oracle v. DeBellis
IPR2024-00221



FIG. 9.9 Entering a criterion in the Expression Builder to create a criterion to match *shop*.

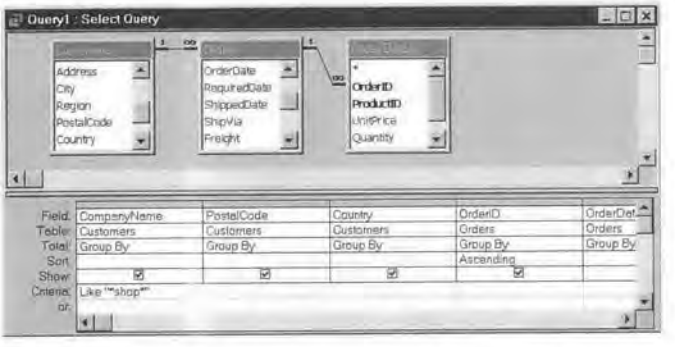


FIG. 9.10 The Query Design grid with the expression that you created in the Expression Builder.

6. Click the Run button of the toolbar to test the expression. The query result for the example in Figure 9.10 appears as shown in Figure 9.11.

7. Return to Query Design mode, then select and delete the expression by pressing the Delete key.

8. Repeat steps 2–7 for each expression that you want to test. When you test expressions using Date/Time functions, sort the OrderDate field in ascending order. Similarly, sort on the Amount field when queries are based on amount criteria. You can alter the expressions and try combinations with the implied And condition by entering criteria for other fields in the same row. Access warns you with an error message if you make a mistake in an expression's syntax.

Oracle Ex. 1009, p. 339
Oracle v. DeBellis
IPR2024-00221

9. After you finish experimenting, save your query with a descriptive name, such as **qryOrderAmount**.



FIG. 9.11 The query result set resulting from adding the `Like "*shop*"` criteria to the Company Name field.

Expressions for Calculating Query Field Values

The preceding section demonstrated that you can use expressions to create new, calculated fields in query tables. Calculated fields display data computed based on the values of other fields in the same row of the query table. Table 9.13 shows some representative expressions that you can use to create calculated query fields.

Table 9.13 Typical Expressions to Create Calculated Query Fields

Column Name	Expression	Values Calculated
TotalAmount	`[Amount] + [Freight]`	Sum of the OrderAmount and Freight fields.
FreightPercent	100 * [Freight]/[Amount]	Freight charges as a percentage of the order amount.
FreightPct	Format([Freight]/[Amount], "Percent")	Freight charges as a percentage of the order amount, but with formatting applied.
SalesTax	Format([Amount] * 0.05, "$#,###.00")	Sales tax of 5 percent of the amount of the order added with a display similar to the Currency data type.

To create a query containing calculated fields, follow these steps:

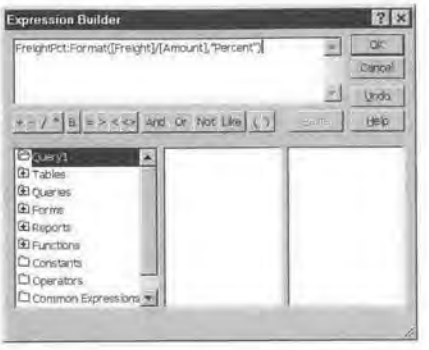
1. In Query Design view, move to the first blank column of the query that you created in the preceding section. Type the column name shown in Table 9.13, followed by a colon and then the expression. (Click the Build button of the toolbar to use the Expression Builder to enter the expression, as shown in Figure 9.12, or press Shift+F2 to use the Zoom box to enter the expression.) The expression for the example in Figure 9.12 is

```
FreightPct: Format([Freight]/[Amount],"Percent")
```


Oracle Ex. 1009, p. 340
Oracle v. DeBellis
IPR2024-00221

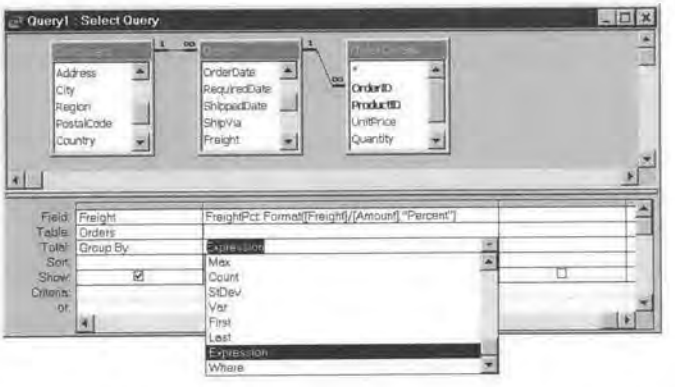
If you don't type the field name and colon, Access provides the default Expr1 as the calculated field name.

FIG. 9.12 Entering an expression in the Expression Builder to create a calculated field in a query.



2. Place the caret in the Total cell of the calculated field and select Expression from the drop-down list, as shown in Figure 9.13. (If you don't select Expression, your query opens a Parameters dialog or returns an error message when you attempt to execute it.)

FIG. 9.13 The query design for two of the calculated fields of Table 9.13



3. Run the query. The result set for the query with the calculated field appears as shown in Figure 9.14.

4. Repeat steps 1, 2, and 3 for each of the four examples in Table 9.13.

Oracle Ex. 1009, p. 341
Oracle v. DeBellis
IPR2024-00221



FIG. 9.14 The query result set displaying the order total and freight charges as a percent of the order amount.

You use the **Format** function with your expression as its first argument to display the calculated values in a more readable form. When you add the percent symbol (%) to a format expression or specify `"Percent"` as the format, the value of the expression argument multiplies by 100, and the percent symbol preceded by a space appends to the displayed value. (Access 2.0 introduced `"Percent"` as a new format specifier equivalent to entering `"0.00%"`.)

Figure 9.15 shows the rather complex SQL statement that creates the query with two calculated fields in Access's SQL window. (Choose View, SQL or right-click a blank area of the upper pane and choose View SQL from the popup menu to display the SQL window.) Note that you cannot include periods to indicate abbreviations in field names. Periods and exclamation points are identifier operators, and you cannot include them within identifiers.

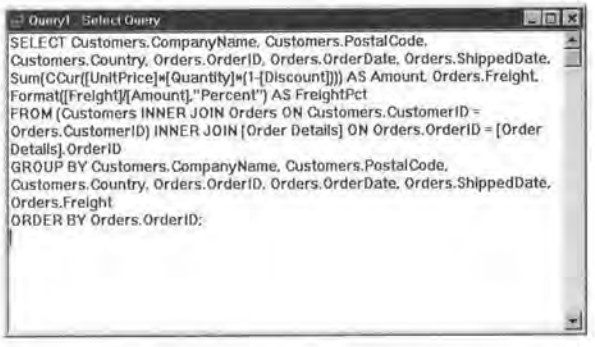


FIG. 9.15 The SQL statement used to create a query with two calculated fields.

Troubleshooting

When attempting to execute a query that contains an expression, a `Can't evaluate expression` *or Wrong data type message box appears.*

Oracle Ex. 1009, p. 342
Oracle v. DeBellis
IPR2024-00221

The `Can't evaluate expression` message usually indicates a typographic error in naming a function or an object. Depending on the use of the function, an Enter Parameter Value dialog might appear if the named object does not exist. The `Wrong data type` message is most likely to occur as a result of attempting to use mathematic or trigonometric operators with values of the Text or Date/Time field data types. If your expression refers to a control contained in a `Form` or `Report` object, the form or report must be open when you execute the function.

Other Uses for Expressions

You can use expressions with update queries, to calculate fields in forms and reports, as conditions for the execution of a VBA procedure, or as an argument for an action you program in VBA. SQL `SELECT` statements use expressions as in the following fragment:

```
WHERE [Birth Date] >= #1/1/60#
```

See Chapter 24, "Working with Structured Query Language," for more information. Access VBA code also uses expressions extensively to control program flow and structure. The chapters focusing on SQL and Access VBA programming (Parts VI and VII) describe these uses for expressions.

Oracle Ex. 1009, p. 343
Oracle v. DeBellis
IPR2024-00221

Chapter 10

Creating Multitable and Crosstab Queries

Your purpose in acquiring Access is undoubtedly to take advantage of this application's relational database management capabilities. To do so, you have to be able to link related tables based on key fields that have values in common—a process known as a *join* in database terms. Chapters 8, "Using Query by Example," and 9, "Understanding Operators and Expressions in Access," showed you how to create simple queries based on a single table. If you tried the examples in Chapter 9, you saw a glimpse of a multiple-table query when you joined the Order Details table to the Orders table, which you then joined to the Customers table to create the query for testing expressions. The first part of this chapter deals exclusively with queries created from multiple tables that are related through joins.

This chapter provides examples of queries that use each of the four basic types of joins that you can create in Access's Query Design View: *equi-joins*, *outer joins*, *self-joins*, and *theta joins*. Two of the three new query features introduced by Access 2.0, subqueries and `UNION` queries, cannot be used in the queries that you design in Access's graphic Query by Example (QBE) window. You can implement these two new query features only by writing SQL statements—the subject of Chapter 24, "Working with Structured Query Language." Some of the example queries in this chapter use the Personnel Actions table that you created in Chapter 4, "Working with Access Databases and Tables." If you didn't create the Personnel Actions table, refer to the "Creating the Personnel Actions Table" section of Chapter 4 or to Appendix C, "Data Dictionary for the Personnel Actions Table," for instructions on how to build this table.

Other example queries build on queries that you create in preceding sections. You will find, therefore, that reading this chapter and creating the example queries sequentially, as the queries appear in text, is more efficient than taking the random approach.

Oracle Ex. 1009, p. 344
Oracle v. DeBellis
IPR2024-00221

This chapter also includes descriptions and examples of four of the five categories of queries that you can create with Access: *select*, *summary*, *parameter*, and *crosstab* queries. Four types of action queries exist that you can use to create or modify data in tables: *Make-table*, *Append*, *Delete*, and *Update*. The next chapter, "Using Action Queries," presents typical applications for and examples of each type of action query.

Joining Tables to Create Multitable Queries

Before you can create joins between tables, you must know the contents of the tables' fields and which fields are related by common values. As mentioned in Chapter 4, assigning identical names to primary- and foreign-key fields in different tables that contain related data is a common practice. This approach, used by Microsoft when creating the Northwind Traders sample database, makes determining relationships and creating joins among tables easier. The CustomerID field in the Customers table and the CustomerID field in the Orders table, for example, are used to join orders with customers.

Figure 10.1 shows the structure of the Northwind Traders database with a graphical display of the joins between the tables. Access query designs indicate joins with lines between field names of different tables. Bold type indicates primary-key fields. Each join usually involves at least one primary-key field.

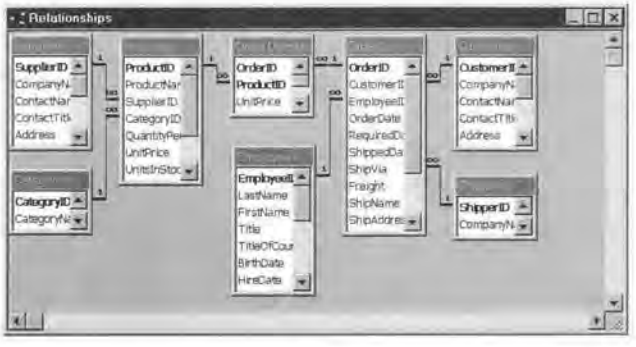


FIG. 10.1 The joins between the tables of the Northwind Traders sample database.




You can display the structure of the joins between the tables in Access 97's Northwind Traders database by giving the Database window the focus (click the Database Window button of the toolbar) and then clicking the Relationship button of the toolbar or choosing Tools, Relationships. The 1 above the line that shows the join between two tables in Figure 10.1 indicates the "one" side of a one-to-many relationship; the infinity symbol ( ) indicates the "many" side.

Note

Access 97 lets you choose between displaying only the direct relationships for a single table (the Show Direct Relationships button on the toolbar) or all relationships for all tables in a database (the

Oracle Ex. 1009, p. 345
Oracle v. DeBellis
IPR2024-00221

Show All Relationships button). All tables of Northwind.mdb appear by default when you open the Relationships window of the Northwind sample database. In this case, clicking the Show Direct Relationships button has no effect.

To show relationships for only one table, click the toolbar's Clear Layout button, click the Show Table button to display the Show Table dialog, select the table to display in the Tables list, and then click Add and Close. Click the Show Direct Relationships button to display the relationships for the selected table. Clearing the layout of the Relationships windows does not affect the underlying relationships between the tables. The Show Direct Relationships feature is useful primarily with databases that contain many related tables.

Access supports four types of joins in the graphical QBE Design mode:

- *Equi-joins* (also called *inner joins*) are the most common join for creating select queries. Equi-joins display in one table all of the records with corresponding records in another table. The correspondence between records is determined by identical values (`WHERE` *field1* = *field2* in SQL) in the fields that join the tables. In most cases, joins are based on a unique primary-key field in one table and a foreign-key field in the other table in a one-to-many relationship. If none of the table's records that act as the *many* side of the relationship have a field value that corresponds to a record in the table of the *one* side, the corresponding records in the *one* side don't appear in the query result.

  Access automatically creates the joins between tables if the tables share a common field name that is a primary key of one of the tables. These joins are also automatically created if you previously specified the relationships between the tables in the Relationships window.

- *Outer joins* are used in database maintenance to remove orphan records and duplicate data from tables by creating new tables that contain records with unique values. Outer joins display records in one member of the join, regardless of whether corresponding records exist on the other side of the join.

- *Self-joins* relate data within a single table. You create a self-join in Access by adding to the query a duplicate of the table (Access provides an alias for the duplicate) and then creating joins between the fields of the copies.

- *Theta joins* relate data by using comparison operators other than =. Theta joins include *not-equal joins* (<>) used in queries designed to return records that lack a particular relationship. You implement theta joins by `WHERE` criteria rather than by the SQL `JOIN` reserved word. The Query Design window does not indicate theta joins by drawing lines between field names, nor do theta joins appear in the Relationships window.

Creating Conventional Single-Column Equi-Joins

Joins based on one column in each table are known as *single-column equi-joins*. Most relational databases are designed to employ single-column equi-joins only in one-to-many

Oracle Ex. 1009, p. 346
Oracle v. DeBellis
IPR2024-00221

relationships. The following list details the basic rules for creating a database that enables you to use simple single-column equi-joins for all queries:



◀◀ See "Establishing Relationships Between Tables," p. 146

- Each table on the *one* side of the relationship must have a primary key with a No Duplicates index to maintain referential integrity. Access automatically creates a No Duplicates index on the primary-key field(s) of a table.
- Many-to-many relationships, such as the relationship of Orders to Products, are implemented by an intermediary table (in this case, Order Details) having a one-to-many relationship (Orders to Order Details) with one table and a many-to-one relationship (Order Details to Products) with another.
- Duplicated data in tables, where applicable, is extracted to a new table that has a primary-key, no-duplicates, one-to-many relationship with the table from which the duplicate data is extracted. Using a multicolumn primary key to identify extracted data uniquely often is necessary because individual key fields might contain duplicate data. The combination (also known as *concatenation*) of the values of the key fields, however, must be unique. Access 97's Table Analyzer Wizard locates and extracts most duplicate data automatically. Chapter 11 describes how to extract duplicate data manually from tables.



▶▶ See "Creating New Tables with Make-Table Queries," p. 371

All of the joins in the Northwind Traders database, shown by the lines that connect field names of adjacent tables in Figure 10.1, are single-column equi-joins between tables with one-to-many relationships. Access uses the ANSI SQL-92 reserved words `INNER JOIN` to identify conventional equi-joins, and `LEFT JOIN` or `RIGHT JOIN` to specify outer joins.

Among the most common uses for queries based on equi-joins is to match customer names and addresses with orders received. You might want to create a simple report, for example, that lists the customer name, order number, order date, and amount. To create a conventional one-to-many, single-column equi-join query that relates Northwind's customers to the orders that the customers place, sorted by company and order date, follow these steps:

1. If Northwind.mdb is open, close all windows except the Database window by double-clicking the windows' close box. Otherwise, choose File, 1 Northwind.mdb to open the Northwind Traders database.



2. Click the Queries tab of the Database window and then choose New to create a new query. The New Query dialog appears. With the Design View item selected, click OK. Access displays the Show Table dialog superimposed on an empty Query Design window.

Oracle Ex. 1009, p. 347
Oracle v. DeBellis
IPR2024-00221

3. Select the Customers table from the Show Table list and click the Add button. Alternatively, you can double-click the Customers table name to add the table to the query. Access adds to the Query Design window the Field Names list for Customers.

4. Double-click the Orders table in the Show Table list, and then click the Close button. Access adds to the window the Field Names list for Orders, plus a line that indicates a join between the CustomerID fields of the two tables, as shown in Figure 10.2. Access creates the join automatically because CustomerID is the key field of the Customers table, and Access found a field with the same field name (a foreign key) in the Orders table.

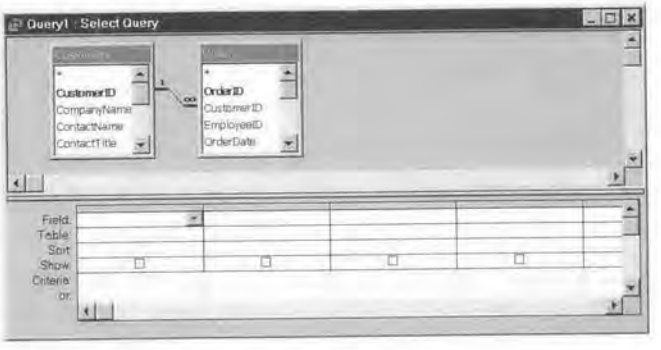


FIG. 10.2 A join between fields of two tables with a common field name, created automatically by Access.

5. To identify each order with the customer's name, select the CompanyName field of the Customers table and drag the field symbol to the Field row of the Query Design grid's first column.

6. Select the OrderID field of the Orders table and drag the field symbol to the second column's Field row. Drag the OrderDate field to the third column. Your query design appears as shown in Figure 10.3.

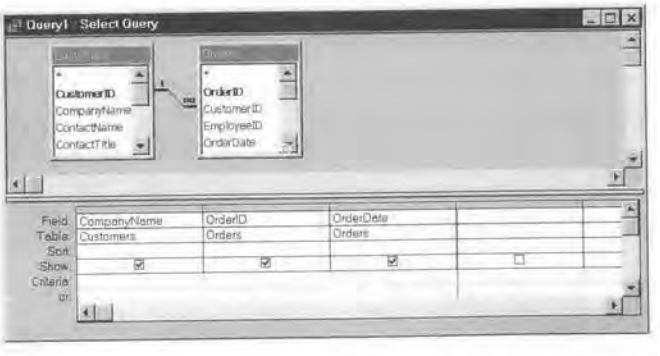


FIG. 10.3 A query to display orders placed by customers, sorted by company name and order date.

Oracle Ex. 1009, p. 348
Oracle v. DeBellis
IPR2024-00221



7. Click the Run or Query View button to display the result of the query, the `Recordset` shown in Figure 10.4. Notice that the field headers of the query result set show the captions for the table fields, which include spaces, rather than the actual field names, which don't have spaces.



FIG. 10.4 The result of the query design of Figure 10.3 that joins the Customers and Orders tables.

Specifying a Sort Order for the Query Result Set

Access displays query result sets in the order of the index on the primary-key field. If more than one column represents a primary-key field, Access sorts the query result set in left-to-right key-field column precedence. Because Company Name is the primary-key field located farthest to the left, the query result set displays all orders for a single company in order-number sequence. You can override the primary-key display order by adding a sort order. For example, if you want to see the latest orders first, you can specify a descending sort by the order date. To add this sort sequence to your query, follow these steps:



◀◀ See "Adding Indexes to Tables," p. 154



1. Click the Design View button to return to Query Design mode.
2. Place the caret in the Sort row of the Order Date column of the Query Design grid and press F4 to open the drop-down list.
3. Select Descending from the drop-down list to specify a descending sort on date—latest orders first (see Figure 10.5).



4. Click the Run button or the Query View button to display the query result set with the new sort order (see Figure 10.6).

Creating Queries from Tables with Indirect Relationships

You can create queries that return indirectly related records, such as the categories of products purchased by each customer. You must include in the queries each table that serves as a link in the chain of joins. If you are creating queries to display the categories of products purchased by each customer, for example, include each of the tables that

Oracle Ex. 1009, p. 349
Oracle v. DeBellis
IPR2024-00221

link the chain of joins between the Customers and Categories tables. This chain includes the Customers, Orders, Order Details, Products, and Categories tables. You don't need to add any fields, however, from the intermediate tables to the Query Design grid; the CompanyName and the CategoryName fields suffice.

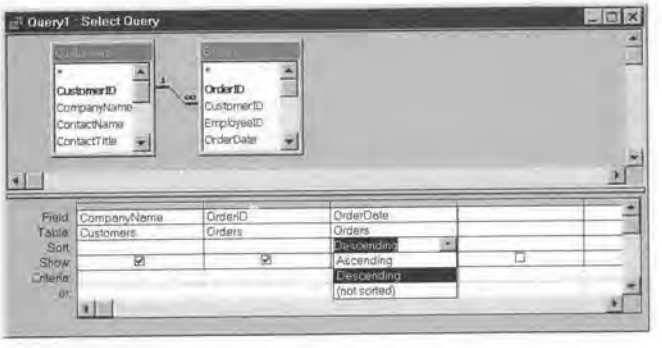


FIG. 10.5 Adding a special sort order to the query.

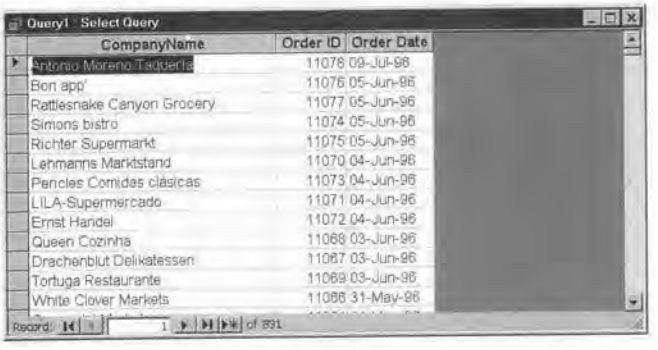


FIG. 10.6 The result of adding a descending sort on Order Date to the query.

To modify your customers and orders query so that you create a query that displays fields of indirectly related records, follow these steps:

1. In Query Design view, delete the OrderID column of the query by clicking the thin bar above the Field row to select (highlight) the entire column, then press Delete. Perform the same action for the OrderDate columns so that only the CompanyName column appears in the query.
2. Click the Show Table button of the toolbar or choose Query, Show Table and add the Order Details, Products, and Categories tables to the query, in sequence, then click the Close button of the Add Table dialog. The upper pane of Figure 10.7 shows the chain of joins that Access automatically creates between Customers and Categories based on the primary-key field of each intervening table and the identically named foreign-key field in the adjacent table.



Oracle Ex. 1009, p. 350
Oracle v. DeBellis
IPR2024-00221

As you add tables to the Query Design window, the table field lists might not appear in the upper pane. Use the upper pane's vertical scroll bar to display the "hidden" tables. You can drag the table field lists to the top of the upper pane and then rearrange the field lists to match the appearance of Figure 10.7.

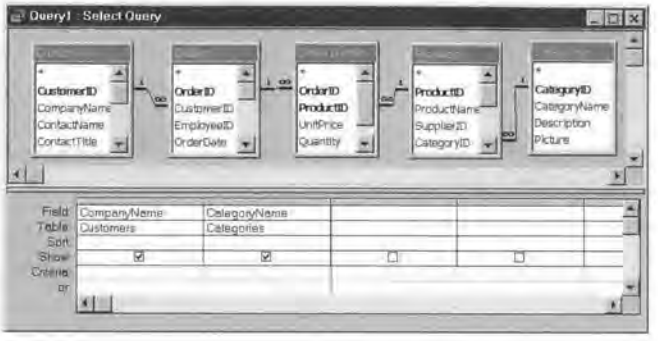


FIG. 10.7 The chain of joins required to create queries from tables that have an indirect relationship.

3. Drag the CategoryName from the Categories field list to the Field row of the grid's second column. Alternatively, you can double-click the field name to add it to the next empty column of the grid.

4. If you want to see the SQL statement that Access uses to create the query, choose View, SQL to display the SQL window, shown in Figure 10.8. The table's joins appear as `INNER JOIN...ON...` clauses. Cascaded joins use the `INNER JOIN...ON...ON...` syntax, which is explained in Chapter 24, "Working with Structured Query Language."

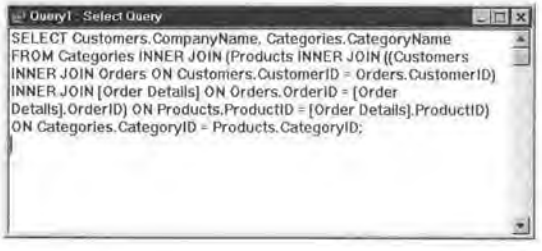


FIG. 10.8 The SQL statement that creates the query to determine which customers purchased specific categories of products.



5. Click the toolbar's Query Design View button to close the SQL window. Then click the Run button on the toolbar. The query result set shown in Figure 10.9 appears.

6. Close the query by clicking the window close box. This query is only an example, so you don't need to save it.

Oracle Ex. 1009, p. 351
Oracle v. DeBellis
IPR2024-00221



FIG. 10.9 The Customers-Categories `Recordset`, resulting from the query of Figure 10.7.

Queries made on indirectly related tables are common, especially when you want to analyze the data with SQL aggregate functions or Access's crosstab queries. For more information, see the sections "Using the SQL Aggregate Functions" and "Creating Crosstab Queries" later in this chapter.

Creating Multicolumn Equi-Joins and Selecting Unique Values

You can have more than one join between a pair of tables. You might, for example, want to create a query that returns the names of customers for which the billing and shipping addresses are the same. The billing address is the Address field of the Customers table, and the shipping address is the ShipAddress field of the Orders table. Therefore, you need to match the Customer ID fields in the two tables and `Customers.Address` with `Orders.ShipAddress`. This task requires a *multicolumn equi-join*.

To create this example of an address-matching multicolumn equi-join, follow these steps:

1. Create a new query by giving the Database window the focus, clicking the Queries tab (if necessary), and then choosing the New button. With Design View selected in the New Query dialog, click OK.



2. Add the Customers and Orders tables to the query by selecting each table in the Show Table dialog and clicking the Add button. Click Close.
3. Click and drag the Address field of the Customers table's Field List box to the ShipAddress field of the Orders table's Field List box. This creates another join, indicated by the new line between Address and ShipAddress (see Figure 10.10). The new line in the graphical QBE window has dots at both ends, indicating that the join is between a pair of fields that do not have a specified relationship, the same field name, or a primary-key index.
4. Drag the Customers table's CompanyName and Address fields to the Field row of the first and second query columns, and drop the fields. Drag the Orders table's ShipAddress field to the query's third column, and drop the field in the Field row.
5. Add an ascending sort to the CompanyName column.

Oracle Ex. 1009, p. 352
Oracle v. DeBellis
IPR2024-00221

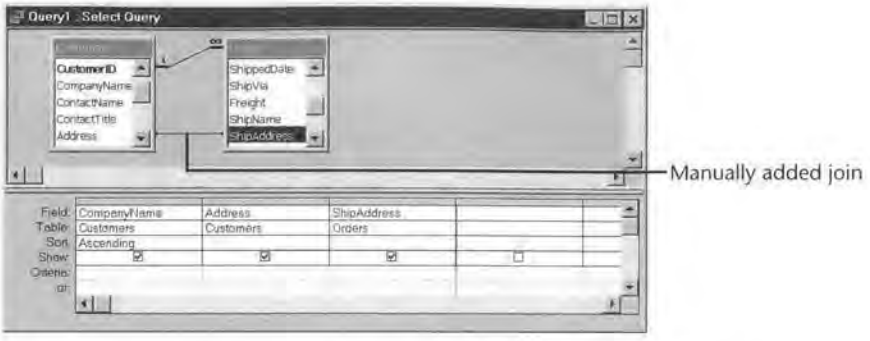


FIG. 10.10 Creating a multicolumn equi-join by dragging one field name to a field in another table.



6. Click the Run button on the toolbar. Figure 10.11 shows the query's result.



FIG. 10.11 A query result set of orders for customers who have the same billing and shipping addresses.



7. To eliminate the duplicate rows, you must use the Unique Values option of Access's Query Properties window. To display the Query Properties window, which is shown in Figure 10.12, click the Design View button, and then click the toolbar's Properties button or double-click an empty area in the Query Design window's upper pane. If the title bar of the Properties window displays Field Properties or Field List, click an empty area in the Query Design window's upper pane so that the title bar displays Query Properties. Alternatively, right-click an empty region of the upper pane and select Properties from the popup menu.

8. By default, both the `UniqueRecords` query property and the `UniqueValues` property are set **False** (`No`). Place the caret in the Unique Values text box and press F4 to open the drop-down list. Select Yes and close the list. Setting the `UniqueValues` property `True` adds the ANSI SQL reserved word `DISTINCT` to the query. Click the Properties button again to close the Properties window.

Oracle Ex. 1009, p. 353
Oracle v. DeBellis
IPR2024-00221

Tip

You can also change the property settings for the `UniqueRecords` and `UniqueValues` properties by double-clicking their text boxes in the Properties window.

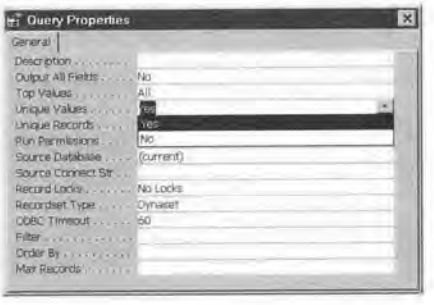


FIG. 10.12 Using the Query Properties window to display only rows with unique values.

9. Click the Run button of the toolbar. The result set no longer contains duplicate rows, as shown in Figure 10.13.



FIG. 10.13 The query result set after you remove duplicate rows.

10. Choose View, SQL to display the SQL statement (see Figure 10.14). The `DISTINCT` modifier of the `SELECT` statement causes the query to display only those records whose field values included in the query differ.

11. Click the Close Window button to close the query without saving it. You then avoid cluttering the Database window's Queries list with obsolete query examples.

Because most of the orders have the same billing and shipping addresses, a more useful query is to find the orders for which the customer's billing and shipping addresses differ. You cannot create this query with a multicolumn equi-join, however, because the `INNER JOIN` reserved word in Access SQL doesn't accept the <> operator. Adding a not-equal join uses a criterion rather than a multicolumn join, as explained in the section "Creating Not-Equal Theta Joins with Criteria" later in this chapter.

Oracle Ex. 1009, p. 354
Oracle v. DeBellis
IPR2024-00221



FIG. 10.14 The SQL statement that results in the query result set of Figure 10.13.

Troubleshooting

When I run my query, an Enter Parameter Value dialog appears that asks me to enter a value. I didn't specify a parameter for the query.

The Enter Parameter Value dialog appears when the Jet engine's query parser cannot identify an object specified in the query or evaluate an expression. Usually, the Enter Parameter Value dialog appears because of a typographic error. Intentionally creating parameter queries is the subject of the section "Designing Parameter Queries" later in this chapter.

Using Lookup Fields in Tables

Access 97's lookup feature for table fields lets you substitute combo (drop-down list) boxes or list boxes for conventional field text boxes. The lookup feature lets you provide a list of acceptable values for a particular field. When you select the value from the list, the value automatically is entered in the field of the current record. You can specify either of the following two types of lookup field:

- In a field that contains foreign-key values, a list of values from one or more fields of a related base table is provided. As an example, the Orders table of Northwind.mdb has two foreign key fields: CustomerID and EmployeeID. The lookup feature of the CustomerID field displays in a combo box the CompanyName field value from the Customers table. The EmployeeID field displays the LastName and FirstName fields of the Employees table, separated by a comma and space (see Figure 10.15). The foreign key lookup feature is implemented by a simple Access SQL select query, `SELECT DISTINCTROW [CustomerID],[CompanyName] FROM [Customers] ORDER BY [CompanyName];`, in the case of the CustomerID field.
- With any field except a single primary-key field, a list of fixed values from which to select is provided.

Oracle Ex. 1009, p. 355
Oracle v. DeBellis
IPR2024-00221

Note

Lookup is a feature of a field, not a field data type. The field data type is that of the content of the field, such as Number (Long Integer) for the EmployeeID field of the Orders table. The lookup feature is implemented by a special set of Access-specific properties that are stored with the common (Data Access Object, or DAO) field properties of the table. (Lookup properties are included in the `Properties` collection of the `Field` object of the `TableDef` object for the table. These objects are described in Chapter 29, "Understanding the Data Access Object Class.")



FIG. 10.15 A lookup field's combo box and value list.

You can add a new lookup field in either Design view or Table Datasheet view; however, you only can add the lookup feature to an existing field in Design view. In Datasheet view, only the combo box control is displayed, even if you specify a list box control. You can display a combo box or a list box on a form that is bound to a table with lookup fields. In practice, the drop-down list (a combo box with the Limit to List property set to Yes) is the most common type of lookup field control. The following sections describe how to add foreign key and fixed list lookup features to table fields.

Adding a Foreign Key Drop-Down List with the Lookup Wizard

The Personnel Actions table you created in earlier chapters of this book is a candidate for a lookup field that uses a foreign-key drop-down list. Follow these steps to use the Lookup Wizard to change the paID field of the Personnel Actions table to a lookup field:

1. In the Database window, select the Personnel Actions table, and press Ctrl+C to copy the table to the Clipboard.



2. Press Ctrl+V to display the Paste Table As dialog. Enter a name for the copy, such as **tblLookup**, and click the OK button to create the copy.
3. Open the table copy in Design view and select the paInitiatedBy field. Click the Lookup tab to display the current lookup properties; a text box control has no lookup properties. Open the Data Type drop-down list and select Lookup Wizard (see Figure 10.16). The first dialog of the Lookup Wizard appears.



Oracle Ex. 1009, p. 356
Oracle v. DeBellis
IPR2024-00221

4. You want the field to look up values in another table (Employees), so accept the first (default) option (see Figure 10.17). Click the Next button to display the Lookup Wizard's second dialog.
5. With the View Tables option enabled, select the Employees base table to which the paInitiatedBy field is related (see Figure 10.18). Click the Next button to display the third dialog.
6. Click the > button three times to add the EmployeeID, LastName, and FirstName fields to your lookup list (see Figure 10.19). You must include the base table key field that is related to your foreign key field. Click the Next button for the fourth dialog.

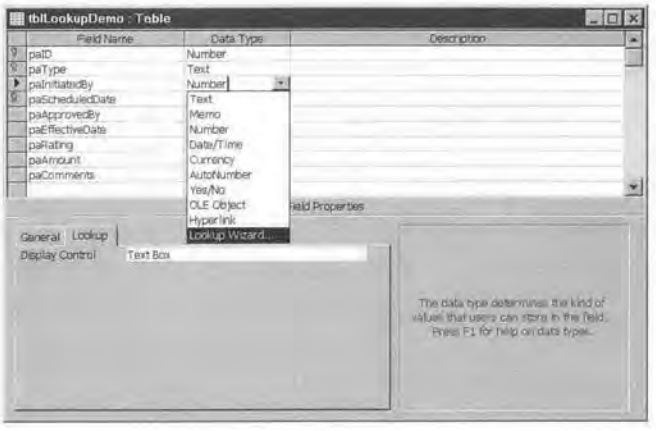


FIG. 10.16 Selecting the Lookup Wizard to add the lookup feature to a field.

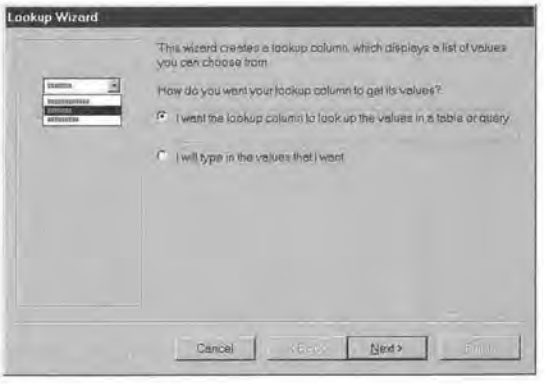


FIG. 10.17 Selecting between a foreign-key and fixed-list lookup in the first dialog of the Lookup Wizard.

Oracle Ex. 1009, p. 357
Oracle v. DeBellis
IPR2024-00221

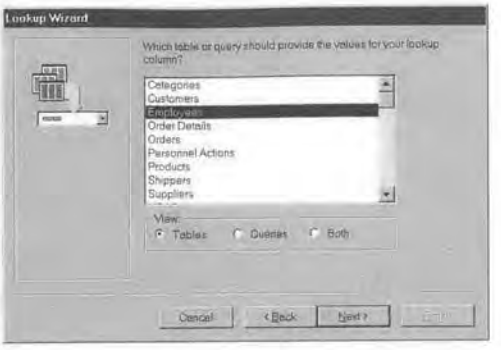


FIG. 10.18 Choosing the base table or query of a foreign-key lookup field.

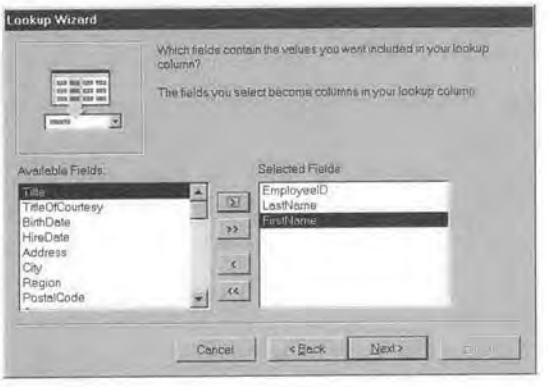


FIG. 10.19 Selecting the fields to include in your lookup list.

7. Adjust the widths of the columns to display the first and last names without excessive trailing "white space." The wizard determines that EmployeeID is the key column and recommends hiding the key column (see Figure 10.20). Click Next to display the fifth and final dialog.

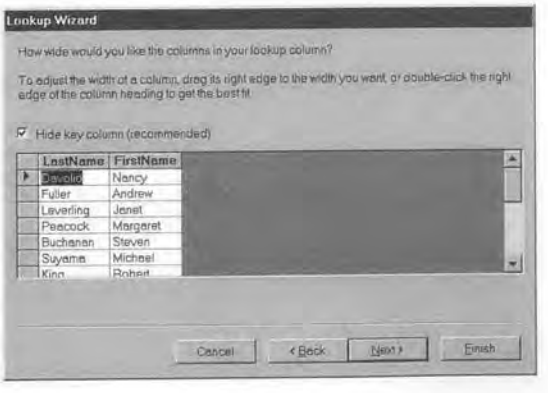


FIG. 10.20 Adjusting column widths of the lookup list and hiding the key column.

Oracle Ex. 1009, p. 358
Oracle v. DeBellis
IPR2024-00221

8. Accept the default "label" for the lookup field in the text box (see Figure 10.21). (If you change the default value, you change the field name, not the caption.)

9. Click the Finish button to complete the wizard's work. Click OK when the wizard asks whether you'd like to save the table design. Your new lookup field properties appear as shown in Figure 10.22. The Access SQL statement created by the wizard is `SELECT DISTINCTROW [Employees].[EmployeeID], [Employees].[LastName], [Employees].[FirstName] FROM [Employees];`.



10. Click the Table View button to display the table datasheet. Only the first visible column of the list appears in the Initiated By column. Adjust the width of the Initiated By column to the width of the drop-down list, about 1.5 inches. With the caret in the Initiated By column, open the drop-down list to display the wizard's work (see Figure 10.23).



FIG. 10.21 Specifying the caption for the lookup field.

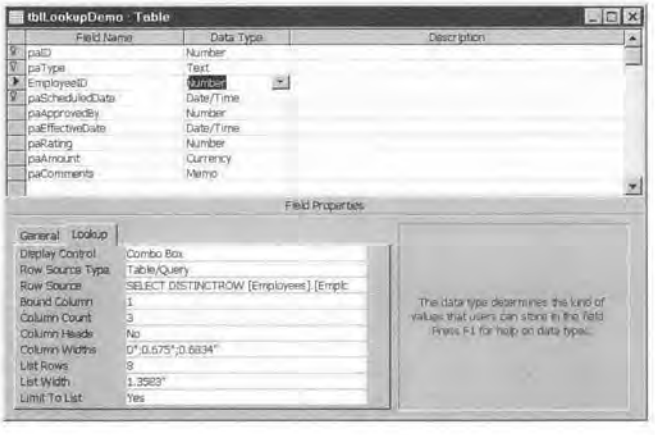


FIG. 10.22 Lookup properties added to the paInitiatedBy field.

Oracle Ex. 1009, p. 359
Oracle v. DeBellis
IPR2024-00221

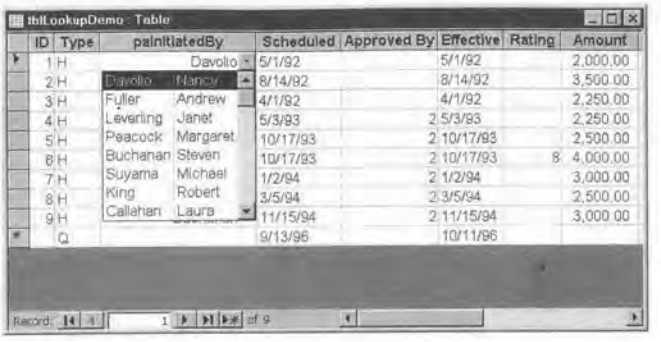


FIG. 10.23 The drop-down lookup list created by the Lookup Wizard.

11. Return to Design view, select the Row Source property of the paInitiatedBy field, and click the Build button to display the Row Source SQL statement in Query Design view (see Figure 10.24), then close the Query Design window.

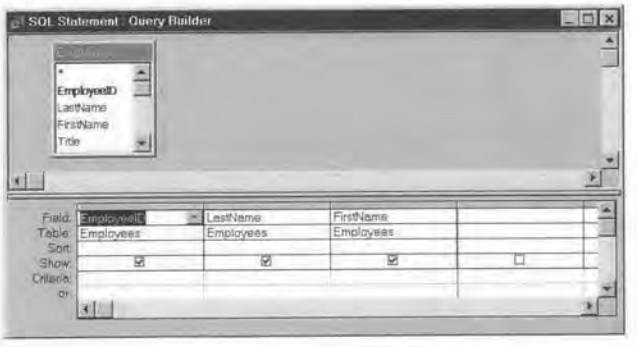


FIG. 10.24 The Query Design view of the SQL statement of the Row Source property in Query Design view.

> **Note**
>
> The properties of the combo box control created by the wizard are described in the "Using List Boxes and Combo Boxes" section of Chapter 13, "Designing Custom Multitable Forms." You can alter the lookup properties of a field to customize the basic work done for you by the wizard. The wizard's entries are quite adequate for most Access applications.

Adding a Fixed Value Lookup List to a Table

You add the alternative lookup feature—a fixed list of values—using the Lookup Wizard in much the same way as you created the foreign-key lookup list in the preceding section. To add a fixed-list lookup feature to the paType field of your copy of the Personnel Actions table, follow these steps:

1. Select the paType field, open the Data Type list, and select Lookup Wizard to launch the wizard.

Oracle Ex. 1009, p. 360
Oracle v. DeBellis
IPR2024-00221

2. In the first Lookup Wizard dialog, select the I Will Type in the Values That I Want option, and click the Next button.
3. In the second Lookup Wizard dialog, type **2** in the Number of Columns text box, and press the Tab key to create the second list column.
4. Enter **H**, **Hired**; **Q**, **Quarterly Review**; **Y**, **Yearly Review**; and **S**, **Salary Adjustment** in the Col1 and Col2 columns of four rows. Adjust the width of the columns to suit the entries (see Figure 10.25). Click the Next button to display the wizard's third dialog.
5. The paType field uses single-character abbreviations for the type of personnel actions, so select Col1 as the "field that uniquely identifies the row," as shown in Figure 10.26. (The paType field does not uniquely identify the row; Col1 contains the single-character value that you want to insert into the field.) Click the Next button to display the fourth and final wizard dialog.

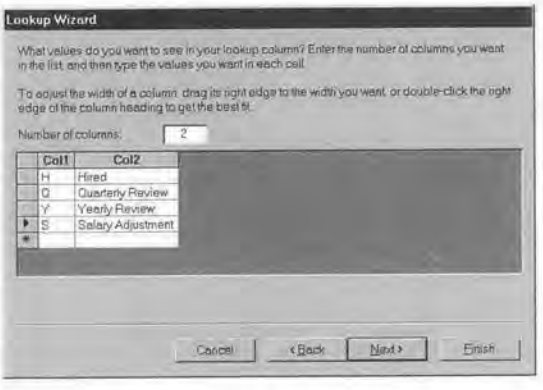


FIG. 10.25 Adding the lookup list values in the Lookup Wizard's second dialog.



FIG. 10.26 Selecting the column that contains the value to insert into the field.

Oracle Ex. 1009, p. 361
Oracle v. DeBellis
IPR2024-00221

6. Accept the default "label" for your column and click the Finish button. The lookup properties for the paType field appear as shown in Figure 10.27. The Row Source Type is Value List. The Row Source contains the following values: `"H";"Hired";"Q";"Quarterly Review";"Y";"Yearly Review";"S";"Salary Adjustment"`.

7. Click the Table View button and save the changes to your table. Increase the width of the Type column to about 1.5 inches, place the caret in the Type column, and open the fixed value list to check the wizard's work (see Figure 10.28).



8. If you don't want the abbreviation to appear in the drop-down list, change the first entry of the Column Widths property value to **0**.

9. If you want to remove the lookup feature from a field, select the field, click the Lookup tab, and choose Text Box from the drop-down Display Control list.

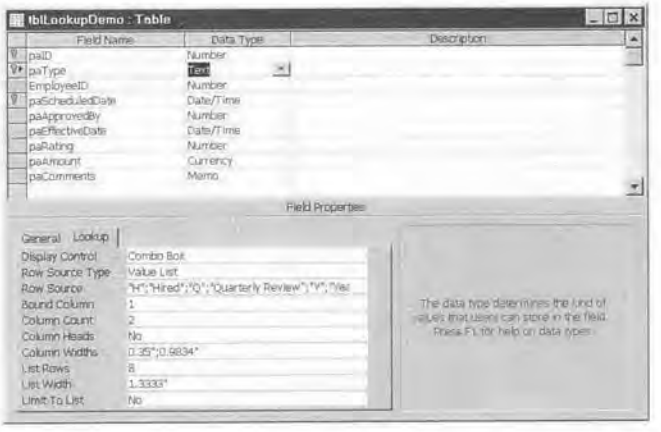


FIG. 10.27 Selecting the column that contains the value to insert into the field.

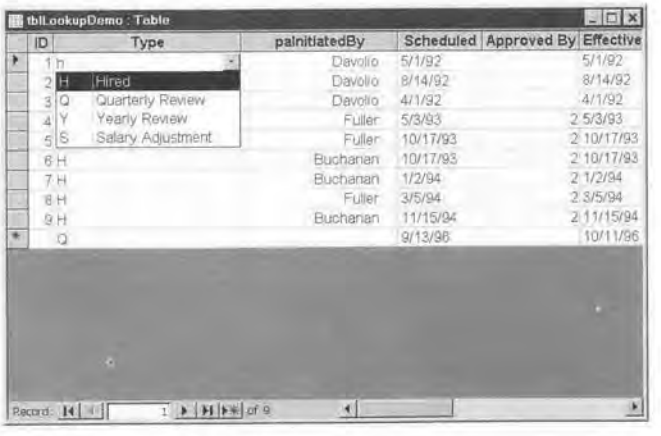


FIG. 10.28 The fixed value list created by the Lookup Wizard.

Oracle Ex. 1009, p. 362
Oracle v. DeBellis
IPR2024-00221

Note

The lookup feature has generated controversy among seasoned database developers. Relational database purists object to the principle of modifying the properties of tables with embedded queries. Another objection to the use of foreign-key, drop-down lists is that it is easy for uninitiated users to inadvertently change data in a table after opening the list. Access 97's lookup feature, however, is a useful tool, especially for new database users.

Outer, Self, and Theta Joins

The preceding sections of this chapter described the equi-join or, in the parlance of SQL-92, an inner join. Inner joins are the most common type of join in database applications. Access also lets you create three other joins: outer, self, and theta. The following sections describe these three less-common types of joins.

Creating Outer Joins

Outer joins allow you to display the fields of all records in a table participating in a query, regardless of whether corresponding records exist in the joined table. With Access, you can choose between left and right outer joins.

When diagramming database structures (a subject of Chapter 23, "Exploring Relational Database Design and Implementation"), you traditionally would draw the primary *one* table to the left of the secondary *many* table. A left outer join (`LEFT JOIN` or `*=` in SQL) query in Access, therefore, displays all of the records in the table with the unique primary key, regardless of whether matching records exist in the *many* table. Conversely, a right outer join (`RIGHT JOIN` or `=*`) query displays all of the records in the *many* table, regardless of a record's existence in the primary table. Records in the *many* table without corresponding records in the *one* table usually, but not necessarily, are orphan records; these kinds of records may have a many-to-one relationship to another table.

To practice creating a left outer join to detect whether records are missing for an employee in the Personnel Actions table, follow these steps:

New

1. Open a new query and add the Employees and Personnel Actions tables.
2. Drag the EmployeeID field symbol to the paID field of Personnel Actions to create an equi-join between these fields if Access didn't create the join. (Access automatically creates the join if you established a relationship between these two fields when you created the Personnel Actions table in Chapter 4, "Working with Access Databases and Tables.")
3. Select and drag the LastName and FirstName fields of the Employees table to columns 1 and 2 of the Query Design grid. Select and drag the paType and paScheduledDate fields of the Personnel Actions table to columns 3 and 4.
4. Click the line joining EmployeeID with paID to select it, as shown in Figure 10.29. The thickness of the center part of the line increases to indicate the selection. (In Figure 10.29, the two Field List boxes are separated so that the thin section of the join line is apparent.)

Oracle Ex. 1009, p. 363
Oracle v. DeBellis
IPR2024-00221

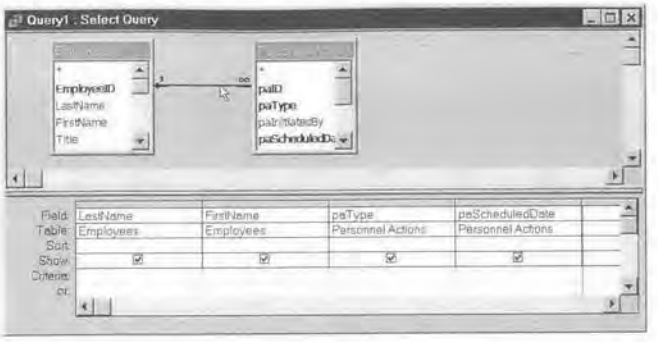


FIG. 10.29 Selecting a join to change its property from an inner to a left or right outer join.

5. Choose View, Join Properties. (The Join Properties command is active only after you select an individual join with a mouse click.) You also can double-click the thin section of the join line. (Double-clicking either of the line's thick sections displays the Query Properties window.) The Join Properties dialog in Figure 10.30 appears. Type 1 is a conventional inner join, Type 2 is a left join, and Type 3 is a right join.

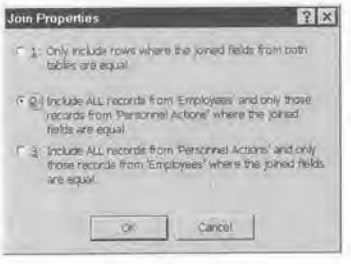


FIG. 10.30 The Join Properties dialog for choosing inner, left, or right joins.

6. Select a Type 2 join—a left join—by choosing 2. Click OK to close the dialog.

   Notice that Access adds an arrowhead to the line that joins EmployeeID and paID. The direction of the arrow, left to right, indicates that you have created a left join between the tables.

7. Click the Run button of the toolbar to display the result of the left join query. In Figure 10.31, three employees without a record in the Personnel Actions table appear in the result table's last rows. Your query result set may differ, depending on the number of entries that you made when creating the Personnel Actions table. (If all employees show a personnel action, open the Personnel Actions table, delete the entries for a few employees, and then rerun the query.)



8. Close, but don't save, the query.

Oracle Ex. 1009, p. 364
Oracle v. DeBellis
IPR2024-00221

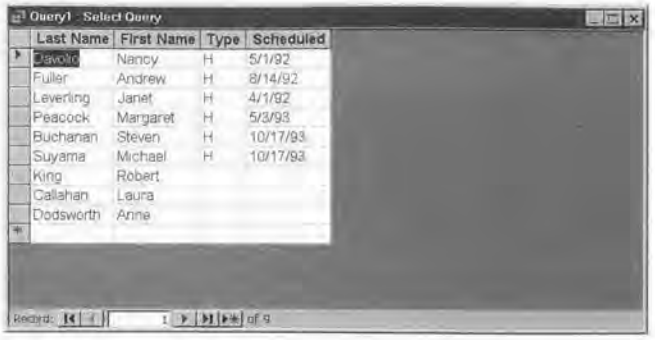


FIG. 10.31 The result of creating a left join between the ID fields of the Employees and Personnel Actions tables.

If you could add a personnel action for a nonexistent EmployeeID (the validation rule that you added in Chapter 9, "Understanding Operators and Expressions in Access," prevents you from doing so), a right join would show the invalid entry with blank employee name fields.

Creating Self-Joins

Self-joins relate values in a single table. Creating a self-join requires that you add to the query a copy of the table and then add a join between the related fields. An example of self-join use is to determine whether supervisors have approved personnel actions that they initiated, which is prohibited by the imaginary personnel manual for Northwind Traders.

To create this kind of self-join for the Personnel Actions table, follow these steps:



1. Open a new query and add the Personnel Actions table.
2. Add to the query another copy of the Personnel Actions table by clicking the Add button again. Access names the copy Personnel Actions_1. Close the Show Tables dialog.
3. Drag the original table's paInitiatedBy field to the copied table's paApprovedBy field. The join appears as shown in the upper pane of Figure 10.32.
4. Drag the paID and paInitiatedBy fields of the original table, and the paApprovedBy and paType fields of the copy of the Personnel Actions table, to the Field row of columns 1–4, respectively, of the Query Design grid.



5. With self-joins, you must specify that only unique values are included. Click the Properties button on the toolbar or double-click an empty area in the Query Design window's upper pane, and set the value of the Query Properties window's `UniqueValues` property to `Yes`. Click the Properties button again to close the Query Properties window.

Oracle Ex. 1009, p. 365
Oracle v. DeBellis
IPR2024-00221

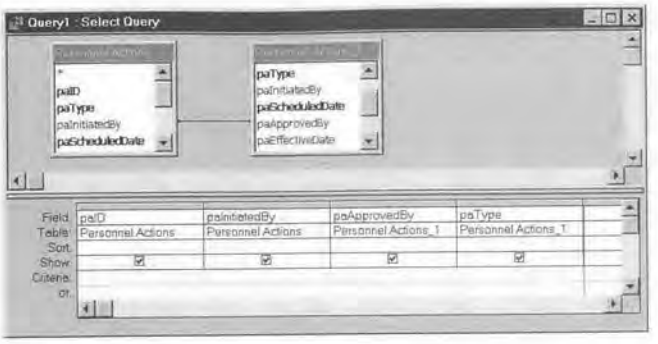


FIG. 10.32 Designing the query for a self-join on the Personnel Actions table.

6. Click the Run button of the toolbar to display the records in which the same employee initiated and approved a personnel action, as shown in Figure 10.33. In this case, EmployeeID 2 (Mr. Fuller) is a vice president and can override personnel policy. (Your results may differ, depending on the number of entries that you made in the Personnel Actions table.)

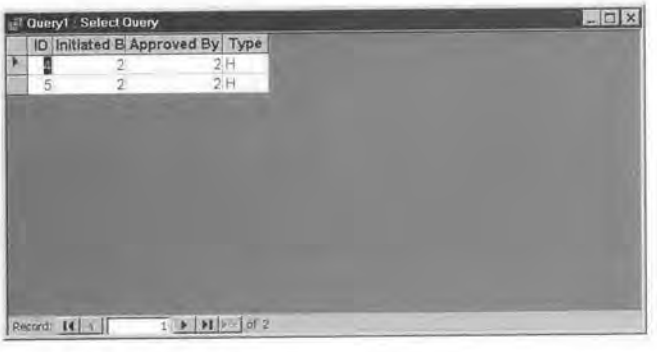


FIG. 10.33 The result of a self-join that tests for supervisors approving personnel actions that they initiated.

In this example, you can add the Employees table to the query in order to display the employee name. Adding the Employees table creates an additional join between the original paID field of the Personnel Actions table and the EmployeeID field of the Employees table. You then must drag the LastName field to the Query Design grid's fifth column. Because this join includes a primary-key field, EmployeeID, the default `DISTINCT` process yields unique values. To verify that the values are unique, click the Properties button of the toolbar, or double-click an empty area in the Query Design window's upper pane. Set the value of the `UniqueValues` property to `Yes` and then rerun the query.

▶▶ See "Maintaining Referential Integrity," p. 834

Oracle Ex. 1009, p. 366
Oracle v. DeBellis
IPR2024-00221

Full-fledged relational database applications seldom use self-joins because validation criteria and enforcement of referential integrity can (and should) eliminate the types of problems that self-joins can detect.

Creating Not-Equal Theta Joins with Criteria

Most joins are based on fields with equal values, but sometimes you need to create a join on unequal fields. Joins that you create with graphical QBE in Access are restricted to conventional equi-joins and outer joins. You can create the equivalent of a not-equal theta join by applying a criterion to one of the two fields you want to test for not-equal values.

Finding customers that have different billing and shipping addresses, as mentioned previously, is an example in which a not-equal theta join is useful. To create the equivalent of this join, follow these steps:

New

1. Create a new query and add the Customers and Orders tables.
2. Select the Customers table's CompanyName and Address fields and the Orders table's ShipAddress field. Drag them to the Query Design grid's first three columns.
3. Type <> **Customers.Address** in the Criteria row of the ShipAddress column. (Access automatically adds square brackets surrounding table and field names, regardless of whether the names include spaces or other punctuation.) The Query Design window appears as shown in Figure 10.34. This criterion adds the `WHERE Orders.ShipAddress <> [Customers].[Address]` clause to the SQL `SELECT` statement shown in Figure 10.35.

   Typing <> **Orders.ShipAddress** in the Address column gives an equivalent result. This criterion adds a `WHERE Customers.Address <> Orders.[ShipAddress]` clause to the SQL `SELECT` statement.

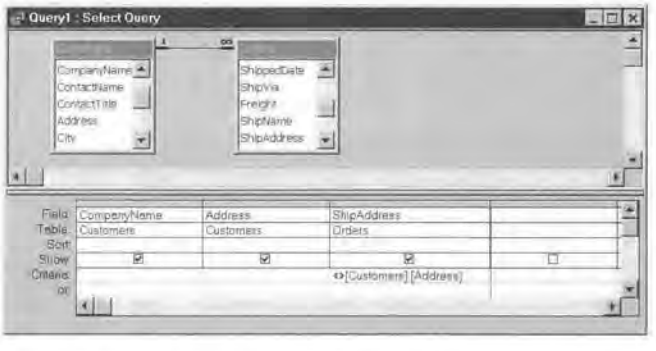


FIG. 10.34 Designing the query for a not-equal theta join.

4. Click the toolbar's Properties button or double-click an empty area in the Query Design window's upper pane to open the Query Properties window and set the value of the Unique Values property to `Yes`.

Oracle Ex. 1009, p. 367
Oracle v. DeBellis
IPR2024-00221

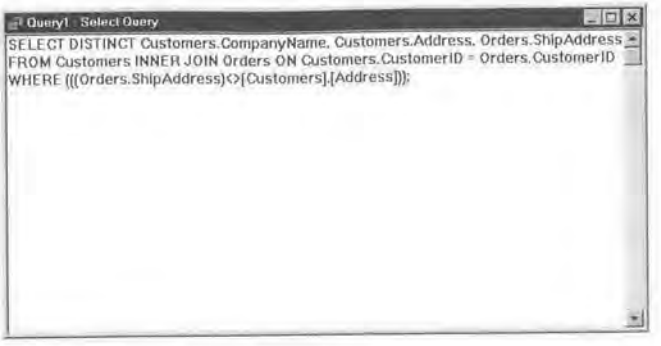


FIG. 10.35 The SQL statement for a not-equal theta join.

5. Run the query. Only the records for customers that placed orders with different billing and shipping addresses appear, as shown in Figure 10.36.



6. Click the Close Window button and save your query, if desired.



FIG. 10.36 The result of a not-equal theta join designed to identify different billing and shipping addresses.

Updating Table Data with Queries

Many of the queries that you create with the Unique Records property set to `Yes` are updatable because you use Access SQL's `DISTINCTROW` modifier to create them. (Setting the Unique Records property to Yes causes Access to add the `DISTINCTROW` modifier to the query's SQL statement.) These queries create `Recordset` objects of the updatable `Dynaset` type. You cannot update a query unless you see the tentative (blank) append record (with the asterisk in the select button) at the end of the query result table. Queries that you create with the Unique Values property set to `Yes` create `Recordset` objects of the `Snapshot` type by substituting ANSI SQL's `DISTINCT` modifier. You cannot edit, add new records to, or otherwise update a `Recordset` object of the `Snapshot` type. The next few sections describe the conditions under which you can update a record of a table included in a query. The following sections also discuss how to use the Output Field Properties window to format a query data display and edits.

Oracle Ex. 1009, p. 368
Oracle v. DeBellis
IPR2024-00221

Note

You cannot set both the Unique Values and Unique Records properties to Yes—these choices are mutually exclusive. Setting Unique Values to Yes causes Access to include the `DISTINCT` modifier in the SQL statement for the query, resulting in a `Recordset` that cannot be updated. Setting the Unique Records property to Yes causes Access to include the `DISTINCTROW` modifier in the SQL statement, resulting in a Recordset that *can* be updated. In Access 97, the default setting of both the Unique Values and Unique Records properties is No.

Note

`Dynaset` and `Snapshot` are subtypes of Access 97's `Recordset` object. Microsoft states that the use of the reserved words `Dynaset` and `Snapshot` as object names is supported in Access 97 only for compatibility with Access 1.x. Although the distinctive terminology of `Dynaset` and `Snapshot` makes it unlikely that these two terms will disappear from common use among Access developers, future versions of Access might not support `Dynaset` and `Snapshot` objects and might instead require you to specify `Recordset` objects of different types in Access VBA.

Characteristics That Determine Whether You Can Update a Query

Adding new records to tables or updating existing data in tables included in a query is a definite advantage in some circumstances. Correcting data errors that appear when you run the query is especially tempting. Unfortunately, you cannot append or update records in most of the queries that you create. The following properties of a query prevent you from appending and updating records:

- The Unique Values property is set to Yes in the Query Properties window.
- Self-joins are used in the query.
- SQL aggregate functions, such as `Sum()`, are employed in the query. Crosstab queries, for example, use SQL aggregate functions.
- No primary-key fields with a unique (No Duplicates) index exist for the *one* table in a one-to-many relationship.

When designing a query to use as the basis of a form for data entry or editing, make sure that none of the preceding properties apply to the query.

If none of the preceding properties apply to the query or all tables within the query, you can append records to and update fields of queries in the following:

- A single-table query
- Both tables in a one-to-one relationship
- The *many* table in a one-to-many relationship
- The *one* table in a one-to-many relationship if none of the fields of the *many* table appear in the query

Oracle Ex. 1009, p. 369
Oracle v. DeBellis
IPR2024-00221

Updating the *one* table in a one-to-many query is a special case in Access. To enable updates to this table, follow these steps:

1. Add to the query the primary-key field or fields of the *one* table and additional fields to update.
2. Add the field or fields of the *many* table that correspond to the key field or fields of the *one* table; this step is required to select the appropriate records for updating.
3. Add the criteria to select the records for updating to the fields chosen in step 2.
4. Click the Show box so that the *many* table field(s) does not appear in the query.

After following these steps, you can edit the non-key fields of the *one* table. You cannot, however, alter the values of key fields that have relationships with records in the *many* table. Such a modification violates referential integrity. You also cannot update a calculated column of a query; tables cannot include calculated values.

Note

By adding lookup fields to tables, you often can avoid writing one-to-many queries and precisely following the preceding rules to make such queries updatable. For example, the Orders table, which includes three lookup fields (CustomerID, EmployeeID, and ShipVia), is updatable. If you want to allow updates in Datasheet view (called *browse updating*), using lookup fields is a better approach than creating an updatable query. Most database developers consider simple browse updating to be a poor practice because of the potential for inadvertent data-entry errors.

Formatting Data with the Output Field Properties Window

The display format of data in queries is inherited from the format of the data in the tables that underlie the query. You can override the table format by using the `Format(ColumnName, FormatString)` function described in Chapter 9, "Understanding Operators and Expressions in Access," to create a calculated field.



Access 97 provides an easier method: it adds a Field Properties window that you can use to format the display of query data. You also can create an input mask to aid in updating the query data. To open the Field Properties window, place the caret in the Field cell of the query column that you want to format, and then click the Properties button of the toolbar or double-click an empty area in the upper pane of the Query Design window. Figure 10.37 shows the Field Properties window for the Order Date column of a simple one-to-many query that was created from the Customers and Orders tables.

- *Description*. Enables you to enter the text to appear in the status bar when the user selects the field in Datasheet view.
- *Format*. Enables you to control the appearance of the data in Datasheet view, such as Medium Date.

Oracle Ex. 1009, p. 370
Oracle v. DeBellis
IPR2024-00221

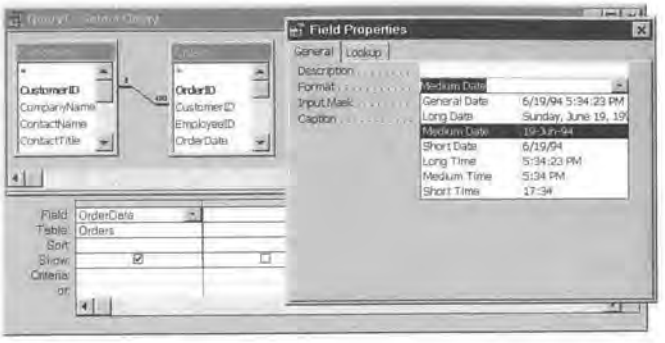


FIG. 10.37 Changing the display format for a query column of the Date/Time field data type.

The Field Properties window displays the following subset of the properties that apply to a table's fields:

- *Input Mask*. Enables you to establish the format for entering data, such as `90/90/00`. (To create an input mask that is appropriate for the field data type, click the ellipsis button to open the Input Mask Wizard.)
- *Caption*. Enables you to change the query column heading, such as Received, for the Order Date column.

Each of the preceding query properties follows the rules described in Chapter 4, "Working with Access Databases and Tables," for setting table field properties. Adding a value (Received) for the Caption property is the equivalent of adding a column alias by typing **Received**: as a prefix in the Order Date column's Field cell. The value of the Input Mask property need not correspond to the value of the Format property. For example, the Received (OrderDate) field in Figure 10.38, which shows the effect of setting the property values shown in the preceding list, has a Medium Date display format and an input mask for updating in Short Date format.

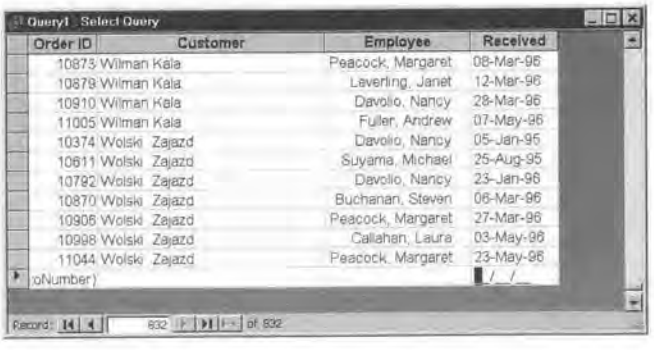


FIG. 10.38 Adding a new order record with an Order Date input mask.

Oracle Ex. 1009, p. 371
Oracle v. DeBellis
IPR2024-00221

Troubleshooting

I can't create an updatable one-to-many query with my attached dBASE tables, despite the fact that my query only displays fields from the many side of the relationship.

You must specify (or create) primary-key indexes for each dBASE table that participates in the query. The field or fields that you choose must uniquely identify a record; the index doesn't allow duplicate values. Delete the attachment to the dBASE tables, and then reattach the table with the primary-key indexes. Make sure that you specify which index is the primary-key index in the Select Unique Record Identifier dialog that appears after you attach each table.

Also, make sure that you don't include the field of the *many*-side table on which the join is created in the query. If you add the joined field to the field list, your query is not updatable.

Making All Fields of Tables Accessible

Most queries that you create include only the fields you specifically choose. To choose these fields, you either select them from or type them into the drop-down combo list in the Query Design grid's Field row, or you drag the field names from the field lists to the appropriate cells in the Field row. You can, however, include all fields of a table in a query. Access provides three methods, which are covered in the following sections.

Using the Field List Title Bar to Add All Fields of a Table

One way to include all fields of a table in a query is to use the field list title bar or asterisk. To use this method in your query to include all fields, together with their field name headers, follow these steps:

New

1. Open a new query and add the tables required for your query.
2. Double-click the field list title bar of the table for which you want to include all fields. This selects all of the fields in the field list.
3. Click and drag a field name to the Query Design grid's Field cell, and drop the field name where you want the first field to appear. Figure 10.39 shows an example of the result of the preceding steps for the Customers table.

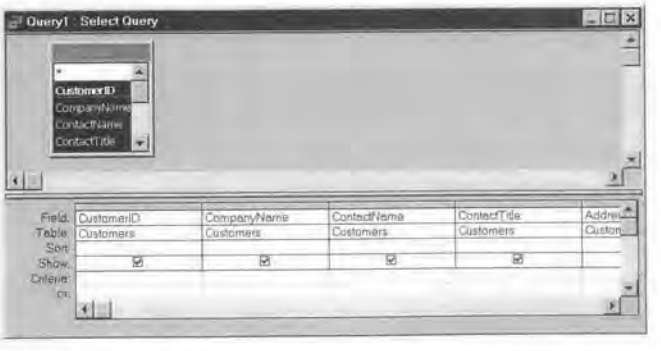


FIG. 10.39 Adding all fields of the Customers table to a query by double-clicking the header of the Customers field list and dragging the fields to the Query Design grid.

Oracle Ex. 1009, p. 372
Oracle v. DeBellis
IPR2024-00221

Adding all fields to the query by this method creates a SQL statement that is the equivalent of the following:

```
SELECT TableName.FirstField,
       TableName.SecondField, ... TableName.LastField
  FROM TableName
```

Using the Asterisk to Add All Fields Without Field Names

To include in the query all fields of the table without displaying their field names, click and drag the asterisk to the first Field cell of the Query Design grid, and drop the asterisk where you want all of the fields to appear in the query result table. The asterisk column is equal to the Access SQL statement `SELECT * FROM TableName`.

You cannot sort on a column with an asterisk in its Field cell, nor can you establish criteria on such a column. If you choose the asterisk approach, you can sort or apply criteria to one or more fields in the table by following this technique:

1. After you add the asterisk to the Field cell, drag the name of the field that you want to sort or to which you want to apply a criterion. Drop that field name in the adjacent column's Field row.
2. Add the sort specification to the Sort cell or add the criterion to the Criteria cell.
3. Click the Show box, which removes the check mark, so that the field doesn't appear twice in the query. Figure 10.40 shows the resulting query design with the Customers table.

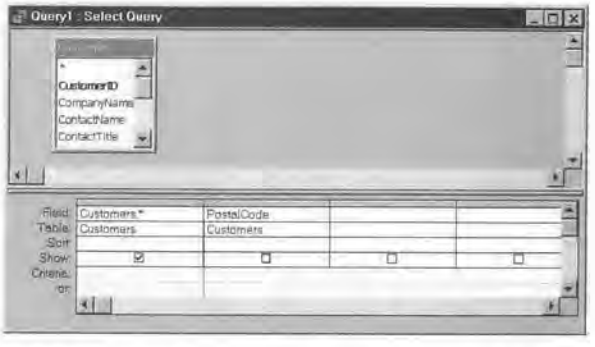


FIG. 10.40 Adding all fields of the Customers table with the asterisk field and providing a hidden column for sorting by PostalCode.

You can use this method to add to the query as many columns from the asterisk table as you need. The field on which you sort the data is added to the SQL statement as an `ORDER BY TableName.FieldName` clause, and a criterion is added in a `WHERE CriterionExpression` clause.

Selecting All Fields with the Output All Fields Property

Usually, only the fields whose names appear in the query are available for updating in forms or including in reports. All other fields are excluded from the result set. You can

Oracle Ex. 1009, p. 373
Oracle v. DeBellis
IPR2024-00221

make all fields of every table used in the query available to the forms and reports that you create—even though the query design does not include the fields by name—by setting the Output All Fields property to Yes. To use the Output All Fields property to make all table fields available, follow these steps:

1. Open a new query and add the table(s) that participate(s) in the query.



2. Click the Properties button of the toolbar or double-click an empty area in the Query Design window's upper pane to display the Query Properties window.



3. In the Query Properties window, place the caret in the Output All Fields text box, press F4 to open the drop-down list, and change the default value—No—to Yes.
4. If the Unique Values Only text box displays Yes, you cannot update fields.
5. Click the Properties button to close the Query Properties window.



Turning on the Output All Fields option adds an all-fields asterisk to the list of specified fields in the SQL statement, as in the following example:

```
SELECT Customers.[CompanyName],
    Categories.[CategoryName], *
```

When you include all fields in your queries, you might find that running the query takes longer, especially with queries that create many rows in the result set.

Making Calculations on Multiple Records

One of QBE's most powerful capabilities is that of obtaining summary information almost instantly from specified sets of records in tables. Summarized information from databases is the basis for virtually all management information systems (MIS). Such systems usually answer questions such as, "What are our sales to date for this month?" or "How did last month's sales compare with the same month last year?" To answer these questions, you must create queries that make calculations on field values from all or selected sets of records in a table. To make calculations on table values, you must create a query that uses the table and employs Access's SQL aggregate functions to perform the calculations.

Using the SQL Aggregate Functions

Summary calculations on fields of tables included in query result tables use the SQL aggregate functions listed in Table 10.1. These are called *aggregate functions* because they apply to groups (aggregations) of data cells. The SQL aggregate functions satisfy the requirements of most queries needed for business applications. You can write special user-defined functions with Access VBA code to apply more sophisticated statistical, scientific, or engineering aggregate functions to your data.

Oracle Ex. 1009, p. 374
Oracle v. DeBellis
IPR2024-00221

Table 10.1 SQL Aggregate Functions

Function	Description	Field Types
`Avg()`	Average of values in a field	All types except Text, Memo, and OLE Object
`Count()`	Number of Not Null values in a field	All field types
`First()`	Value of a field of the first record	All field types
`Last()`	Value of a field of the last record	All field types
`Max()`	Greatest value in a field	All types except Text, Memo, and OLE Object
`Min()`	Least value in a field	All types except Text, Memo, and OLE Object
`StDev()`, `StDevP()`	Statistical standard deviation of values in a field	All types except Text, Memo, and OLE Object
`Sum()`	Total of values in a field	All types except Text, Memo, and OLE Object
`Var()`, `VarP()`	Statistical variation of values in a field	All types except Text, Memo, and OLE Object

`StDev()` and `Var()` evaluate population samples. You can choose these functions from the drop-down list in the Query Design grid's Total row. (The Total row appears when you click the Totals button of the toolbar or choose View, Totals.) `StDevP()` and `VarP()` evaluate populations and must be entered as expressions. If you're familiar with statistical principles, you recognize the difference in the calculation methods of standard deviation and variance for populations and samples of populations. The following section explains the method of choosing the SQL aggregate function for the column of a query.

Note

ANSI SQL and most SQL (client/server) databases support the equivalent of Access SQL's `Avg()`, `Count()`, `First()`, `Last()`, `Max()`, `Min()`, and `Sum()` aggregate functions as `AVG()`, `COUNT()`, `FIRST()`, `LAST()`, `MAX()`, `MIN()`, and `SUM()`, respectively. ANSI SQL and few, if any, SQL databases provide equivalents of the `StdDev()`, `StdDevP()`, `Var()`, and `VarP()` functions.

Making Calculations Based on All Records of a Table

Managers, especially sales and marketing managers, are most often concerned with information about orders received and shipments made during specific periods of time. Financial managers are interested in calculated values, such as the total amount of unpaid invoices and the average number of days between the invoice and payment dates. Occasionally, you might want to make calculations on all records of a table, such as finding the historical average value of all invoices issued by a firm. Usually, however, you apply criteria to the query in order to select specific records that you want to total.



Access considers all SQL aggregate functions to be members of the Totals class of functions. You create queries that return any or all SQL aggregate functions by clicking the Totals button (with the Greek sigma, Σ, which represents summation) on the toolbar.

Oracle Ex. 1009, p. 375
Oracle v. DeBellis
IPR2024-00221

The Orders table of Access 97's Northwind.mdb sample database does not include an OrderAmount field that represents the total amount of the order, less freight. (The "Entering a Query Criterion" section of Chapter 9 used a simplified version of this example to demonstrate the use of functions to calculate field values.) To create a sample query that uses the SQL aggregate functions to display the total number of orders, total sales, and the average, minimum, and maximum order values, you need a field that contains the total amount of each order. Follow these steps to create a new table that includes an additional field with a computed Order Amount:

1. Create a new query and add the Orders and Order Details tables to it.



2. Drag the OrderID field of the Orders table to the first column of the Query Design grid, and then drag the OrderDate field to the second column.
3. Type **Order Amount: Sum([Quantity]*[UnitPrice]*(1-[Discount]))** in the Field row of the third (empty) column. This expression sums the net amount of all line items for each order. With the caret in the Order Amount column, click the Properties button of the toolbar to open the Field Properties window. Type **Currency** in the text box for the Format property to format your new column.



4. Click the Totals button on the toolbar. A new row, Total, is added to the Query Design grid. Access adds Group By, the default action, to each cell in the Totals row. The following section discusses the use of Group By.



5. Move to the third column's Total row and press F4 to display the drop-down list of SQL aggregate functions. Select Expression from the list. Your Query Design grid appears as shown in Figure 10.41.

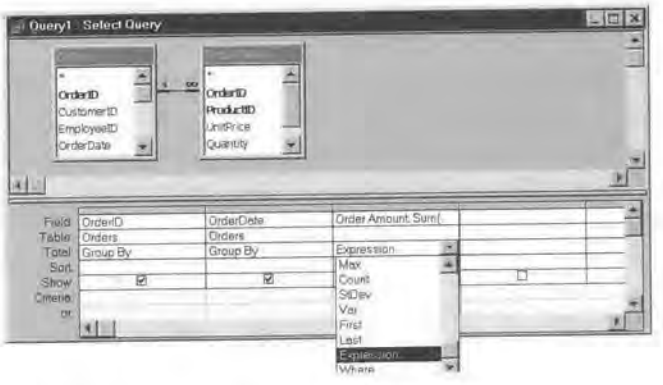


FIG. 10.41 Creating a calculated field with the `Sum()` function.

6. Click the Run button of the toolbar to test your initial entries. Your query in Datasheet view appears as in Figure 10.42.



Oracle Ex. 1009, p. 376
Oracle v. DeBellis
IPR2024-00221

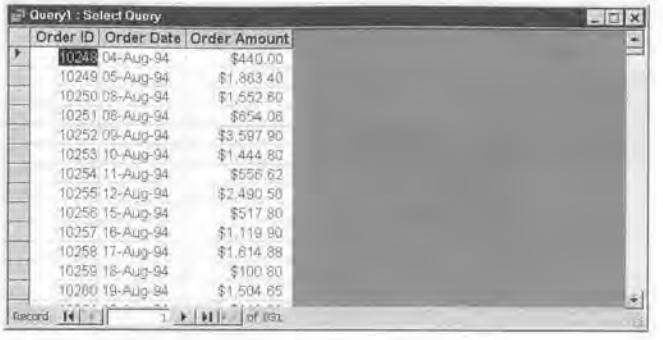


FIG. 10.42 Running the query design shown in Figure 10.41.

7. Close and save your query with the name **Order Totals**.

> **Note**
>
> When you apply the `Format` property to the Order Amount column by selecting or typing **Currency** in the Field Properties window, successive queries that you create do not inherit the value of the `Format` property (instead, the default `Format` value, `Double`, is applied). If you type **Order Amount: CCur(Sum([Quantity]*[Unit Price]*(1-[Discount])))** in the Order Amount column's Field row, however, the `Format` property of successive queries containing the Order Amount field will be set to `Currency`. The **CCur**( ) function coerces the field's data type to `Currency`.

Follow these steps to apply the SQL aggregate functions to the Order Amounts field of the query result set of the Order Totals query:



1. Open a new query and add the Order Totals query. (To base a query on a prior-saved query, click the Queries tab of the Show Table dialog, and add the query as you would a table.)
2. Drag the OrderID field to the first column and then drag the Order Amount column four times to the adjacent column to create four Order Amount columns.)
3. Choose View, Totals to add the Totals row to your Query Design grid. Alternatively, right-click in the grid region and choose Totals from the popup menu.
4. Move to the Total row of the OrderID column and press F4 to display the drop-down list of SQL aggregate functions. Choose Count as the function for the Order ID, as shown in Figure 10.43.
5. Move to the first Order Amount column, open the list, and choose Sum from the Total drop-down list. Repeat the process, choosing Avg for the second Order Amount column, Min for the third, and Max for the fourth.



6. Place the caret in the Count field, and click the Properties button of the toolbar (or right-click in the Count field and then click Properties in the popup menu) to display the Field Properties window. Type **Count** as the value of the Caption property.

Oracle Ex. 1009, p. 377
Oracle v. DeBellis
IPR2024-00221

7. Repeat step 6 for the four Order Amount columns, typing **Currency** for the `Format` property and typing **Sum**, **Average**, **Minimum**, and **Maximum** as the values of the `Caption` property for the four columns, respectively. (You don't need to set the Format property if you used the `CCur()` function in the Order Totals query.)

8. Click the Run button of the toolbar to display the query's result. You haven't specified criteria for the fields, so the result shown in Figure 10.44 is for the whole table. (The values in the result may differ from the values in Figure 10.44 because of records that Microsoft might add to the sample database after this book is published.)

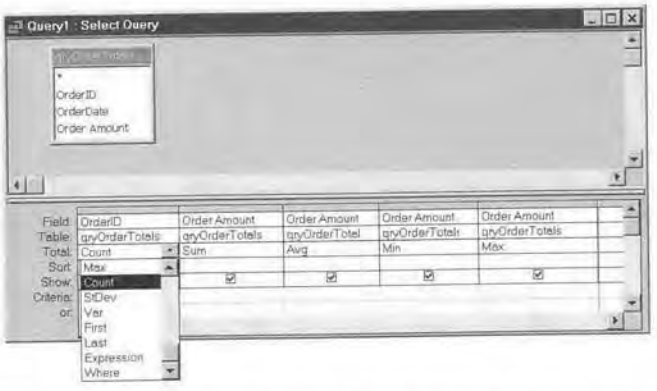


FIG. 10.43 Choosing the SQL aggregate function for calculations based on multiple records in a table.



FIG. 10.44 The result of the all-records query shown in Figure 10.43.

9. Save your query with a descriptive name, such as **qrySQLAggregates**, because you use this query in the two sections that follow.

Oracle Ex. 1009, p. 378
Oracle v. DeBellis
IPR2024-00221

Note

When you run qrySQLAggregates, the Jet database engine determines that Order Totals is a query (`QueryDef` object) rather than a table (`TableDef` object). Thus Jet executes the Order Totals query before executing the qrySQLAggregates query. One of the most important features of Access is that you can execute queries against the query result sets (`Recordset` objects) of other queries—a process called *nesting* queries. In theory, at least, there is no limit to the depth to which you can nest Access queries. As you increase the number of queries in the chain, however, execution slows for the last query in the nested sequence.

Making Calculations Based on Selected Records of a Table

The preceding example query performed calculations on all orders received by Northwind Traders that were entered in the Orders table. Usually, you are interested in a specific set of records—a range of dates, for example—from which to calculate aggregate values. To restrict the calculation to orders that Northwind Traders received in March 1995, follow these steps:



1. Click the Query View button on the toolbar to return to design mode so that you can add criteria to select a specific group of records based on the date of the order.
2. Drag the OrderDate field onto the OrderID column to add OrderDate as the first column of the query. You need the OrderDate field to restrict the data to a range of dates.
3. Open the Total drop-down list in the Order Date column, and choose Where to replace the default Group By. Access deselects the Show box of the OrderDate column. (If you attempt to show a column that provides the SQL `WHERE` restriction, you receive an error message when you run your query.)
4. In the Order Date column's Criteria row, type **Like "3/*/95"** to restrict the totals to orders received in the month of March 1995 (see Figure 10.45). When you use the `Like` criterion, Access adds the quotation marks if you forget to type them.

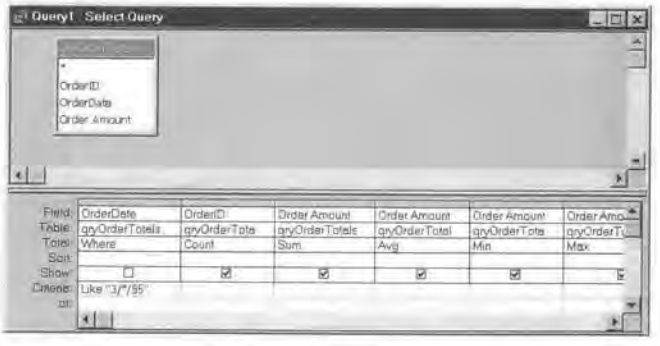


FIG. 10.45 Adding a Where criterion to restrict the totals to a range of records.

Oracle Ex. 1009, p. 379
Oracle v. DeBellis
IPR2024-00221

5. Choose View, SQL to display your query's SQL statement. The Where criterion in the Total row adds a `WHERE` clause to the SQL statement—in this case, `WHERE ((([Order Totals.OrderDate Like "3/*/94")))`—to restrict the totaled records to the records for the specified date range. (The Access query parser tends to add extra sets of parentheses to expressions.) If you don't add the Where instruction to the Total row, the query result consists of rows with the totals of orders for each day of March 1995, not for the entire month.

6. Click the Run button on the toolbar to display the result: the count, total, and average value of orders received during the month of March 1995 (see Figure 10.46).



FIG. 10.46 The result of adding the Where criterion to the query.

You can create a more useful grouping of records by replacing the field name with an expression. For example, you can group aggregates by the year and month (or year and quarter) by grouping on the value of an expression created with the `Format()` function. The following steps produce a sales summary record for each month of 1995, the latest year for which 12 months of data are available in the Orders table:

1. Click the Query View button of the toolbar, and then click the header bar of the query's OrderDate column to select the first column. Press the Insert key to add a new, empty column to the query.

2. Type **Month: Format([OrderDate],"yy-mm")** in the first (empty) column's Field row. (You use the `"yy-mm"` format so that the records sort in date order. For a single year, you also can use `"m"` or `"mm"`, but not `"mmm"` because `"mmm"` sorts in alphabetic sequence starting with Apr.)

3. Change the Where criterion of the Order Date column to **Like "*/*/95"**. Your query design appears as shown in Figure 10.47.

4. Click the toolbar's Run button to display the result of your query (see Figure 10.48). The query creates sales summary data for each month of 1995.

Oracle Ex. 1009, p. 380
Oracle v. DeBellis
IPR2024-00221



FIG. 10.47 Designing a query for a yearly sales summary by month.

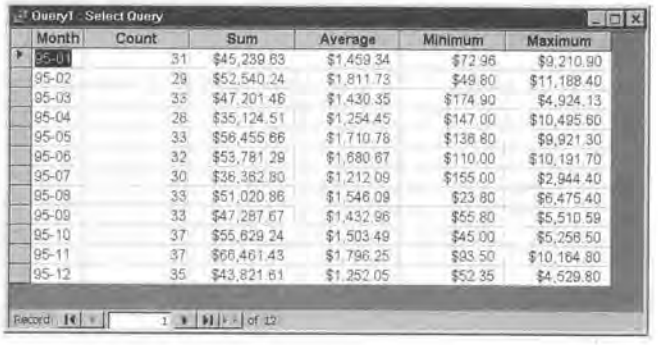


FIG. 10.48 The result set of the query design shown in Figure 10.47.

5. Choose View, SQL to display the SQL statement that created the query result set. The SQL statement in the SQL window of Figure 10.49 has been reformatted for clarity. (Formatting a SQL statement with spaces and newline pairs does not affect the statement's execution.)

6. Choose File, Save As and save the query under a different name, such as **qryMonthlySales**, because you modify the query in the next section.

Designing Parameter Queries

If you expect to run a summary or another type of query repeatedly with changes to the criteria, you can convert the query to a *parameter query*. Parameter queries—which Chapter 8, "Using Query by Example," explained briefly—enable you to enter criteria with the Enter Parameter Value dialog. You are prompted for each parameter required. For the example qryMonthlySales query that you created previously in this chapter, the only parameter likely to change is the range of dates for which you want to generate the product sales data. The two sections that follow show you how to add a parameter to a query and specify the data type of the parameter.

Oracle Ex. 1009, p. 381
Oracle v. DeBellis
IPR2024-00221

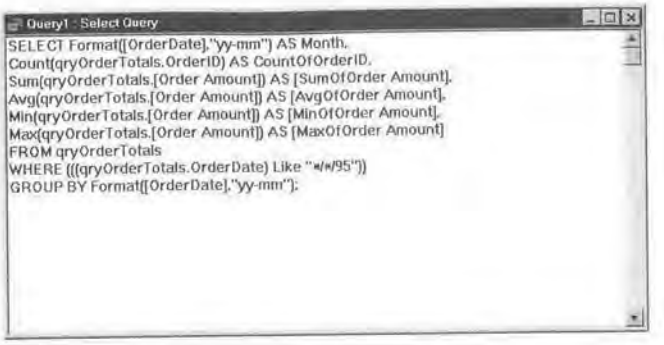


FIG. 10.49 The SQL statement for a yearly sales summary by month.

Adding a Parameter to the Monthly Sales Query

To convert the qryMonthlySales summary query to a parameter query, you first create prompts for the Enter Parameter Value dialog that appears when the query runs. You create parameter queries by substituting the text with which to prompt, enclosed within square brackets, for actual values. Follow these steps:

1. In Design mode, open the qryMonthlySales query that you created in the preceding section.
2. With the caret in the Month column's Field row, press F2 to select the expression in the Field cell. Then press Ctrl+C to copy the expression to the Clipboard.
3. Move the caret to the OrderDate column's Field row, and press F2 to select OrderDate. Then press Ctrl+V to replace OrderDate with the expression used for the first column.
4. Move to the OrderDate column's Criteria cell and replace `Like "*/*/95"` with **[Enter the year and month in YY-MM format:]** (see Figure 10.50).
5. Click the Run button of the toolbar. The Enter Parameter Value dialog appears with the label that you assigned as the value of the criterion in step 4.

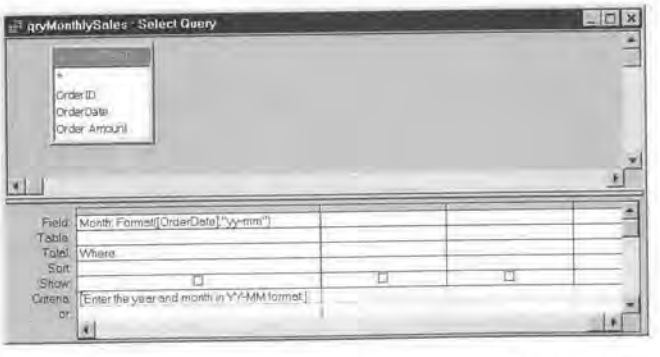


FIG. 10.50 The expression to create the Enter Parameter Value dialog with boxes for the year and month.

Oracle Ex. 1009, p. 382
Oracle v. DeBellis
IPR2024-00221

6. Type **96-04** in the text box to display the data for April 1996, as shown in Figure 10.51.



FIG. 10.51 The Enter Parameter Value dialog for entering the year and month.

7. Click OK to run the query. The result appears as shown in Figure 10.52.



FIG. 10.52 The query result for the 96-04 parameter.

Specifying the Parameter's Data Type

The default field data type for parameters of Access queries is Text. If the parameter creates a criterion for a query column of the Date/Time or Number field data type, you must assign a data type to each entry that is made through an Enter Parameter Value dialog. Data types for values entered as parameters are established in the Query Parameters dialog. Follow these steps to add an optional data type specification to your parameter:

1. Use the mouse to select Enter the Year and Month in YY-MM Format: in the Month column's Criteria cell (omit the square brackets) and copy the text of the prompt to the Clipboard by pressing Ctrl+C.
2. Choose Query, Parameters to display the Query Parameters dialog.
3. To insert the prompt in the Parameter column of the dialog, place the caret in the column and press Ctrl+V. The prompt entry in the Parameter column must match the prompt entry in the Criteria field exactly; copying and pasting the prompt text ensures an exact match. Do not include the square brackets in the Parameter column.
4. Press Tab to move to the Data Type column, press F4 to open the Data Type drop-down list, and choose Date/Time (see Figure 10.53). Click OK to close the dialog.

Oracle Ex. 1009, p. 383
Oracle v. DeBellis
IPR2024-00221

Note

Complete your query design and testing before you convert any type of query to a parameter query. Using fixed criteria with the query maintains consistency during the testing process. Furthermore, you can make repeated changes between Design and Run mode more quickly if you don't have to enter one or more parameters in the process. After you finish testing the query, edit the criteria to add the Enter Parameter Value dialog.



FIG. 10.53 The Query Parameters dialog for assigning data types to user-entered parameters.

The parameter-conversion process described in this section applies to all types of queries that you create if one or more of the query columns includes a criterion expression. The advantage of the parameter query is that you or a user of the database can run a query for any range of values—in this case, dates—such as the current month to date, a particular fiscal quarter, or an entire fiscal year.

Creating Crosstab Queries

Crosstab queries are summary queries that enable you to determine exactly how the summary data appears on-screen. Crosstab queries display summarized data in the traditional row-column form of spreadsheets. Crosstab queries use the Access SQL `TRANSFORM` keyword to indicate that the statements following the keyword are for a crosstab query. (`TRANSFORM` is not an ANSI SQL keyword.) With crosstab queries, you can perform the following processes:

- Specify the field that creates labels (headings) for rows by using the Group By instruction.
- Determine the field(s) that create(s) column headers and the criteria that determine the values appearing under the headers.
- Assign calculated data values to the cells of the resulting row-column grid.

The following list details the advantages of using crosstab queries:

- You can display a substantial amount of summary data in a compact format familiar to anyone who uses a spreadsheet application or columnar accounting form.
- The summary data is presented in a format that is ideally suited for creating graphs and charts automatically with the Access Chart Wizard.

Oracle Ex. 1009, p. 384
Oracle v. DeBellis
IPR2024-00221

- Designing queries to create multiple levels of detail is quick and easy. Queries with identical columns but fewer rows can represent increasingly summarized data. Highly summarized queries are ideal to begin a drill-down procedure by instructing the user, for example, to click the Details button to display sales by product.

Using crosstab queries imposes only one restriction: You cannot sort your result table on calculated values in columns. You cannot, therefore, create a query that ranks products by sales volume. Columns are likely to have values that cause conflicts in the sorting order of the row. You can choose an ascending sort, a descending sort, or no sort on the row label values in the first column.

One of Access 97's Query Wizards is designed to help you create crosstab queries. However, the Crosstab Query Wizard is limited to creating crosstab queries for a single table or query. If your database follows the rules of relational database design, a usable crosstab query is far more likely to be based on at least two tables. Using the Crosstab Query Wizard requires that you create a query including the tables needed for the crosstab query. Thus, you create the examples of crosstab queries in the next two sections with help from this book instead of the Crosstab Query Wizard.

Creating a Monthly Product Sales Crosstab Query

To create a typical crosstab query that displays products in rows and the monthly sales volume for each product in the corresponding columns, follow these steps:

New

1. Open a new query and add the Products, Order Details, and Orders tables to the query.
2. Drag the ProductID and ProductName fields from the Products table to the query's first two columns, and then drag the OrderDate field of the Orders table to the third column.
3. Choose Query, Crosstab Query. The title bar of the query changes from Query1: Select Query to Query1: Crosstab Query. Another row, Crosstab, is added to the Query Design grid.
4. Open the drop-down list of the ProductID column's Crosstab row and select Row Heading. Repeat this process for the ProductName column. These two columns provide the required row headings for your crosstab.
5. Open the Total drop-down list of the OrderDate column and select Where. Type **Like "*/*/95"** in this column's Criteria row to restrict the crosstab to orders received in 1995, the latest year for which 12 months of data are available.
6. Move to the next (empty) column's Field row and type the following:

   Sales: Sum([Order Details].[Quantity]* [Order Details].[UnitPrice])

   Move to the Total row, choose Expression from the drop-down list, and then choose Value from the Crosstab row. The expression calculates the gross amount of the orders received for each product that populates your crosstab query's data cells. (You need to specify the Orders Detail table name; if you don't, you receive an `Ambiguous field reference` error message.)

Oracle Ex. 1009, p. 385
Oracle v. DeBellis
IPR2024-00221

7. In the next (empty) column's Field row, type **Format([OrderDate], "mmm")**. Access adds a default field name, `Expr1:`. Accept the default because the `Format()` function that you added creates the column names, the three-letter abbreviation for the months of the year (`"mmm"` format), when you run the query. The months of the year (Jan through Dec) are your column headings, so move to the Crosstab row and choose Column Heading from the drop-down list. The design of your crosstab query appears as shown in Figure 10.54.

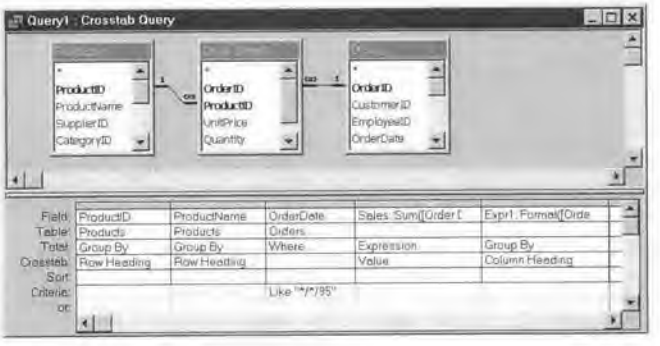


FIG. 10.54 The design of a crosstab query for monthly sales of products.

8. Click the Run button on the toolbar to execute the query. A period of disk activity occurs, followed by a display of the crosstab query's result, shown in Figure 10.55.

Notice that the crosstab query result contains a major defect: The columns are arranged alphabetically by month name rather than in calendar order. You can solve this problem by using fixed column headings, which you learn about in the following section, "Using Fixed Column Headings with Crosstab Queries."

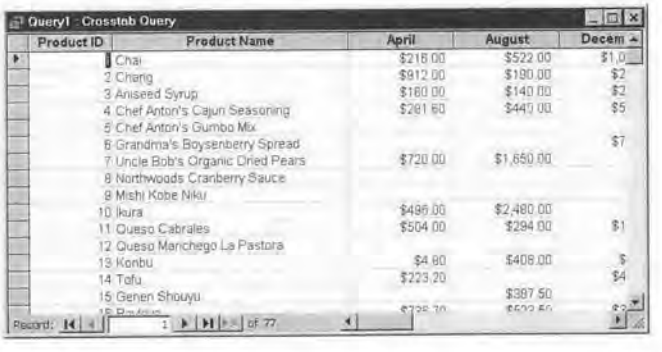


FIG. 10.55 The first result set from the crosstab query design shown in Figure 10.54.

Using Fixed Column Headings with Crosstab Queries

Access uses an alphabetical or numerical sort on row and column headings to establish the sequence of appearance in the crosstab query result table. For this reason, if you use

Oracle Ex. 1009, p. 386
Oracle v. DeBellis
IPR2024-00221

short or full names for months, the sequence is in alphabetic rather than calendar order. You can correct this problem by assigning fixed column headings to the crosstab query. Follow these steps to modify and rerun the query:

◀◀ See "Functions for Date and Time," p. 300

1. Return to query design mode and click the Properties button of the toolbar, or double-click an empty area in the Query Design window's upper pane. The Query Properties window contains an option that appears only for crosstab queries: Column Headings.
2. In the Column Headings text box, type the three-letter abbreviations of all 12 months of the year (see Figure 10.56). You must spell the abbreviations of the months correctly; data for months with spelling mistakes does not appear. You can separate entries with commas or semicolons, and you don't need to type quotation marks because Access adds them. Spaces are unnecessary and undesirable between the Column Headings values. After you complete all 12 entries, close the Query Properties window.
3. Click the Run button of the toolbar. Now the result table, shown in Figure 10.57, includes columns for all 12 months, although you can see only January through April in Figure 10.57. (Scroll to the right to see the remaining months.) If the crosstab appears differently, check whether you properly entered the fixed column headings in the Query Properties window. A misspelled month causes Access to omit the month from the query result set; if you specified `"mmmm"` instead of `"mmm"`, most of your month columns will be blank.
4. Choose File, Save As and save the query with an appropriate name, such as **qry1995MonthlyProductSales**.



FIG. 10.56 Entering fixed column headings in the crosstab Query Properties window.

You can produce a printed report quickly from the query by clicking the Print Preview button on the toolbar and then clicking the Print button.

Oracle Ex. 1009, p. 387
Oracle v. DeBellis
IPR2024-00221

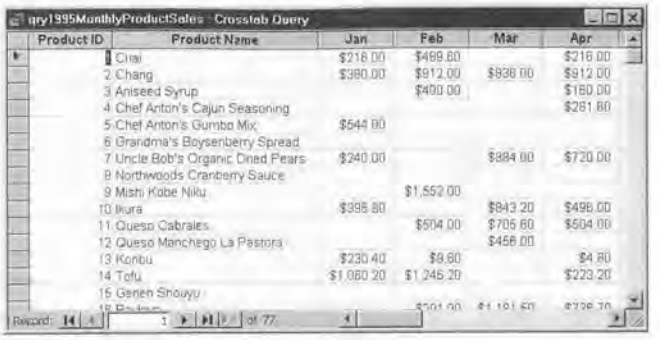


FIG. 10.57 The result table from the crosstab query design with fixed column headings and a date-limiting criterion.

Note

You might want to use fixed column headings if you use the Group By instruction with country names. Users in the United States will probably place USA first, and Canadian firms will undoubtedly choose Canada as the first entry. If you add a record with a new country, you must remember to update the list of fixed column headings with the new country value. Fixed column headings have another hidden benefit: they usually make crosstab queries operate more quickly.

Decreasing the Level of Detail in Crosstab Queries

The result table created by the preceding example query has a row for every product for which Northwind Traders received an order in each month of 1995. Higher-level management usually wants information in the form of a graph or chart to analyze trends in the data. Therefore, you must reduce the number of rows and columns so that you can create a readable graph from the values in the query result table.

To create a summary query that reports quarterly gross sales of products by category (rather than by ProductID), follow these steps:

1. Choose File, Save As and save a copy of the query that you created in the preceding section with a descriptive name, such as **qry1995QuarterlyCategorySales**.
2. In Query Design view, click the Show Table button on the toolbar and add the Categories table to the query.
3. To make the relationships among the tables more clear, click the title bar of each of the field lists in the upper pane and drag the field lists to the positions shown in Figure 10.58.
4. Drag and drop the Categories table's CategoryID and CategoryName fields to the first column's Field cell, which contains ProductID. New CategoryID and CategoryName columns are added to the query. Move down to the Crosstab cell and choose Row Heading from the drop-down list for both new columns.

Oracle Ex. 1009, p. 388
Oracle v. DeBellis
IPR2024-00221

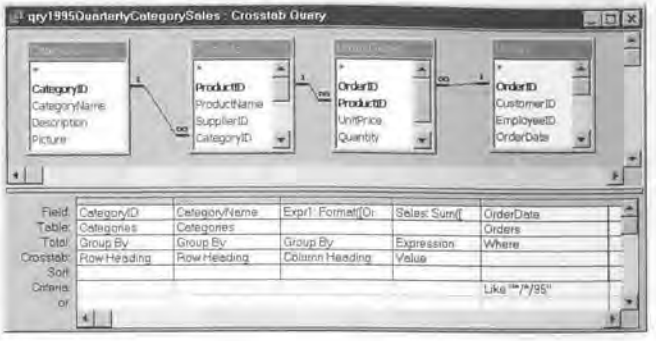


FIG. 10.58 The design of a summary query for quarterly sales by product category.

5. Click the selection bar above the third column's ProductID cell, and then press Delete to delete this column. Repeat this process for the ProductName column. The Crosstab Query Design grid then appears as shown in Figure 10.58.

6. Edit Expr1:, which creates the column headings, so that the expression appears as **Format([OrderDate],""""Quarter ""q"""")**. This results in column headings of Quarter 1 through Quarter 4. (The multiple quotation marks are necessary to specify `Quarter` and a space as literals and `q` as a formatting character.)

7. Double-click an empty area in the Query Design window's upper pane to open the Query Properties window. Delete the Column Headings entries for the month abbreviations, then close the Query Properties window. If you don't delete the fixed column headings, your query can't return any rows. To make your query run more quickly, you can add the four Quarter number headings, separated by commas or semicolons, as the value of the Column Heading property in the Query Properties window.



8. Click the Run button on the toolbar. The query result set appears as shown in Figure 10.59.

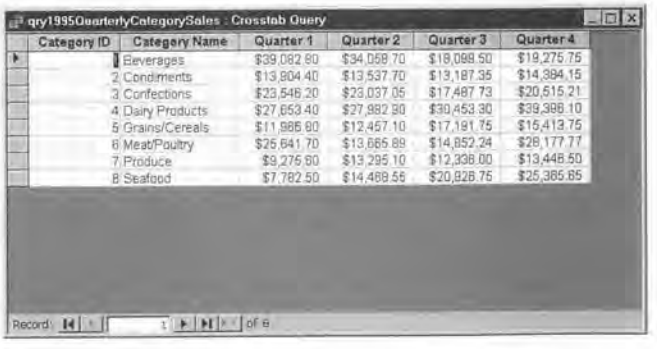


FIG. 10.59 The result set of the query design for the quarterly sales by product category.

Oracle Ex. 1009, p. 389
Oracle v. DeBellis
IPR2024-00221

9. Choose View, SQL to view the SQL statement that creates the crosstab query (see Figure 10.60). The Access SQL TRANSFORM clause, which corresponds to the Values crosstab property, defines the values that appear in the data cells. The PIVOT statement defines the column headings. (The IN predicate following the PIVOT statement specifies the fixed column headings, if any.) TRANSFORM and PIVOT are not reserved words of ANSI SQL. Access crosstab queries and conventional SQL SELECT queries interpret the ANSI SQL IN predicate differently.

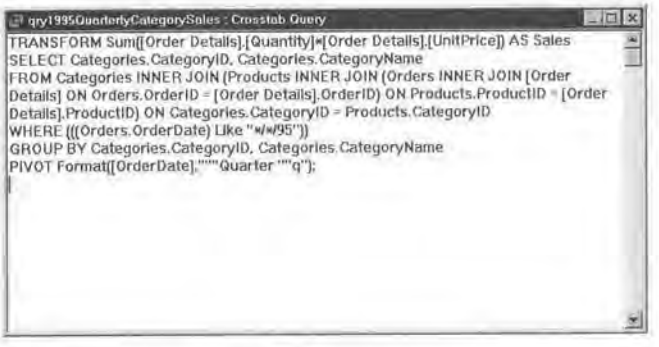


FIG. 10.60 The SQL statement that creates the quarterly sales-by-product query.

10. Choose File, Save to save your query. The qry1995QuarterlyCategorySales query is used for many different purposes in later chapters of this book.

Crosstab queries that display time-series data, such as monthly and quarterly sales for products or categories of products, are often used as the basis for graphs. Chapter 20, "Adding Graphics to Forms and Reports," uses the qry1995QuarterlyCategorySales query to create two types of Access graphs.

Creating Queries from Tables in Other Databases

Access's Query Properties window includes two properties that let you create a query based on tables contained in a database other than the current database. Access calls the database that you open after you launch Access the *current database*. Databases other than the current database commonly are called *external* databases. The use of these two properties is as follows:

- The value of the Source Database property for desktop databases is the path to the external database and, for Access databases, the name of the database file. To run a query against tables contained in the Solutions.mdb sample database, replace (current) in the Source Database text box with the following, as shown in Figure 10.61:

  C:\Program Files\Microsoft Office\Access\Samples\Solutions.mdb

Oracle Ex. 1009, p. 390
Oracle v. DeBellis
IPR2024-00221

To run a query against a set of Paradox tables in the D:\Paradox folder, you type the path only (**d:\paradox**, where *d:* is the logical drive letter). If you're using the ODBC API to connect to a client/server database, you leave the Source Database text box empty.



FIG. 10.61 Setting the Source Database property for a query against an external database.

- The value of the Source Connect Str property also depends on the type of external database being used. If your external Access database is not secure, leave the Source Connect Str text box empty; otherwise, type **UID=UserID;PWD=Password** to specify the user ID and password needed to open the external database. For other desktop databases, type the product name, such as **Paradox 3.5** or **dBASE IV**. ODBC data sources require the complete ODBC connect string. Using ODBC databases is one of the subjects of Chapter 25, "Securing Multiuser Network Applications."

Running a query against an external database is related to running a query against linked tables. When you link tables, the data in the tables is available at any time that your application is running. When you run a query against an external database, the connection to the external database is open only while your query is open in Design or Run mode. A slight performance penalty exists for running queries against an external database—each time that you run the query, Access must make a connection to open the database. The connection is closed when you close the query.

Figure 10.62 shows a query design based on tables contained in the Solutions.mdb sample database that accompanies Access 97. In this case, Access 97's Northwind.mdb is the current database. Figure 10.63 shows the result of executing the query design of Figure 10.62 against the Example Objects, Examples, and Example Topics tables of Solutions.mdb.

Oracle Ex. 1009, p. 391
Oracle v. DeBellis
IPR2024-00221

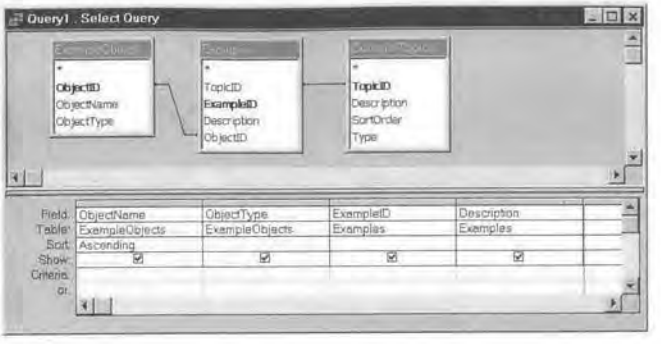


FIG. 10.62 A query design based on tables in an external Access database.

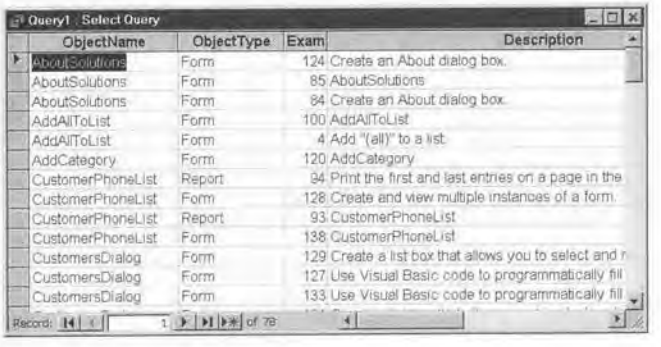


FIG. 10.63 The query result set of the query design shown in Figure 10.62.

Oracle Ex. 1009, p. 392
Oracle v. DeBellis
IPR2024-00221

Chapter 11

Using Action Queries

Action queries create new tables or modify the data in existing tables. Four types of action queries are available in Access:

- *Make-table* creates new tables from the data contained in query result sets. One of the most common applications for make-table queries is to create tables that you can export to other applications, or that summarize data from other tables. A make-table query provides a convenient way to copy a table to another database. In some cases, you can use make-table queries to speed the generation of multiple forms and reports based on a single, complex query.
- *Append* adds new records to tables from the query's `Recordset`.
- *Delete* deletes records from tables that correspond to the rows of the query result set.
- *Update* changes the values of existing fields of table records corresponding to rows of the query result set.

This chapter shows you how to create each of the four types of action queries and how to use Access's cascading deletions and cascading updates of related records. This chapter covers cascading deletions and cascading updates because these features are related to delete and update action queries, respectively.

Creating New Tables with Make-Table Queries

In the following sections, you learn how to use a make-table query to create a new table, Shipping Address, for customers that have different shipping and billing addresses. This process enables the deletion of the shipping address data that, in most of the records in the Orders table, duplicates the address data in the Customers table. Removing duplicated data to new tables is an

Oracle Ex. 1009, p. 393
Oracle v. DeBellis
IPR2024-00221

important step when you are converting data contained in a flat (non-relational) database to a relational database structure. You can use the Table Analyzer Wizard, described in Chapter 23, "Exploring Relational Database Design and Implementation," to perform an operation similar to that described in the following sections. Removing duplicated data manually, however, is one of the best methods of demonstrating how to design make-table queries.

Caution

Always make a backup copy of a table that you are going to modify with an action query. Changes made to table data with action queries are permanent; an error can render a table useless. Invalid changes made to a table with an action query containing a design error often are difficult to detect.



◀◀ See "Creating Not-Equal Theta Joins with Criteria," p. 344
▶▶ See "Using Access 97's Table Analyzer Wizard," p. 813

The example that you create in the following sections extracts data from the Orders table based on data in the Customers table and creates a new table, tblShipAddresses. A modification of the query that you created in the "Creating Not-Equal Theta Joins with Criteria" section of Chapter 10 generates the data for the new table. Make-table queries are useful in converting flat-file tables that contain duplicated data, including tables created by spreadsheet applications, to relational form.

Designing and Testing the Select Query

To create the new shipping addresses table from the data in the Orders table, you first must build a select query. To build a select query, follow these steps:



1. Create a new query and add the Customers and Orders tables to it.
2. Drag the CustomerID field from the Customers table and drop it in the query's first column. The CustomerID field links the Shipping Address table to the Orders table.
3. Drag the ShipName, ShipAddress, ShipCity, ShipRegion, ShipPostalCode, and ShipCountry fields and drop them in columns 2–7, respectively. You use these fields, in addition to CustomerID, to create the new Shipping Address table.

   Next, you want to add criteria to select only those records of the Orders table for which the ShipName doesn't match the CompanyName or the ShipAddress doesn't match the Customers table's address.

4. In the ShipName column's first Criteria row, type the following:

   ```
   <>[Customers].[CompanyName]
   ```

5. In the next row of the ShipAddress column, type the following:

   ```
   <>[Customers].[Address]
   ```


Oracle Ex. 1009, p. 394
Oracle v. DeBellis
IPR2024-00221

This entry must be in a different Criteria row than the CompanyName criterion so that the **Or** operator is applied to the two criteria. The Query Design grid appears as shown in Figure 11.1.

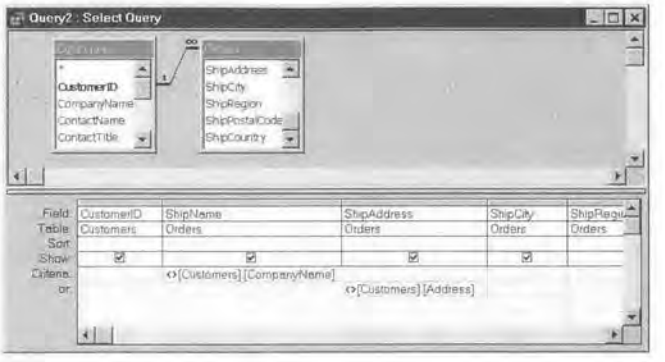


FIG. 11.1 Creating the select query for the new Shipping Address table.

> **Note**
>
> A more precise approach is to add additional **Or** criteria to test for not-equal cities, regions, postal codes, and countries. A customer having exactly the same address in two different cities, however, is highly improbable.

6. Double-click an empty area in the Query Design window's upper pane to open the Query Properties window. Open the Unique Values drop-down list and select Yes.
7. Click the toolbar's Run button to run the select query. Data for customers that placed orders with different billing and shipping addresses appears, as shown in Figure 11.2. (Figure 11.2 contains the CustomerID LETSS, which appears only if you performed the exercises in Chapter 8.)



FIG. 11.2 The data to be added to the new Shipping Address table.

Oracle Ex. 1009, p. 395
Oracle v. DeBellis
IPR2024-00221

Converting the Select Query to a Make-Table Query

Now that you have tested the select query to make sure that it creates the necessary data, you can create the table from the query. To create the table, follow these steps:

1. Choose Query, Make-Table Query. The Make Table dialog appears. Type the name of the table, **tblShipAddresses**, in the Table Name text box (see Figure 11.3). Click OK.



FIG. 11.3 The Make Table dialog for make-table queries.

2. Click the Run button on the toolbar. A message confirms the number of records that you are adding to the new table (see Figure 11.4). Choose Yes to create the new tblShipAddresses table.



FIG. 11.4 The confirmation message box that precedes the creation of the new table.

3. Click the toolbar's Database Window button to activate the Database window, click the Table tab, and open the new tblShipAddresses table. The entries appear as shown in Figure 11.5.



FIG. 11.5 The tblShipAddresses table created by the make-table query.

Oracle Ex. 1009, p. 396
Oracle v. DeBellis
IPR2024-00221

Now complete the design of the new tblShipAddresses table by following these steps:

1. Click the Design View button of the toolbar. The table's basic design is inherited from the properties of the fields of the tables used to create the new table. The tblShipAddresses table does not, however, inherit the primary-key assignment from the Customers table's CustomerID field.



2. Click the toolbar's Properties button to display the Table Properties window. In the Table Properties window's Description text box, type **Shipping addresses different from billing addresses**, as shown in Figure 11.6. Click the Properties button to close the Properties window.





FIG. 11.6 The design of the newly created tblShipAddresses table.

3. The tblShipAddresses table presently has a partial one-to-one relationship with the Customers table because only one shipping address record exists for each customer that has different shipping and billing addresses. A customer might have many different shipping addresses, however, so the relationship of Customers to tblShipAddresses is one-to-many, and you must allow duplicate values in the CustomerID field of tblShipAddresses. You cannot, therefore, create a primary key for the tblShipAddresses table unless you include three fields—CustomerID, ShipName, and ShipAddress—to make multiple entries for one customer unique. This particular example doesn't use a primary key. Choose the CustomerID field, open the Indexed property drop-down list, and choose the Yes (Duplicates OK) value. Indexing improves the performance of queries when you have many different shipping addresses for customers.

4. The CustomerID, ShipName, ShipAddress, ShipCity, and ShipCountry fields are required, so set the value for each of these fields' `Required` property to **Yes**. (Many countries do not have values for the ShipRegion field, and a few countries do not use postal codes.)

Oracle Ex. 1009, p. 397
Oracle v. DeBellis
IPR2024-00221

Establishing Relationships for the New Table

Now you need to complete the process of adding the new table to your database by establishing default relationships and enforcing referential integrity so that all records in the tblShipAddress table have a corresponding record in the Customers table. Access 97's graphical Relationships window makes this process simple and intuitive. To establish the relationship of tblShipAddress and the Customers table, follow these steps:



1. Close the tblShipAddress table. Answer Yes when asked whether you want to save changes to the table's design, and answer Yes again when asked whether you want to apply the new data integrity rules to the table. Click the toolbar's Database Window button to make the Database window active.



2. Click the Relationships button of the toolbar or choose Tools, Relationships to open the Relationships window that establishes the default relationships between tables. Click the toolbar's Show Table button and double-click the tblShipAddresses table to add the table to the Relationships window; then click the Close button. Move the field list to the position shown in Figure 11.7. (Drag up the bottom of the Orders field list to make room for the tblShipAddress field list.)

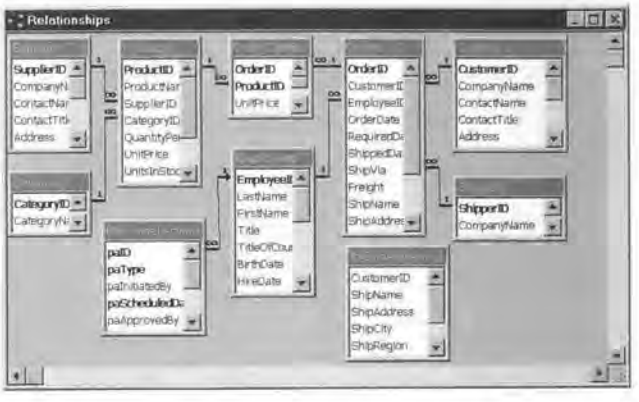


FIG. 11.7 Adding the tblShipAddresses table to the Relationships window.

3. Click the Customers table's CustomerID field, drag the field symbol to the tblShipAddresses table's CustomerID field, and drop the symbol. This example emphasizes the table names, so the direction in which you drag the field is important. The Relationships dialog appears (see Figure 11.8). The field that you select to drag appears in the Table/Query list (the *one* side of the relationship), and the field on which you drop the dragged field appears in the Related Table/Query list (the *many* side of the relationship).

4. Select the Enforce Referential Integrity check box. Access sets the default relation type, One-To-Many, which is the correct choice for this relation. Access also establishes a conventional equi-join as the default join type, so in this case you don't

Oracle Ex. 1009, p. 398
Oracle v. DeBellis
IPR2024-00221

need to click the Join Type button to display the Join Properties window. Select the Cascade Update Related Fields and the Cascade Delete Related Records check boxes to maintain referential integrity automatically.

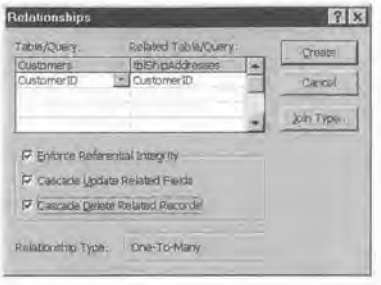


FIG. 11.8 The Relationships dialog for the new shipping addresses table.

5. Choose the Create button of the Relationships dialog to close it. Your Relationships window appears as shown in Figure 11.9. (The tables in Figure 11.9 have been rearranged slightly, and the new join line has been selected to make it more visible.)

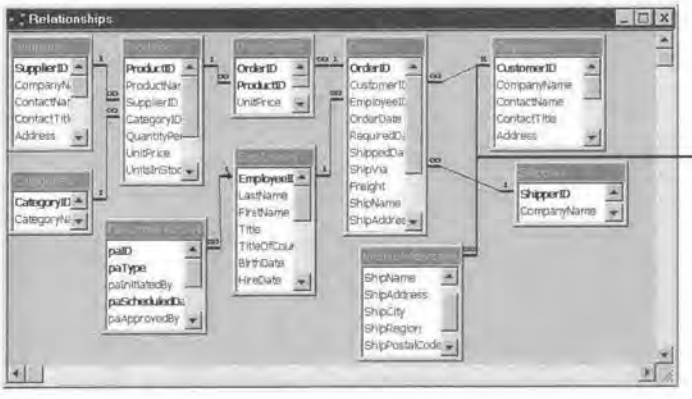


FIG. 11.9 The Relationships window showing a relationship established for the new table.

6. Close the Relationships window and click Yes to save your changes. Save your make-table query with an appropriate name, such as **qryMakeShipAddressesTable**.

Using the New tblShipAddresses Table

After creating a new table from a make-table query, you must take care of several "housekeeping" chores before you can take advantage of your new table. The purpose of creating the new shipping addresses table is to eliminate the data in the Orders table

Oracle Ex. 1009, p. 399
Oracle v. DeBellis
IPR2024-00221

that duplicates information in the Customers table. The additional steps that you must take to use the new table include the following:

- You need a new Number (Long Integer) field, ShipID, for the tblShipAddresses and Orders tables. In the Orders table's ShipID field, you can have a 0 value indicate that the shipping and billing addresses are the same. You then assign a sequential number to each shipping address for each customer. (In the present case, the value of the ShipID field is 1 for all records in tblShipAddresses.) By adding the ShipID field to the tblShipAddresses table, you can create a composite primary key on the CustomerID and ShipID fields. The no-duplicates index on the composite primary key prevents accidental duplication of a ShipID value for a customer.
- Do not delete fields that contain duplicate data extracted to a new table until you confirm that the extracted data is correct and modify all the queries, forms, and reports that use the table. You use the update query described later in this chapter to assign the correct ShipID field value for each record in the Orders table. After you verify that you have assigned the correct value of the ShipID field, you can delete the duplicate fields.
- Add the new table to any queries, forms, reports, macros, and modules that require the extracted information.
- Change references to fields in the original table in all database objects that refer to fields in the new table.

During this process, you have the opportunity to test the modification before deleting the duplicated fields from the original table. Making a backup copy of the table before you delete the fields also is a low-cost insurance policy.

Creating Action Queries to Append Records to a Table

A make-table query creates the new table structure from the structure of the records that underlie the query. Only the fields of the records that appear in the query are added to the new table's structure. If you design and save a Shipping Address table before extracting the duplicated data from the Orders table, you can use an Append query to add the extracted data to the new table.



Another situation in which append queries are useful is when removing duplicate data from a table currently in use. In this case, you use make-table queries to create the related tables and then change them to append queries. You change the type of query by opening the Query menu and choosing Select, Crosstab, Make Table, Append, or Delete, or by clicking the Query Type button of the toolbar while in Design mode and choosing the type of query from the menu.

Oracle Ex. 1009, p. 400
Oracle v. DeBellis
IPR2024-00221

An append query also differs from a make-table query because an append query can have fewer fields than the table to which the query is appending the data. Otherwise, the make-table and append processes are basically identical. To append records to the tblShipAddress table, for example, follow these steps:

1. Open the tblShipAddresses table in Datasheet view, choose Edit, Select All Records, and then press the Delete key to delete all the records from the table. Click Yes when asked to confirm the deletion, and then close the table.



2. Open your make-table query, qryMakeShipAddressTable, from the Database window in Design mode (or choose the query from the Window menu if it is open). If you double-click qryMakeShipAddressTable or open qryMakeShipAddressTable in Datasheet view, you run the make-table query.



3. Choose Query, Append Query. The Append dialog appears with tblShipAddresses as the default value in the Table Name drop-down list, as shown in Figure 11.10. Click the OK button to close the Append dialog.



FIG. 11.10 The Query Properties window for an append query.

> **Note**
>
> To append data to a table, the field names of the query and of the table to which you are appending the records must be identical, or you must specify the field of the table to which the append query column applies. Access does not append data to fields in which the field name differs by even a single space character. The Query Design grid for append queries has an additional row, Append To (shown in Figure 11.11), that Access attempts to match by comparing field names of the query and the table. Default values appear in the Append To row of columns for which a match occurs. If a match does not occur, open the Append To row's drop-down list and select the table's field.

4. Click the toolbar's Run button to execute the append query. A message box displays the number of records that the query will append to the table (see Figure 11.12). Choose Yes to append the records.



5. Open the tblShipAddresses table to verify that you have added the records.

Oracle Ex. 1009, p. 401
Oracle v. DeBellis
IPR2024-00221

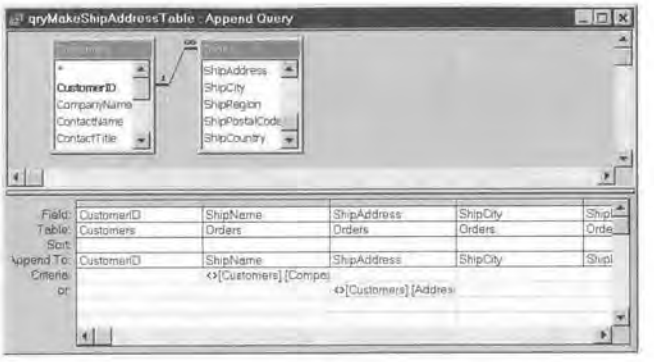


FIG. 11.11 The Append Query Design grid with the added Append row.

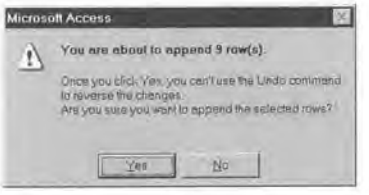


FIG. 11.12 The message box announcing an impending append.

Troubleshooting

After appending records to an existing table, I can't create a primary key on the table.

The Unique Values Only test that you specify in the Query Properties window applies only to the query, not to the table to which you are appending the records. If you want to preclude the possibility of appending duplicate records to the tblShipAddress table, you must first create the composite primary key, discussed in the preceding section, which creates a No Duplicates index on the primary key, and then append the records.

You cannot append records that contain values that duplicate those of the key fields in existing records. If you try to do so, you see a message box that indicates the number of records that cause key-field violations. Unlike the Paste Append operation described in previous chapters, however, Access does not create a Paste Errors table that contains the unappended records.

Deleting Records from a Table with an Action Query

Tip

It's a good practice to run a select query to display the records that you are about to delete and then convert the select query to a deleted query.

Oracle Ex. 1009, p. 402
Oracle v. DeBellis
IPR2024-00221

Often, you might have to delete records from a table. For example, you might want to delete records for canceled orders, or records for customers that have made no purchases for several years. Deleting records from a table with a delete query is the reverse of the append process. You create a select query with all fields (using the * choice from the field list) and then add the individual fields to be used to specify the criteria for deleting specific records. If you don't specify any criteria, Access deletes all the table's records when you convert the select query into a delete query and run it against the table.

To give you some practice at deleting records—you stop short of actual deletion in this case, however—suppose that Northwind Traders' credit manager has advised you that Austrian authorities have declared Ernst Handel (CustomerID ERNSH) insolvent and that you are to cancel and delete any orders from Ernst Handel not yet shipped. To design the query that selects all of Ernst Handel's open orders, follow these steps:

Queries

New

1. Open a new query and add the Orders table to it.
2. Drag the * (all fields) item from the field list to the Field cell of the query's first column.
3. Drag the CustomerID field to the second column's Field cell. You need this field to select a specific customer's record. The fields that comprise the query must be exactly those of the Orders table, so click the Show box to prevent the CustomerID field from appearing in the query's result twice. This field is already included in the first column's * (all fields) indicator.
4. In the CustomerID field's Criteria cell, type **ERNSH** to represent Ernst Handel's ID.
5. A `Null` value in the ShippedDate field indicates orders that have not shipped. Drag the ShippedDate field from the field list to the third column's Field cell. Click the Show box to prevent the ShippedDate field from appearing in the query's result twice, because the first column also includes that field.
6. In the ShippedDate field's Criteria cell, type **Is Null**. To ensure that you delete only records for Ernst Handel *and* those that have not been shipped, you must place this criterion on the same line as that for the CustomerID field. The select query design for the delete query appears as shown in Figure 11.13.

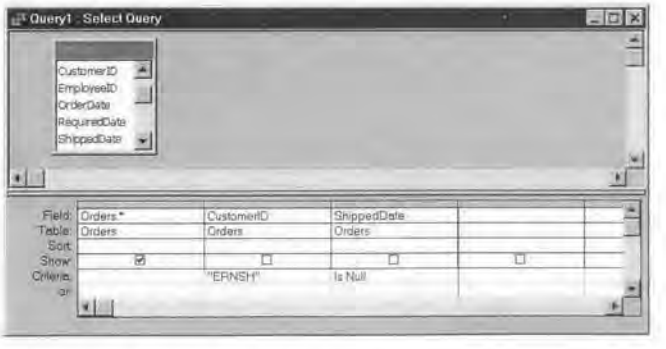


FIG. 11.13 The select query design for a delete query.

Oracle Ex. 1009, p. 403
Oracle v. DeBellis
IPR2024-00221

7. Run the select query to display the records to delete when the delete query runs. Figure 11.14 shows the query result set.

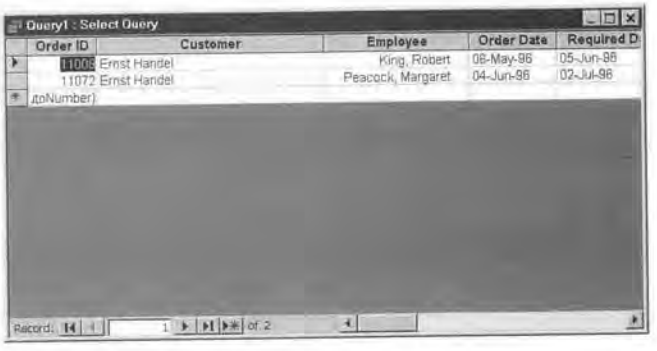


FIG. 11.14 The unshipped orders for Ernst Handel to be deleted from the Orders table.

To proceed with the simulated deletion, follow these steps:

1. Click the toolbar's Database Window button to activate the Database window, then click the Tables button to display the table list. Create a copy of the Orders table by clicking the Orders table entry and pressing Ctrl+C to copy the table to the Clipboard. Press Ctrl+V. The Paste Table As dialog appears. Type **tblOrders** as the name of the new table copy and press Enter. Repeat this process for the Order Details table, naming it **tblOrderDetails**. These two tables are backup tables in case you actually delete the two records for Ernst Handel.

2. Activate your select query and click the toolbar's Design button to return to Design mode. Choose Query, Delete Query. Access then replaces the select query grid's Sort and Show rows with the Delete row, as shown in Figure 11.15. The From value in the Delete row's first column, Orders, indicates that Access will delete records that match the Field specification from the Orders table. The Where values in the remaining two cells indicate fields that specify the deletion criteria.

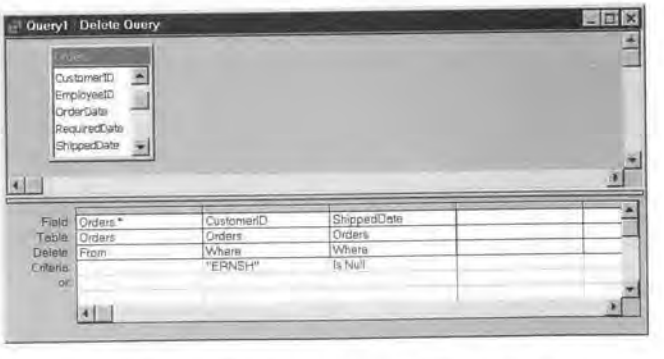


FIG. 11.15 The delete query design created from the select query of Figure 11.13.

Oracle Ex. 1009, p. 404
Oracle v. DeBellis
IPR2024-00221

3. Click the Run button. The message box shown in Figure 11.16 appears, asking you to confirm the deletion of the rows. Click No to prevent the deletion.



FIG. 11.16 The deletion confirmation message box.

Note

Deleting records in a one table when records corresponding to the deleted records exist in a related many table violates the rules of referential integrity; the records in the many table would be made orphans. In this situation, referential integrity is enforced between the Order Details and Orders table, preventing the creation of orphan Order Details records for the two records of the Orders table that you want to delete.

To delete the two records of the Orders table, you must use a process called cascading deletion. First, you delete the Order Details records, then you delete the Orders records. Northwind.mdb's one-to-many relationship between Orders and Order details includes cascading deletions. If you clear the Relationships dialog's Cascade Delete Related Records check box, attempting to delete the two open Orders records for Ernst Handel results in the message shown in Figure 11.17.

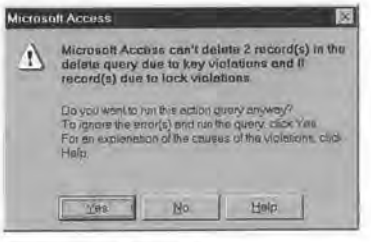


FIG. 11.17 The message box that indicates a violation of the rules of referential integrity.

If you accidentally delete records for Ernst Handel, reverse the process that you used to make the backup tables: Copy the backup tables—tblOrders and tblOrderDetails—to Orders and Order Details, respectively.

Updating Values of Multiple Records in a Table

Update queries change the values of data in a table. Such queries are useful when you must update field values for many records with a common expression. For example, you

Oracle Ex. 1009, p. 405
Oracle v. DeBellis
IPR2024-00221

might need to increase or decrease the unit prices of all products or products within a particular category by a fixed percentage.

To see how an update query works, you perform some of the housekeeping chores discussed earlier in the chapter that are associated with using the tblShipAddress table. To implement this example, you must have created the tblShipAddress table, as described in the "Creating New Tables with Make-Table Queries" section earlier in this chapter. You also must modify the tblOrders and tblShipAddresses tables to include a field for the ShipID code, by following these steps:



1. Click the Tables tab in the Database window and open the tblOrders table in Design mode. If you didn't create the tblOrders table as a backup table for the example of the preceding section, do so now.
2. Select the OrderDate field by clicking the selection button, and press Insert to add a new field between EmployeeID and OrderDate. (Access inserts fields in tables above the selected field.)
3. Type **ShipID** as the field name, select Number as the field data type, and select Long Integer as the field's Field Size. Set the `Required` property's value to **Yes**. The table design pane appears as in Figure 11.18 (which shows the new ShipID field selected).

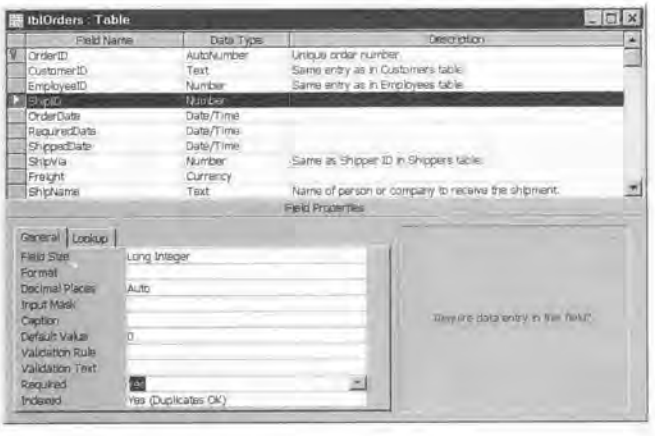


FIG. 11.18 Adding the ShipID field to the tblOrders table.

4. Close the tblOrders table and save the changes to your design. You changed the domain integrity rules when you added the `Required` property, so the message box in Figure 11.19 appears. Choose No to avoid the test, which would fail because no values have been added to the ShipID field.
5. Open the tblShipAddresses table in Datasheet view.

Oracle Ex. 1009, p. 406
Oracle v. DeBellis
IPR2024-00221

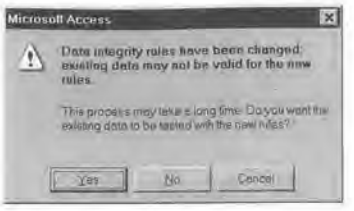


FIG. 11.19 Choosing whether to test changes to domain integrity rules.

6. Click the ShipName field header and choose Insert, Column to add a Field1 field between the CustomerID and the ShipName fields. The capability to add new columns (fields) in Table Datasheet view was a new feature of Access 95.
7. Type **1** in the Field1 cell for each record of the tblShipAddress table.
8. Change to Design mode, and change the name of Field1 to **ShipID**. Access 97 detects from your data entries that the field should be a Number field, and assigns Long Integer as the default Field Size property value. Change the value of the Required property to **Yes**.
9. Select both the CustomerID and the ShipID field by clicking and dragging the mouse.
10. Click the toolbar's Primary Key button to create a composite primary key on the CustomerID and ShipID fields, and then close the tblShipAddress table. This time, you test the changes that you made to the table.

Now, you need to set up a query to select the orders to which you want to add a value of 1 to the ShipID field to indicate the use of data from the tblShipAddresses table. This query is quite similar to that which you used to create the tblShipAddresses table earlier in this chapter. Follow these steps to design your update query:

1. Create a new query and add the Customers and tblOrders tables to it.
2. Drag the Customers table's CompanyName and Address fields to columns 1 and 2, and the tblOrders table's ShipName and ShipAddress fields to columns 3 and 4 of the Query Design grid.
3. Type **<>[Customers].[CompanyName]** in the first Criteria row of the ShipName column and **<>[Customers].[Address]** in the second Criteria row of the Ship Address column. Your query design appears as shown in Figure 11.20.
4. Run the query to verify that you have correctly selected the set of records to be updated. If you changed the name of Let's Stop N Shop to `Let's Stop 'N Shop` in the section "Editing Table Data in Query View" in Chapter 8, "Using Query by Example," the five records for orders placed by this firm appear when you run the query. Using the apostrophe in the entries for Alfred's Futterkiste is inconsistent in

Oracle Ex. 1009, p. 407
Oracle v. DeBellis
IPR2024-00221

the versions of the Customers and Orders tables that were used to write this book (see the highlighted records in Figure 11.21).

Checking for errors of this type is one reason for running the select query before you run the update query. If these records appear, however, you also have a tblShipAddress record for the firm. This is only an example, so you can proceed with the update or you can correct the errors in the query, because the query is updatable.

FIG. 11.20 The select query to test for orders that will require 1 as the value of ShipID.

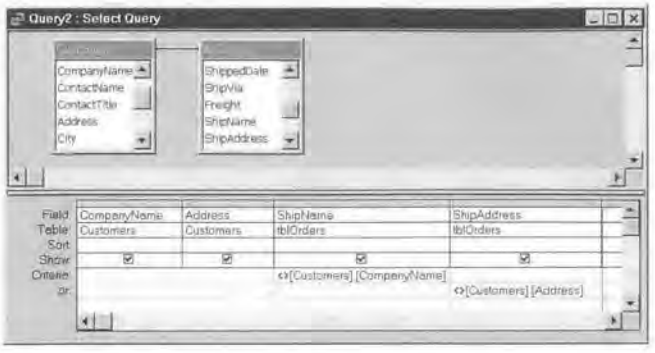


FIG. 11.21 The result of the select query used to test the records to be updated.



After ensuring that you have selected the appropriate records of the tblOrders table for updating, you are ready to convert the select query to an update query by following these steps:



1. Return to Design mode and drag the tblOrders table's ShipID field to the query's first column.
2. Choose Query, Update Query. A new Update To row replaces the Sort and Show rows of the Select Query Design grid.

Oracle Ex. 1009, p. 408
Oracle v. DeBellis
IPR2024-00221

3. In the ShipID column's Update To cell, type **1** to set ShipID's value to 1 for orders that require the use of a record from the tblShipAddresses table. The Update Query Design grid appears as shown in Figure 11.22. The Update To cells of the remaining fields are blank, indicating that Access is not to update values in these fields.

4. Run the update query. A message box such as that shown in Figure 11.23 indicates the number of records that you will update. (The query might return a different number of records because of changes made to the sample database since this book was written.)

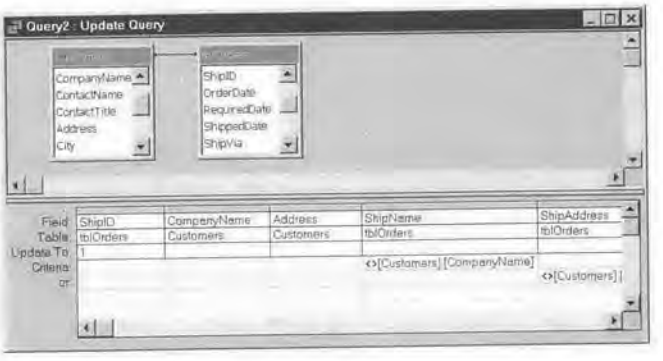


FIG. 11.22 The completed update query's Design grid.

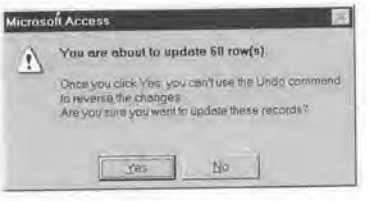
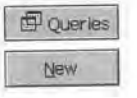


FIG. 11.23 The message box that indicates the number of records to be updated.

5. Click the Database Window button and open the tblOrders table. Check a few records to see that you correctly added the ShipID value of 1.

6. You must add 0 values to the ShipID cells of records that have the same shipping and billing addresses. Close the update query, create a new query, and add only the tblOrders table.



7. Drag the ShipID field to the query's first column and choose Query, Update Query.

8. Type **0** in the Update To row, and **Is Null** in the Criteria row. Then click the Run Query button to replace `Null` values in the ShipID column with 0.

After you check the tblOrders table to verify the result of your second update query, you can change to Table Design mode and safely delete the ShipName, ShipAddress, ShipCity, ShipRegion, ShipPostalCode, and ShipCountry fields.

Oracle Ex. 1009, p. 409
Oracle v. DeBellis
IPR2024-00221

Testing Cascading Deletion and Updating

Access 2.0 introduced two new features that were requested by Access 1.x users: cascading deletion and cascading updating of records having a many-to-one relationship. When you delete a record in a primary or base table on which records in a related table depend, *cascading deletion* automatically deletes the dependent records. Similarly, if you modify the value of a table's primary-key field and a related table has records related by the primary-key field's value, *cascading updating* changes the value of the related foreign-key field for the related records to the new primary-key field value.

Cascading deletions and cascading updates are special types of action queries that the Jet engine executes for you. The following three sections show you how to use Access's cascading deletion and cascading updating features with a set of test tables copied from the Orders and Order Details tables of Northwind.mdb.

Creating the Test Tables and Establishing Relationships

When experimenting with database features, you should work with test tables rather than "live" data. As mentioned in the note at the beginning of this chapter, using copied test tables is particularly advisable when the tables are participants in action queries. The remaining sections of this chapter use the two test tables, tblOrders2 and tblOrderDetails, that you create in the following steps:



1. Click the Database window's Tables tab and then select the Orders table from the list.
2. Press Ctrl+C to copy the table to the Clipboard.
3. Press Ctrl+V to display the Paste Table As dialog.
4. In the Table Name text box, type **tblOrders2**, then click OK or press Enter to create the test tblOrders2 table.
5. Repeat steps 1–4 for the Order Details table, naming the copy **tblOrderDetails**.



6. Open the tblOrders2 table in Table Design view and change the field data type of the OrderID field from AutoNumber to **Number**, and make sure that the Field Size property is set to **Long Integer**. (This change is necessary to test cascading updates in the next section.)

Cascading deletions and updates require that you establish a default relationship between the primary and related tables, and enforce referential integrity. To add both cascading deletions and updates to the tblOrderDetails table, follow these steps:



1. Click the toolbar's Relationships button to display the Relationships window.
2. Click the Clear Layout button to clear the display of the Relationships window. A message box appears warning you that the Relationships window will be cleared. Click Yes to continue.

Oracle Ex. 1009, p. 410
Oracle v. DeBellis
IPR2024-00221

3. Click the toolbar's Show Table button to display the Add Table dialog. Alternatively, right-click the upper pane of the Query window and choose Show Table.



4. Double-click the tblOrders2 and tblOrderDetails items in the list, and then click the Close button to close the Show Table dialog.
5. Click the OrderID field of tblOrders2, then drag the field symbol to the tblOrderDetails table's OrderID field to establish a one-to-many join on the OrderID field. The Relationships window appears.
6. Select the Enforce Referential Integrity check box to enable the two cascade check boxes.
7. Select the Cascade Update Related Fields and Cascade Delete Related Records check boxes, as shown in Figure 11.24.

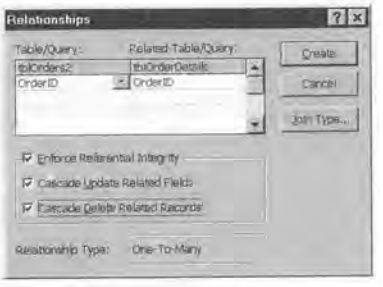


FIG. 11.24 Setting the cascading deletions and updates options.

8. Click the Relationships dialog's Create button to make your changes to the join effective, and then click the Close Window button to close the Relationships window. Click Yes when Access asks if you want to save your changes to the window's layout.

Troubleshooting

When I try to enforce referential integrity, I get a `Can't create relationship to enforce referential integrity` *message.*

You dragged the field symbols in the wrong direction when you created the relationship. The related table is in the Table/Query list and the primary or base table is in the Related Table/Query list. Close the Relationships dialog, click the thin area of the join line to select the join, and then press the Delete key to delete the join. Make sure that you drag the field name that you want from the primary table to the related table.

Oracle Ex. 1009, p. 411
Oracle v. DeBellis
IPR2024-00221

Testing Cascading Deletion

To try cascading deletion with the test tables, follow these steps:

Tables

1. Open the tblOrders2 and tblOrderDetails tables in Datasheet view.
2. Click the surface of the tblOrders2 datasheet to make it the active window, and then click a record-selection button to pick an order in tblOrders2 to delete.
3. Press the Delete key to delete tentatively the selected records and the related order's line-item records in tblOrderDetails.
4. The message shown in Figure 11.25 appears requesting that you confirm the deletion. Choose Yes to delete the records.

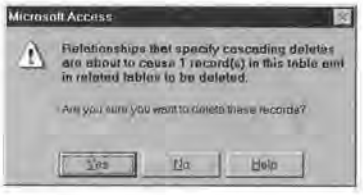


FIG. 11.25 Confirming the cascading deletion.

To verify that you have deleted the related records, you can scroll to the related record or records for the order that you deleted in the tblOrderDetails table. If you opened tblOrderDetails in step 1, the data cell values for the deleted related records are replaced with `#Deleted`.

Testing Cascading Updates

Cascading updates to the foreign-key field of records that depend on a primary-key value that you want to change in a primary table is a valuable feature of Access. Performing updates of primary-key values while enforcing referential integrity is not a simple process; Chapter 4, "Working with Access Databases and Tables," briefly discusses the problems associated with performing such updates manually. To see how Access takes the complexity out of cascading updates, follow these steps:

1. With the tblOrders2 and tblOrderDetails windows open, size and position the two datasheets as shown in Figure 11.26. Then click the surface of the tblOrders2 datasheet to make it the active window. Positioning the two table datasheet windows as shown in Figure 11.26 enables you to see the cascading updates in the tblOrderDetails window as they occur.
2. Change the value of the OrderID cell of the first record to the order number that you deleted in the preceding section. Alternatively, change the value of the OrderID cell to a value, such as **20000**, that is outside the range of the values of the test table.
3. Move the caret to another record to cause the cascading update to occur. You see the changes in the OrderID foreign-key field of the related dependent records immediately (see Figure 11.26).

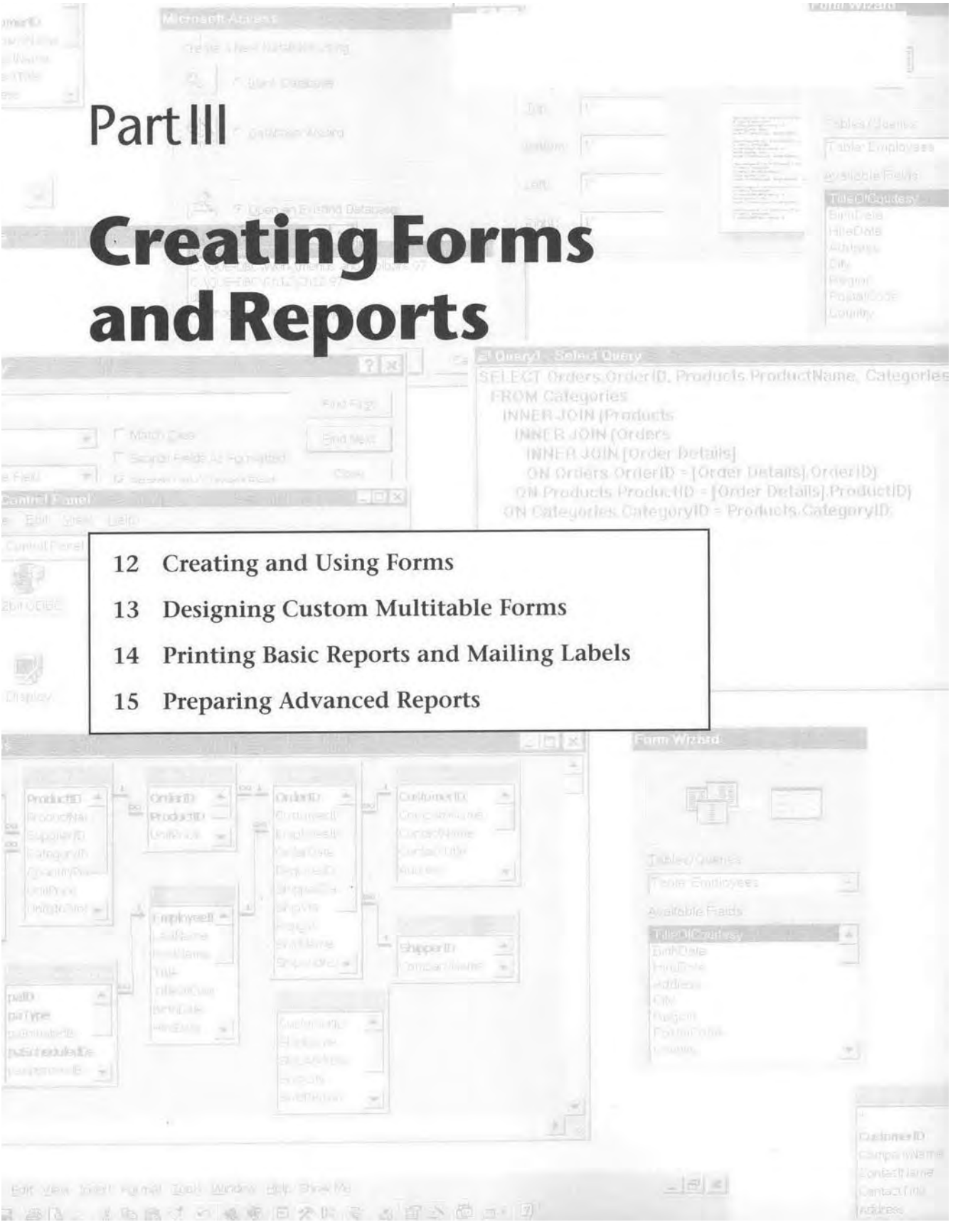
Oracle Ex. 1009, p. 412
Oracle v. DeBellis
IPR2024-00221



FIG. 11.26 An example of a cascading update.

No confirmation message appears when you execute a cascading update, because the effect of a cascading update is reversible. If you make an erroneous entry that causes an undesired cascading update, you can simply change the entry to its original value by choosing Edit, Undo Saved Record. Alternatively, you can simply reenter the original or the correct value manually.

Oracle Ex. 1009, p. 413
Oracle v. DeBellis
IPR2024-00221



Part III

Creating Forms and Reports

12 Creating and Using Forms

13 Designing Custom Multitable Forms

14 Printing Basic Reports and Mailing Labels

15 Preparing Advanced Reports

Oracle Ex. 1009, p. 414
Oracle v. DeBellis
IPR2024-00221

Chapter 12

Creating and Using Forms

Access *forms* create the user interface to your tables. Although you can use Table view and Query view to perform many of the same functions as forms, forms offer the advantage of presenting data in an organized and attractive manner. You can arrange the location of fields on a form so that data entry or editing operations for a single record follow a left-to-right, top-to-bottom sequence. Forms let you create multiple-choice selections for fields that use shorthand codes to represent a set of allowable values. A properly designed form speeds data entry and minimizes operator keying errors.

Forms are constructed from a collection of individual design elements called *controls* or *control objects*. Controls are the components you see in the windows and dialogs of Access and other Windows applications. You use *text boxes* to enter and edit data, *labels* to hold field names, and *object frames* to display graphics. A form consists of a window in which you place two types of controls: some that display the data in your tables, and others that display static data such as labels or logos.

This chapter concentrates on creating forms that consist only of text-based controls. Part V, "Integrating Access with Other Office 97 Applications," provides explanations of object linking and embedding (OLE), the method Access uses to incorporate graphs and other graphical elements in forms and reports.

Access forms are versatile; they let you complete tasks that you cannot complete in Table view or Query view. You can validate entries based on information contained in a table other than the one you are editing. You can create forms that incorporate other forms (a form within a form is called a *subform*). Forms can calculate values and display totals. This chapter shows you how to create a form using the Access Form Wizard and how to modify the form to speed up data entry. Chapter 13, "Designing Custom Multitable Forms," explains how to use the Form Toolbox to add controls to forms, as well as how to establish default values and validation rules with forms.

Oracle Ex. 1009, p. 415
Oracle v. DeBellis
IPR2024-00221

Identifying Types of Forms

The content and appearance of your form depend on its use in your database application. Database applications fall into three basic categories:

- *Transaction processing* applications add new records to tables or edit existing records. Transaction-processing applications require write access to (permissions for) the tables that are linked to the form.
- *Decision support* applications supply information as graphs, tables, or individual data elements but don't allow the user to add or edit data. Decision-support applications require only read access to the tables that are linked to the form.
- *Database maintenance* applications perform administrative functions such as creating databases or database tables, controlling database access by users, providing security assurance via encryption, performing periodic database compaction, and providing backup operations. Database-maintenance applications require full permissions for all of the objects in the database.

Forms are key elements in transaction-processing and decision-support applications, which are described in the following sections. Common database maintenance operations don't require forms, but forms can be useful for maintaining records of maintenance activities.

Forms for Transaction Processing

Forms for transaction processing usually operate directly on tables when only one table is involved. If a single form is used for adding or editing information in more than one table, you can create a query that includes all of the fields you need and then base the form on the query. Your primary form also can use a single table as its data source but use a subform that has a related table as its data source. An example of a transaction-processing form that uses the subform approach is the Orders form of the Northwind Traders sample database (see Figure 12.1). The datasheet subform that appears below the Order ID, Order Date, Required Date, and Shipped Date text boxes is used to display and add line items on an invoice.

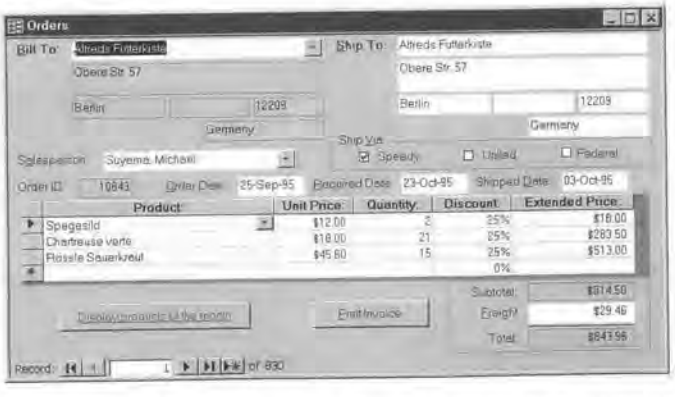


FIG. 12.1 The Orders form of the Northwind Traders database is a typical form for transaction processing.

Oracle Ex. 1009, p. 416
Oracle v. DeBellis
IPR2024-00221

This chapter concentrates on forms used for transaction processing, but the techniques you learn are applicable to forms used for any other purpose.

Forms for Decision Support

Forms designed only to present information fall into the category of decision support; these forms provide historical data that managers and supervisors use to determine a course of action. You can design decision-support forms for long-range planning or short-term decisions. Short-term decisions relate to a single action, such as granting a larger credit line to a customer or sending a sales representative to determine why the customer's purchases have declined. An example of a form to support a short-term decision is Northwind's Quarterly Orders form (see Figure 12.2).

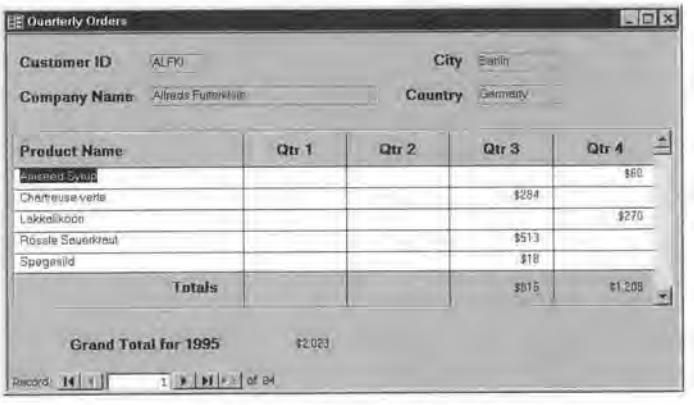


FIG. 12.2 Quarterly Orders is an example of a form used for decision support.

The Quarterly Orders form consists of a main form that displays the Customer ID, Company Name, City, and Country, and a subform that displays quarterly sales of products to the customer. The main form is based on the Customers table. The subform consists of a separate form based on the Quarterly Orders by Product crosstab query. Access enables you to include subforms within forms and even subforms within subforms. (Including a subform within a subform is called *nesting*). You can have up to three levels: main form, subform, and sub-subform.

Forms that support short-term decisions often are based on crosstab queries that summarize data in a time series, such as sales to a customer totaled by month, quarter, or year. A table used to support the decision to grant additional credit to a customer might list by quarters the number of invoices issued to the customer, total purchases, and average payment times in days.

Creating a Transaction-Processing Form with the Form Wizard

The form that you create in this example is typical of the transaction-processing forms used to add new records to the *many* side of a one-to-many relationship. Adding line items to an invoice is an example of when a form of this kind—called a *one-to-many*

Oracle Ex. 1009, p. 417
Oracle v. DeBellis
IPR2024-00221

form—is necessary. The object of the Personnel Actions form is to add new records to the Personnel Actions table or allow you to edit the existing records. If you didn't add the Personnel Actions table shown in Figure 12.3 to the Northwind Traders database in Chapter 4, "Working with Access Databases and Tables," or Chapter 5, "Entering, Editing, and Validating Data in Tables," do so before proceeding with this example. The structure of the Personnel Actions table is provided in Appendix C, "Data Dictionary for the Personnel Actions Table."

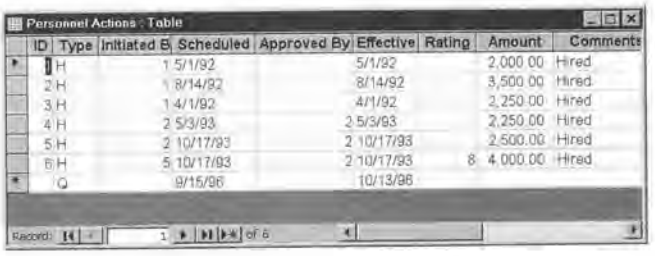


FIG. 12.3 The Personnel Actions table.

If you didn't add records to the Personnel Actions table when you created it in Chapter 4, you can add them with the Personnel Actions form you're going to create now with the assistance of Access's Form Wizard.

Choosing the Form Type

The Personnel Actions form that you create in this exercise allows you to add new entries to the Personnel Actions table. The form also has a subform that displays all of the previous personnel actions for a given employee. The majority of forms found in common database applications are one-to-many forms, and most one-to-many forms require a subform to display data from the many side of the relationship.

The Personnel Actions form is intended as both a transaction-processing form and a decision-support form. You can take two approaches to designing a form that accomplishes the objectives of the Personnel Actions form:

- Use the Employees table as the source of the data for the main form, and use the subform to display, add, and edit records to the Personnel Actions table. This method allows you to add a new employee to the Employees table as well as to add a new Personnel Actions record.
- Use the Personnel Actions table as the source of data for both the main form and the subform. You cannot construct forms of this type with the Form Wizard. In this case, you could not add a new employee using this form because the Employees table has a one-to-many relationship with the Personnel Actions table being edited. This approach is demonstrated in Chapter 13, "Designing Custom Multitable Forms."

The first approach is by far the easiest and is the approach you'll use to create this version of the Personnel Actions form.

Oracle Ex. 1009, p. 418
Oracle v. DeBellis
IPR2024-00221

Creating the Basic Form with the Form Wizard

The easiest way to create a form and subform is with the Access Form Wizard. The Form Wizard allows you to create forms (with or without subforms) that contain fields from one or more tables or queries. The Form Wizard creates the basic design of the form and adds text box controls to display and edit the values of data items.

To create the Personnel Actions form with the Form Wizard, follow these steps:

1. Click the Forms tab of the Database window, then click the New button. The New Form dialog appears, as shown in Figure 12.4.



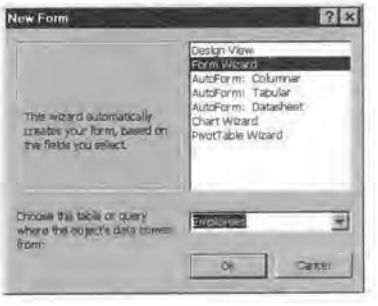


FIG. 12.4 The New Form dialog.

2. Select `Form Wizard` from the list in the New Form dialog. Access 97's Form Wizard allows you to create either a simple form without a subform, or a data form that does contain a subform. The Design View choice opens a blank form in Design mode. The various AutoForm choices automatically create forms with the specified layouts: Columnar, Tabular, and Datasheet. The Chart Wizard choice invokes the ChartWizard to add a graph or chart to your form, while the PivotTable Wizard choice helps you create a form based on Excel pivot tables.

3. The drop-down list at the bottom of the New Form dialog lists all of the existing tables and queries that can serve as a source of data for a form. Click the arrow to open the drop-down list, then click to select the Employees table. Click OK, and Access displays the first dialog of the Form Wizard (see Figure 12.5).

4. Click to select the EmployeeID field in the Available Fields list, then click the > button to move the EmployeeID field from the Available Fields list to the Selected Fields list. Alternatively, you can double-click the field name to move it.

   Repeat this step for the LastName, FirstName, and Title fields of the Employees table. (Placing these fields on the form allows you to edit data from these fields in the Employees table.)

5. Open the Tables/Queries drop-down list and select the Personnel Actions table. The Available Fields list changes to show the available fields in the Personnel Actions table.

Oracle Ex. 1009, p. 419
Oracle v. DeBellis
IPR2024-00221

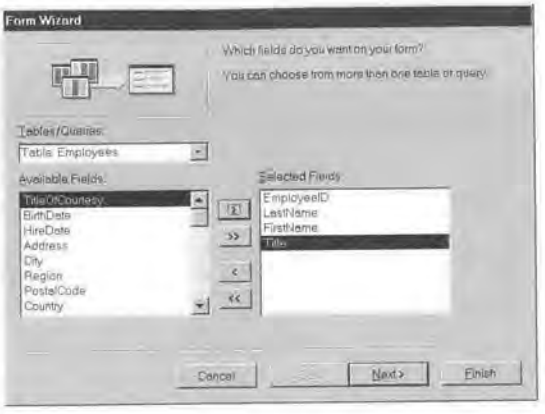


FIG. 12.5 Selecting the fields that you want on your form.

6. Click the >> button to copy all of the fields from the Available Fields list to the Selected Fields list. (You copy all fields from the table onto this form because you need to be able to edit every field in the Personnel Actions table; copying all fields from the table also allows you to edit the FirstName and LastName fields from the Employees table.)

7. Because the EmployeeID field from the Employees table is included in the Selected Fields list, you don't need to include the paID field from the Personnel Actions table on the form. Select the paID field in the list of Selected Fields, then click the < button to move this field out of the Selected Fields list and back to the Available Fields list. Finally, click the Next button to display the second step of the Form Wizard, shown in Figure 12.6.

> **Note**
>
> If you realize that you made an error—or if you change your mind about something—and you're on a later step of the Form Wizard, you can click the Back button to return to and modify your previous choices. You can also click Cancel at any time to abort the form creation and get back to the Database window.

8. Because the fields you have selected to appear on the form come from two different tables, the Form Wizard asks how you want to view the data. Because you want to view the data by employee, with the employee's personnel action data in a subform, click By Employees, and make sure that the Form With Subform(s) option is selected (see Figure 12.6). The picture in the upper-right area of the Form Wizard dialog changes to show the fields of the master form (from the Employees table), with a sunken area containing the fields of the subform (from the Personnel Actions table). Click Next to reach the third step of the Form Wizard.

Oracle Ex. 1009, p. 420
Oracle v. DeBellis
IPR2024-00221

Note

In one-to-many forms, the subform needs to be linked to the main form so that all records displayed in the subform are related to the current record displayed in the main form. The Access 97 Form Wizard obtains the information it needs to link the main form and subform from a join in the Relationships window (in this case, between the Employees table and the Personnel Actions table).

If you haven't established a relationship between the two tables in the Relationships window, the Form Wizard skips the step that asks you to choose how to view the data and lay out the subform (steps 8 and 9). The Form Wizard then goes directly to the dialog that asks which style you want for the form (step 10). For this procedure to work correctly, you must establish a relationship between the two tables, as described in Chapter 4, "Working with Access Databases and Tables."

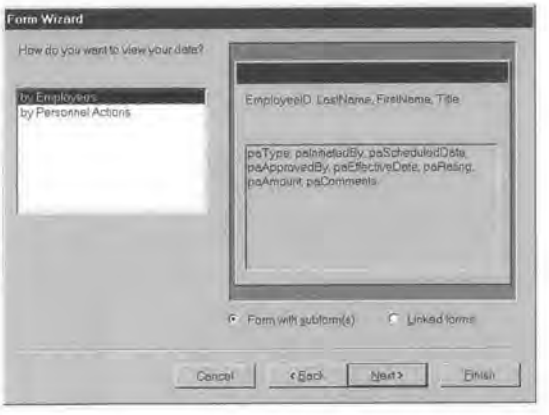


FIG. 12.6 Selecting the master table in the form-subform relationship.

9. The Form Wizard now displays the dialog shown in Figure 12.7, which asks you to select the layout style for the subform. Select the Tabular option. This option creates a subform that displays the data from the Personnel Actions table in a tabular format that is similar to Datasheet view but one in which you can change the formatting (colors, column headings, and so on). Click Next to reach the fourth step in the Form Wizard.

10. The Form Wizard displays the dialog shown in Figure 12.8, which asks you to select a style for the new form. The Access Form Wizard has several predefined styles. Because the sample form we're creating is for use by a data-entry operator—and doesn't require special effects to highlight or decorate any fields—you can click `Standard`. Click Next to reach the Form Wizard's final dialog, shown in Figure 12.9.

11. The Form Wizard asks you to type a name for the master form and any subforms. The Form Wizard also asks you to select what should happen with the form after the Form Wizard has finished creating it. Type **frmPersonnelActions**

Oracle Ex. 1009, p. 421
Oracle v. DeBellis
IPR2024-00221

in the Form text box and **sbfPersonnelActions** in the Subform text box. Select the Open the Form to View or Enter Information option, then click Finish to complete your form. (If you want Access to display help for working with your completed form, select the Display Help on Working with the Form check box before you click Finish.)

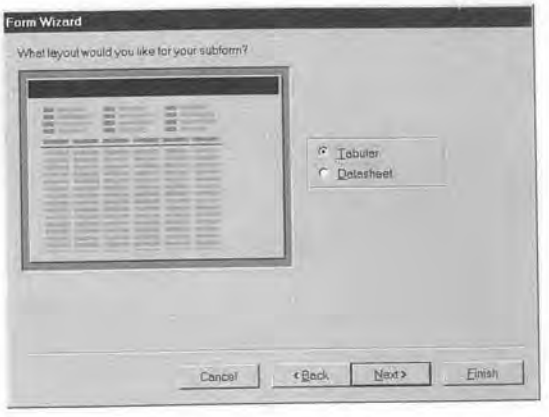


FIG. 12.7 Selecting a tabular layout for the Personnel Actions subform.

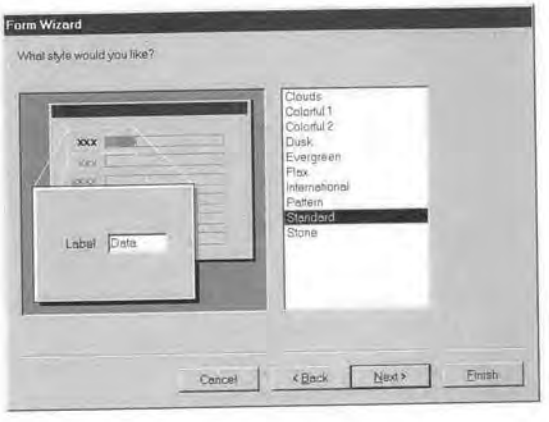


FIG. 12.8 Selecting a predefined form style in the Form Wizard.

Tip

Access suggests default names for the form and any subforms; make sure that you type in names that describe what the form really does. Also, make sure that you include the name of the main form (or an abbreviation) in the name of your subform so that the relationship between the form and subform is evident.

Oracle Ex. 1009, p. 422
Oracle v. DeBellis
IPR2024-00221

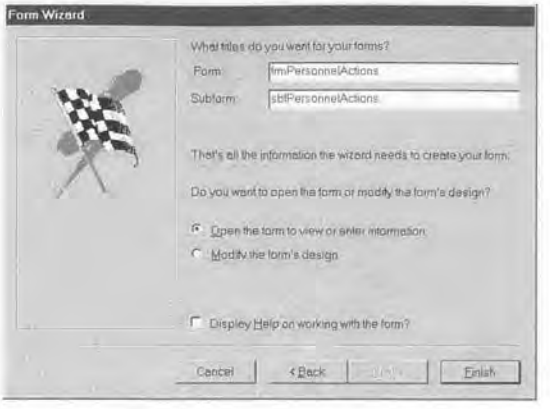


FIG. 12.9 Entering a name for the main form and its subform.

> **Note**
>
> In previous versions of Access, the title you entered for a form was inserted as a heading in the actual form. Instead of doing this, Access 95 and Access 97 use the form title you enter in the final step of the Form Wizard as the name of the form. The title is displayed in the Form window's title bar but not as part of the actual form.

The Form Wizard creates the form and any subforms, then automatically saves them. When the Form Wizard has finished creating the forms, it displays the main form (see Figure 12.10).

> **Caution**
>
> If you didn't select the Open the Form to View or Enter Information option in the final step, you'll be placed automatically into Design view to work with your new form.

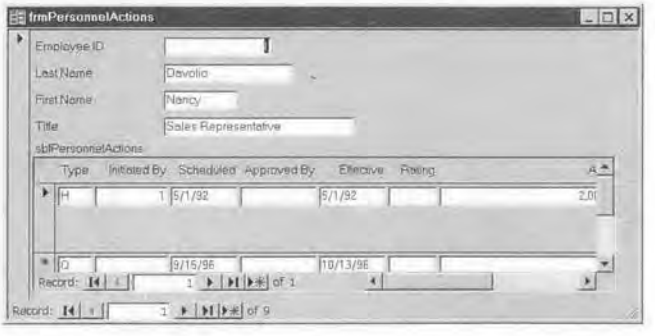


FIG. 12.10 The basic Personnel Action form created by the Form Wizard.

Oracle Ex. 1009, p. 423
Oracle v. DeBellis
IPR2024-00221

On the main form, the Form Wizard creates a single column of text boxes—each with an associated label—for entering or editing data values in each of the fields from the Employees table that you placed on this form. The subform contains all of the fields from the Personnel Actions table (except the paID field) arranged in a tabular layout. Access uses the field names as default text box labels and also as column headings for the tabular subform. Access uses the name that you entered for the subform as the label for the subform area.

In Figure 12.10, notice that the paAmount and paComments fields are partially or completely obscured, and scrollbars appear in the subform area. The subform is larger than the area created for it in the main form, so Access automatically adds scrollbars to let you access all data displayed in the subform.

The basic form as created by the Form Wizard is immediately usable, but could benefit from cosmetic adjustments to the layout of both the main form and subform. The remaining discussions and exercises in this chapter show you how to modify forms created with the Form Wizard; you can apply these form-editing skills when you create your own forms from scratch, as described in the next chapter.

No matter how expert you become at designing Access forms, using the Form Wizard to create the basic form design saves you time.

Using the Form Design Window



To modify the design of your new form, click the Design View button on the toolbar. The Form Design window appears (see Figure 12.11, where the Design window has been maximized). The floating window that appears in Form Design mode contains an undocked toolbar, called the *Toolbox*, that allows you to place new control elements on a form. Using the Toolbox to add new control elements to the form is covered in the next chapter. For this exercise, hide the Toolbox by clicking the Toolbox button on the Forms toolbar or by clicking the Close Window button in the upper-right corner of the Toolbox.



Note

Access usually shows the Toolbox automatically whenever you enter Form Design mode. If you've manually closed the Toolbox, Access does not automatically display it the next time you open the Form Design window. To display the toolbox, click the Toolbox button on the Forms toolbar, or choose View, Toolbox.

The Personnel Action Entry (frmPersonnelActions) form lets you experiment with methods of modifying forms and their contents, which are described in the following sections.

Oracle Ex. 1009, p. 424
Oracle v. DeBellis
IPR2024-00221

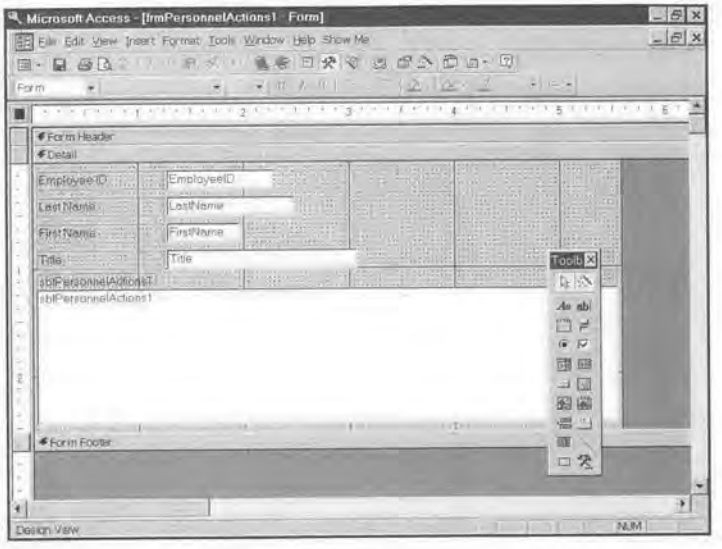


FIG. 12.11 The basic frmPersonnelActions form in Design view.

Caution

Do not save the form with the changes you make when following the instructions in this section. These changes are for demonstration purposes only. Saving these changes would permanently modify the form you created in the preceding section. If you really want to experiment, you can make a copy of the frmPersonnelActions main form and sbfPersonnelActions subform, and then work with the form copies. (Make a backup copy of a form the same way you make a backup copy of a table: choose Edit, Copy to copy the selected form, and then choose Edit, Paste to paste a copy of the form.)

Elements of the Form Design Window

Forms can be divided into three sections: Form Header, Detail, and Form Footer. Headers and footers are optional. The Form Design window includes the following basic elements:

- The Form Design toolbar contains buttons that are shortcuts for menu selections in Form Design mode. The functions of the buttons, and their equivalent menu choices, are listed in tables in the next section, "Form Design Toolbar Buttons and Menu Choices."
- The Formatting toolbar contains buttons that are shortcuts for color, text, border, and various other formatting options. The functions of the formatting buttons and their equivalent menu choices are listed in tables in the next section.
- Vertical and horizontal rulers help you determine the size and placement of objects on the form. The rulers are calibrated in inches for the United States version of Access and in centimeters for versions of Access that are supplied to countries where the metric system is used.

Oracle Ex. 1009, p. 425
Oracle v. DeBellis
IPR2024-00221

- A bold vertical line (shown to the right of the toolbox in Figure 12.11) indicates the position of the right margin of the form. You can move this margin indicator line by clicking and dragging it to the desired location.
- A bold horizontal line indicates the bottom margin of the form. You can click and drag this line to a new location. Margins are important when you are designing a subform to fit within a rectangle of a predetermined size on the main form.
- Vertical and horizontal scroll bars let you view portions of the form outside the boundaries of the form window.
- A Form Header bar defines the height of the form's header section. The form header bar applies only if you choose to add a header and footer to your form. The Form Header section contains static text, graphic images, and other controls that appear at the top of form. The header only appears on the first page of a multipage form.
- A Form Detail bar divides the Form Header from the rest of the form. Form controls that display data from your tables and queries, plus static data elements such as labels and logos, are on the Form Detail bar.
- A Form Footer bar defines the height of the form's footer section. The Form Footer section is similar in function to the Form Header section. If you print a multipage form, the Form Footer appears only at the bottom of the last page.

Note

Although the form shown in Figure 12.11 has both Form Header and Form Footer sections, neither section takes up any space on the form—that's why the Form Header bar touches the Detail bar, and the Form Footer bar touches the bottom margin of the form. Even though no text or other information is in the header and footer areas, the Form Wizard adds these elements to the form automatically. When you create a new, blank form without using the Form Wizard, header and footer sections are not added automatically.

You can add Form Header and Form Footer sections to a form—or delete these sections—by choosing View, Form Header/Footer. (If the form currently contains these sections, a check mark appears to the left of the Form Header/Footer menu choice.) Clear the check marks to delete the Header and Footer sections of the form.

Note

If a header or footer section contains any text or other form controls when you try to delete it, Access displays a dialog warning that you will lose the contents of the header and footer.

Form Design Toolbar Buttons and Menu Choices

The Form Design toolbar of Access 97 contains several buttons that apply only to the design of forms. You select color and font options from the Format toolbar, which was a new feature in Access 95. Table 12.1 lists the function and equivalent menu choice for

Oracle Ex. 1009, p. 426
Oracle v. DeBellis
IPR2024-00221

each of the Form Design toolbar buttons. The buttons that relate to text and color formatting are described in the following section, "The Formatting Toolbar."

Table 12.1 Standard Toolbar Buttons in Form Design Mode

Button	Function	Menu Choice
	Displays the form in Run mode (clicking the arrow at the right of this button displays a drop-down list that allows you to select Datasheet view).	View, Form View
	Saves the current form.	File, Save
	Prints all records in the table using the on-screen form to format the printed data and using the current printer settings.	n/a
	Selects Print Preview to display how your form appears if printed. You can print the form from the Print Preview window.	File, Print Preview
	Starts the spelling checker to check the spelling of the selected label control.	Tools, Spelling
	Cuts selected objects from the form and puts them on the Clipboard.	Edit, Cut
	Copies selected objects from the form onto the Clipboard.	Edit, Copy
	Pastes the contents of the Clipboard onto the form.	Edit, Paste
	Copies formatting from selected objects to another object of similar type.	n/a
	Undoes the last change you made to the form.	Edit, Undo
	Inserts a new Hyperlink control, or allows you to edit an existing Hyperlink control.	Insert, Hyperlink
	Toggles the display of the Web toolbar.	n/a
	Displays a list of the fields in the query or table that is the data source for the main form.	View, Field List
	Displays or closes the Toolbox.	View, Toolbox
	Applies your choice of several predefined form formats, including formatting for the background bitmap of a form, text fonts, and color settings.	Format, AutoFormat

(continues)

Oracle Ex. 1009, p. 427
Oracle v. DeBellis
IPR2024-00221

Table 12.1 Continued

Button	Function	Menu Choice
	Opens the code editing window for Access VBA code contained in a module as an integral part of the form (the form's class module).	View, Code
	Displays the Properties window for one of the two sections of the form when you click the section bars, or displays the properties of a control when you select it.	View, Properties
	Displays the Build Wizard for the selected object or property in the form. This is enabled only if Access has a builder for the selected item.	n/a
	Displays the Database window.	Window, 1 Database
New Object: Table	Creates a new object. Click the arrow at the right of this button to see a drop-down list of the objects you can create.	n/a
	Displays the Microsoft Office Assistant that, in turn, displays help text related to the actions you are performing.	Help, Microsoft Access Help

The appearance of the Form Design window after clicking the Properties button is shown in Figure 12.12.

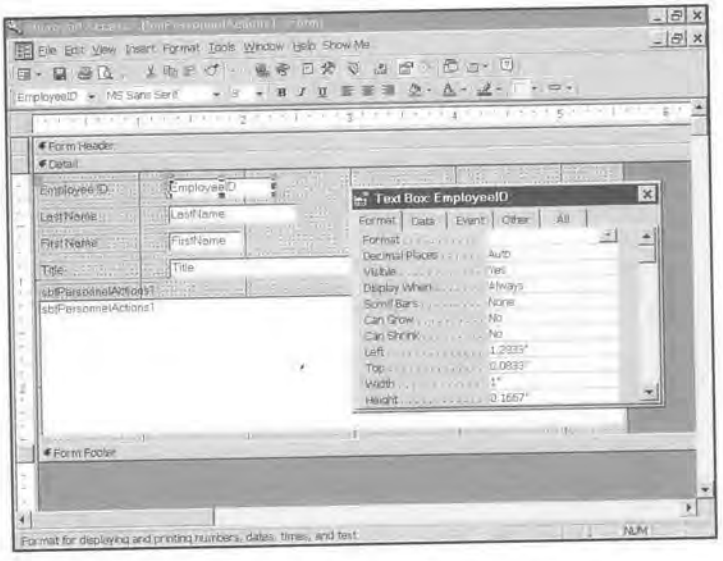


FIG. 12.12 The Form Design window with the Properties window open.

You can close the Properties window by clicking the Close Window button or by clicking the Properties button on the toolbar.

Oracle Ex. 1009, p. 428
Oracle v. DeBellis
IPR2024-00221

The Formatting Toolbar

Access 97 displays in Form and Report Design view shortcut buttons and drop-down lists for all of the text formatting, line, color, and cell effects options on a separate toolbar: the Formatting toolbar. When you select an object (such as a text box) on a form, Access enables the relevant Formatting toolbar buttons. Drop-down lists allow you to choose the desired typeface and type size for the text. Additional buttons on this toolbar, used to format the text, are similar to the buttons you find on the toolbar of Microsoft Word for Windows and other Windows word-processing applications.

Access 97 adds a new drop-down list to the toolbar, the Object list. The Object list displays the name of the currently selected object on the form and allows you to rapidly select another object on the form by selecting its name in the list. In Figure 12.12, the EmployeeID text box is the currently selected object.

The default typeface for forms is MS Sans Serif in an 8-point font. You can select any bitmapped or TrueType family that is installed on your computer from the drop-down list containing type families.

You can choose from a drop-down list of preset type sizes or type a size in points in the Size combo box. You apply attributes and formatting with the text buttons. Table 12.2 lists the function of each text-formatting button and its equivalent property setting.

Table 12.2 Toolbar Buttons for Text Controls in Form Design Mode

Button	Function	Property and Value
	Sets text style to bold (the default for titles and labels).	Font Weight = Bold
	Sets italic text style.	Font Italic = Yes
	Sets underline text style.	Font Underline = Yes
	Left-justifies text within border.	Text Align = Left
	Centers text horizontally within border.	Text Align = Center
	Right-justifies text within border.	Text Align = Right
	Displays a color palette from which you choose the background color for the selected object.	Back Color = number
	Displays a color palette from which you choose the color of the text in the selected object.	Fore Color = number

(continues)

Oracle Ex. 1009, p. 429
Oracle v. DeBellis
IPR2024-00221

Table 12.2 Continued

Button	Function	Property and Value
	Displays a color palette from which you choose the color for the border of the selected object.	Border Color = number
	Displays a drop-down list from which you choose the width of the selected object's borders. You may select a hairline width or widths ranging from 1 to 6 points.	Border Width = width
	Displays a drop-down list from which you choose a special effect for how the selected object is displayed. You may choose Flat, Raised, Sunken, Etched, Shadowed, or Chiseled.	Special Effect = name

Default Values for Forms

You can change some of the default values used in the creation of all forms by choosing Tools, Options and then selecting the Forms/Reports tab (see Figure 12.13). You can create a form to use as a template and replace the standard template, and you can determine how objects are displayed when chosen. The effects of these options are described in the sections that follow. The options that you or other Access users choose in the Options dialog are saved for each user ID in the MSysOptions table of the current System.mdw workgroup system file.

You can change the default values for the current form, section, or controls by choosing the object and then changing the default values displayed in the Properties window for that object. You can also use the AutoFormat feature to quickly apply a predefined format to all of the controls in the form. The next section describes using AutoFormat to change a form's appearance, and subsequent sections describe ways to change the format of text or controls manually on a form.

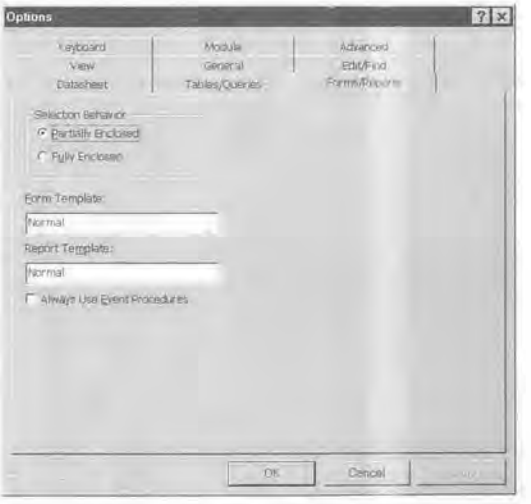


FIG. 12.13 Displaying Forms/Reports options in the Options dialog.

Oracle Ex. 1009, p. 430
Oracle v. DeBellis
IPR2024-00221

Using AutoFormat

The AutoFormat feature was new in Access 95. AutoFormat enables you to apply a predefined format to an entire form with only a few mouse clicks. Access 97 comes with several predefined formats, and you also can create your own formats for use with AutoFormat.

Applying an AutoFormat. To apply a format to a form with AutoFormat, follow these steps:

1. Click the AutoFormat button on the toolbar. Access displays the AutoFormat dialog, shown in Figure 12.14.



FIG. 12.14 Using AutoFormat to apply a predefined format to an entire form.

2. Click to select the format you want to use in the Form AutoFormats list; a preview of the format you select appears in the window in the center of the dialog.
3. Click OK to apply the format to the form. Figure 12.15 shows the frmPersonnelActions form after the International format has been applied.

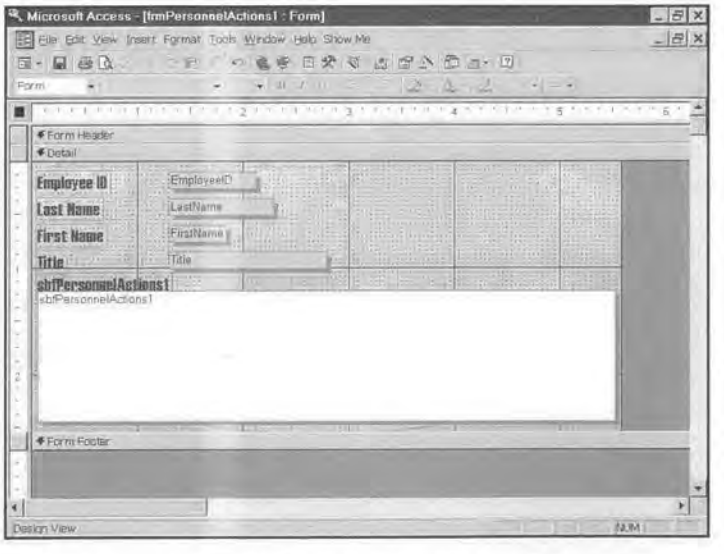


FIG. 12.15 The frmPersonnelActions form after applying the International format.

Oracle Ex. 1009, p. 431
Oracle v. DeBellis
IPR2024-00221

The AutoFormat dialog allows you to omit the application of font, color, or border style information to your form when you apply the AutoFormat. When the AutoFormat dialog is open, click the Options button; the AutoFormat dialog expands to display three additional check boxes (see Figure 12.16). Deselect the check box for the elements of the AutoFormat that you want omitted when you apply the AutoFormat to your form.



FIG. 12.16 The expanded AutoFormat dialog.

Creating, Customizing, and Deleting AutoFormats. The predefined AutoFormat styles might not suit your tastes, or you might want to create AutoFormat styles specific to your company or application.

To create a new AutoFormat or customize an existing one, follow these steps:

1. Create a form and alter its appearance (using the techniques described in the next five sections of this chapter) so that the form has the font, border, background picture, and other options adjusted exactly the way you want them for your new or customized AutoFormat.



2. Click the AutoFormat button to display the AutoFormat dialog. If you want to modify an existing AutoFormat, select it in the Form AutoFormats list now.
3. Click the Customize button to display the Customize AutoFormat dialog shown in Figure 12.17.

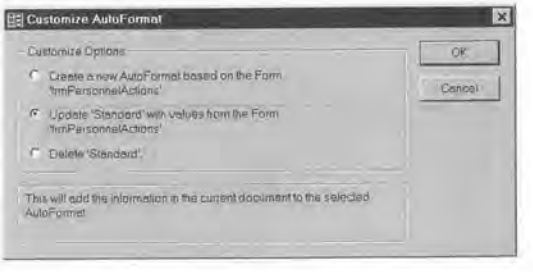


FIG. 12.17 The Customize AutoFormat dialog used to create, modify, or delete an AutoFormat.

Oracle Ex. 1009, p. 432
Oracle v. DeBellis
IPR2024-00221

4. Select the Create a new AutoFormat based on the Form *formname* option, or the Update *formatname* with values from the Form *formname* option to create or modify an AutoFormat, respectively. (Deleting AutoFormats is covered later in this section.)
5. Click OK. If you're creating a new AutoFormat, the New Style Name dialog appears (see Figure 12.18). Type an appropriate name for your new AutoFormat, and click OK. Access now creates or updates the AutoFormat and returns you to the AutoFormat dialog.
6. Click Close to close the AutoFormat dialog.



FIG. 12.18 Entering a name for a new AutoFormat.

If you have created your own AutoFormats, you may want to delete an AutoFormat that you no longer use. To delete an AutoFormat, follow these steps:



1. Open any form in Design view.
2. Click the AutoFormat button to display the AutoFormat dialog.
3. Click to select the AutoFormat you want to delete in the Form AutoFormats list, then click the Customize button. Access displays the Customize AutoFormat dialog.
4. Select the Delete *formname* option, and click OK. Access deletes that AutoFormat from the list.

> **Caution**
>
> Access does not ask for confirmation when you delete an AutoFormat; make sure to select the correct AutoFormat for deletion before you click OK.

5. Click Close to close the AutoFormat dialog.

Applying formatting to a form through an AutoFormat style is by far the easiest way to create standardized forms for your database application—especially because the Form Wizard uses the same format style list as the AutoFormat feature. In other words, any AutoFormats you create become available in the Form Wizard dialog, also.

The next few sections of this chapter describe how you can customize the appearance of various objects on a form.

Changing an Object's Colors

You select object colors through the buttons on the Formatting toolbar, as well as through property settings that are accessible through the Properties window of the form

Oracle Ex. 1009, p. 433
Oracle v. DeBellis
IPR2024-00221

and individual objects on the form. The following sections describe how to use the Formatting toolbar controls and the Property dialog to change background and foreground colors of form sections and control objects, as well as border properties of control objects.

Background Colors. The background color (Back Color property) of a form section (Header, Detail, or Footer) applies to all areas of that section except those occupied by control objects. The default background color of all sections of forms created by the Form Wizard depends on the specific form style you choose when you create the form; the Standard format scheme used to create the frmPersonnelActions form, for example, is light gray.

The default color choices on the palette displayed by the Fill/Back Color toolbar button are 16 of the standard system colors of Windows 95 and Windows NT 4.0. If you are creating a form that you intend to print, a gray or deeply textured background will not only be distracting but will also consume substantial amounts of printer toner. Data-entry operators often prefer a white background rather than a gray, colored, or textured background. Colored backgrounds hinder text visibility.

> **Note**
>
> If you have selected a picture as the background for a form—or used an AutoFormat style that includes a background picture, such as the International style pictured in Figure 12.15—then any changes you make in the background color of the form are hidden by the overlying picture.

To change the background color of a section of a form, follow these steps:

1. Click an empty area within the section of the form (Header, Detail, or Footer) whose background color you want to change. This selects the appropriate section.
2. Click the Fill/Back Color button on the toolbar to display the color palette.
3. Click the box that contains the color you want to use.

Because the background color of each form section is independent, you must repeat the process if you want to change the color for other sections of your form. The Transparent button of the Fill/Back Color palette is disabled when a form section is chosen, because a transparent background color isn't applicable to forms.

You choose the background color of a control object, such as a label, just as you do for forms. In most cases, the chosen background color of labels is the same as that of the form, so click the Transparent button to allow the background color to appear. The default value of the Back Color property of text boxes is white so that text boxes (and the data they contain) contrast with the form's background color.

Changing the Background Bitmap. You can use a bitmap picture as the background for a form. Unlike background colors, of which you can have several, you select a single bitmap picture for the entire form. Access 97 comes with a few bitmap pictures that it

Oracle Ex. 1009, p. 434
Oracle v. DeBellis
IPR2024-00221

uses in the AutoFormat formats—International, for example, uses the Globe.wmf graphics file (stored in the Program Files\Microsoft Office\Access\Bitmaps\Styles folder) as the background for the form. You can use any .wmf, .bmp, .emf, or .dib graphics file as a background for a form.

> **Note**
>
> Forms with bitmap graphics as a background can look dramatic and, therefore, are best suited for public-access information terminals or decision-support forms. These forms, however, tend to be more visually complex, which might make it difficult for users to read text labels or identify specific fields on the form. For accurate, high-speed data-entry, you should keep your transaction-processing forms visually simple so that users can easily distinguish data fields on the form and easily read text labels.

You set or remove a form's background bitmap through the Properties window of the form; you can also specify several viewing and formatting properties for the background picture. Follow these steps to set the background picture properties of a form:

1. Open the form in Design view, if necessary.
2. Click the square at the upper-left corner of the Form Design window (where the horizontal and vertical rulers meet) to select the form as a whole. A black square appears when the form is selected (see Figure 12.19).
3. If the Properties window isn't already open, click the Properties button on the toolbar to display this window.

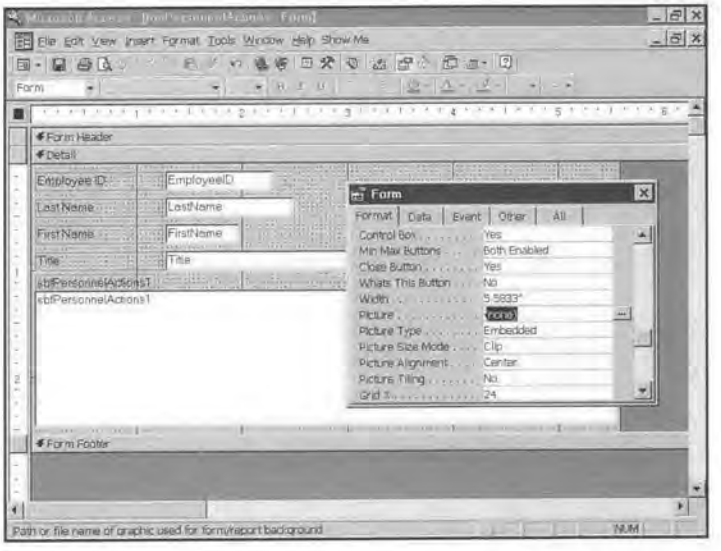


FIG. 12.19 Setting the file name and formatting properties for a form's background picture.

Oracle Ex. 1009, p. 435
Oracle v. DeBellis
IPR2024-00221

4. Click the Format tab in the Properties window, and scroll down to the end of the Format properties list to view the various Picture properties: Picture, Picture Type, Picture Size Mode, Picture Alignment, and Picture Tiling. Each of these properties and their effects are described in the list following these numbered steps.
5. Set the various Picture properties until you are satisfied with the appearance of the form. As you change each property, results of the change become immediately visible on the form.
6. Click the Close window button in the Properties window to close this window.

The following list summarizes form properties related to the background picture, available choices for each property, and the effects of each choice.

- The Picture property contains the folder path and file name of the graphics file that Access uses as the form's background. You may either type the folder path and file name directly in the Picture property text box, or you may use the builder to help you select the background graphics file. To use the builder, click the Picture property field to select that field, then click the Build button that appears next to the text box. Access displays the Insert Picture dialog shown in Figure 12.20. The Insert Picture dialog is a standard Windows 95 or Windows NT 4.0 dialog for opening files. Click the Preview button (second button from the right), if necessary, to display the background image. When you locate the graphics file you want, click to select its name, then click OK to have Access fill in the Picture property.

Tip

To remove a background picture, simply delete the entry in the Picture text box.



FIG. 12.20 The Insert Picture dialog used to select a picture as a form's background.

- The Picture Type property specifies the OLE or ActiveX method that Access uses to attach the background picture to the form. You can select either Embedded or Linked as the picture type. You usually should use the Embedded picture type,

Oracle Ex. 1009, p. 436
Oracle v. DeBellis
IPR2024-00221

especially if you intend to distribute your database application—the resulting form is self-contained and doesn't rely on the presence of external files that might be moved or deleted. If you have many forms that use the same background bitmap graphic, however, linking the background picture can save some disk space.

- The Picture Size Mode property controls how Access sizes the background picture. The available choices are Clip, Stretch, and Zoom. Clip causes Access to display the picture at its full size behind the form; if the picture is larger than the form, it is clipped to fit the form. If the picture is smaller than the form, the form's own background color shows in any part of the form background not covered by the picture. Stretch causes Access to stretch the picture vertically and horizontally to match the size of the form; the Stretch option permits distortions in the picture. Zoom causes Access to magnify the picture, without distortion, to fit the size of the form.
- The Picture Alignment property controls where Access positions the background picture. The available choices are Top-left (aligns the upper-left corner of the picture with the upper-left corner of the form window), Top-right (aligns the upper-right corner of the picture with the upper-right corner of the form window), Center (places the picture in the center of the form window), Bottom-left (aligns the lower-left corner of the picture with the lower-left corner of the form), Bottom-right (aligns the lower-right corner of the picture with the lower-right corner of the form), and Form Center (centers the picture on the form).

Tip

To ensure that a background picture is displayed relative to the form, rather than the form's window, select Form Center as the value for the Picture Alignment property.

- The Picture Tiling property has two permissible values: Yes or No. *Tiling* means that the picture is repeatedly displayed to fill the entire form or form window (if the Picture Alignment property is set to Form Center, the tiling fills just the form).

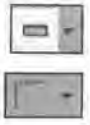
Now that you know how to adjust the background picture and colors of a form, the next section describes how to adjust the foreground colors and border properties of the form and objects on the form.

Foreground Color, Border Color, and Border Style. You may set the foreground color, border color, and border width through buttons on the Formatting toolbar or directly in the Properties window for a selected control. To set a border style (solid style or a variety of dashed-line styles), you must set the property directly in the Properties window.

Foreground color (the Fore Color property) is applicable only to control objects. (The Font/Fore Color button on the toolbar is disabled when you select a form section.) Foreground color specifies the color for the text in labels and text boxes. The default value of the Fore Color property is black. You choose border colors for control objects that have borders by using the Line/Border Color toolbar button.



Oracle Ex. 1009, p. 437
Oracle v. DeBellis
IPR2024-00221



The Special Effects button of the Formatting toolbar allows you to simulate special effects for control objects, such as a raised or sunken appearance. The Line/Border Width button allows you to control the width of the border of controls. The Formatting toolbar buttons were listed earlier in this chapter in Table 12.2. Table 12.3 lists the property name for each border property and the specific values that each may have.

To set a control's foreground color, border width, or border color by using the Formatting toolbar buttons, first click the control whose properties you want to change, then click the arrow to the right of the toolbar button for the property you want to change. Click the color or line width you want for the control.



To set a control's foreground color, border width, border color, or border style in the Properties window, first select the control whose properties you want to change by clicking it. If necessary, open the Properties window by clicking the Properties button on the toolbar. Click the Format tab in the Properties window, then scroll to the text box for the property you want to change. Most of the border properties are selected from drop-down lists; color properties require you to type a number that represents the desired color in Windows 95 or Windows NT 4.0 color notation. (Windows 95/NT 4.0 color notation is too complex to explain here; the easiest way to enter color values is with the toolbar buttons or by using the color builder described in the following section of this chapter, "Creating Custom Colors with the Color Builder.")

Table 12.3 Border Style Properties and Values

Property Name	Function	Values
Border Style	Determines the line style of the border.	Transparent, Solid, Dashes, Short Dashes, Dots, Sparse Dots, Dash Dot, Dash Dot Dot
Border Color	Sets the color of the border.	Depends on the color
Border Width	Determines the width of the border.	Hairline, or any whole point size from 1 to 6

Creating Custom Colors with the Color Builder. If you aren't satisfied with one of the 16 Windows system colors for your form sections or control objects, you can create your own custom colors by following these steps:

1. Place the caret in the Back Color, Fore Color, or Border Color text box of the Properties window for a control.
2. Click the ellipsis button to display the Color dialog. The basic form of this dialog enables you to choose from a set of 48 colors. If one of these colors suits your taste, click the color square and then click OK to assign that color as the value of the property, and close the dialog. If you want a custom color, proceed to step 3.
3. Click the Define Custom Colors button to expand the Color dialog to include the Hue/Saturation and Luminance windows, as shown in Figure 12.21.

Oracle Ex. 1009, p. 438
Oracle v. DeBellis
IPR2024-00221

4. Click and drag the cursor within the square Hue/Saturation area to choose the color you want.
5. Click and drag the arrow at the right of the rectangular luminance area while observing the Color block; release the mouse button when the Color block has the luminance (brightness) value you want.
6. Click Add to Custom Colors to add your new color to the first of the 16 custom color blocks.
7. Click the new custom color block to select it. Click OK to add this color value to the property, and close the Color dialog.

Many PCs used for data entry and editing applications run in 16-color VGA mode because this mode is slightly faster than the standard 256-color mode used by most of today's PCs. In 16-color VGA mode, any colors you choose or create, other than the standard Windows system colors, are simulated by a dithering process. *Dithering* alternates pixels of differing colors to create the usually imperfect illusion of a solid color. Thus, it is good programming practice to stick with the 16 system colors of Windows unless you have a very good reason to do otherwise.

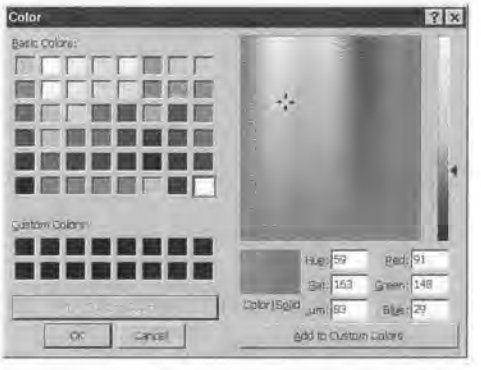


FIG. 12.21 Defining a custom color in the expanded Color dialog.

Selecting, Editing, and Moving Form Elements and Controls

The properties that apply to the entire form, to the five sections of the form, and to each control object on the form are determined by the values shown in the Properties window. To view the Properties window for a control, select the control by clicking anywhere on its surface; then click the Properties button on the toolbar. The sections of an Access form are shown in Figure 12.22.

Oracle Ex. 1009, p. 439
Oracle v. DeBellis
IPR2024-00221

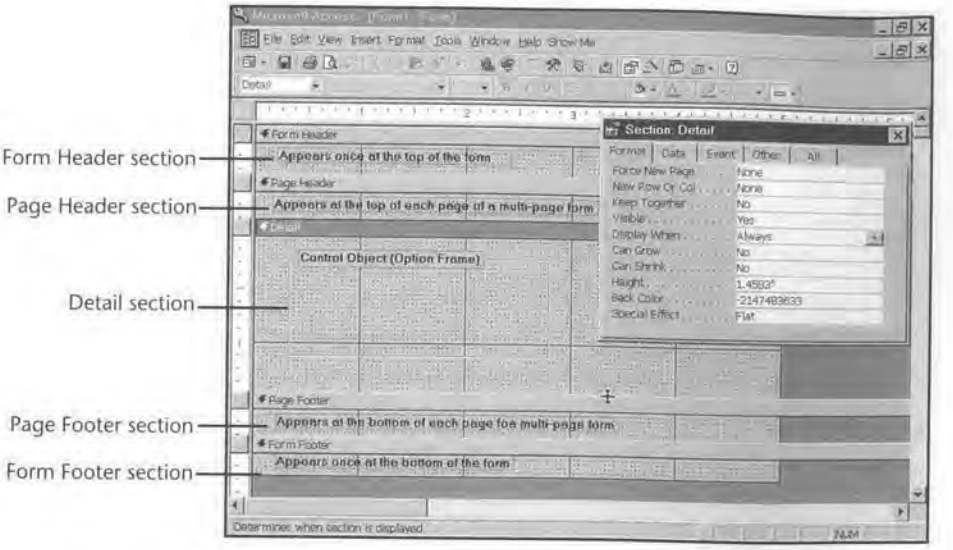


FIG. 12.22 The five sections of an Access form with the Detail section selected.

The following list describes how to choose and display the properties of the entire form, its sections, and its control objects:

- *Form.* To select the Form object, click the area of the form to the right of the right-margin indicator line. Alternatively, click the square where the ruler lines meet or choose Edit, Select Form to select the Form object. (If the Properties window isn't open, you must double-click the square where the ruler lines meet.) Selecting the form allows you to set properties for the form as a whole by entering values for the properties listed in the Properties window.
- *Header section only.* To select the Form Header, click the Form Header or Page Header bar. The set of properties you work with applies only to the Form Header or Page Header section. A Form Header and Footer appear when you choose View, Form Header/Footer. A Page Header and Footer appear when you choose View, Page Header/Footer. Page Headers and Footers primarily are used in conjunction with printing forms. You delete headers and footers by choosing View, Form Header/Footer or View, Page Header/Footer a second time.
- *Detail section only.* To select the Detail section, click the Detail bar. You get a set of properties similar to those of the Form Header section, but all of these apply to the Detail section.
- *Footer section only.* To select the Footer section, click the Form Footer or Page Footer bar. A set of properties identical to the header properties is available for the footer sections. A Form Footer appears only if a Form Header has been added. The same applies to Page Headers and Footers.
- *Control object* (or both elements of a control with an associated label). Click the surface of the control to select the control. Each type of control has its own set of

Oracle Ex. 1009, p. 440
Oracle v. DeBellis
IPR2024-00221

properties. Displaying the properties of multiple-control objects is the subject of the section "Selecting, Moving, and Sizing a Single Control" later in this chapter.

Changing the Size of the Form Header and Form Footer

You can change the height of a form section by dragging the Form Header, Page Header, Detail, Page Footer, or Form Footer bar vertically with the mouse. When you position the mouse pointer at the top edge of a section divider bar, it turns into a line with two vertical arrows (refer to Figure 12.22). You drag the pointer with the mouse to adjust the size of the section above the mouse pointer.

The height of the Detail section is determined by the vertical dimension of the window in which the form is displayed, less the combined heights of all the header and footer sections that are fixed in position. When you adjust the vertical scroll bar, only the Detail section scrolls.

Selecting, Moving, and Sizing a Single Control

When you select a control object by clicking its surface, the object is enclosed by a shadow line with an anchor rectangle at its upper-left corner and five smaller, rectangular sizing handles (see Figure 12.23).

> **Note**
>
> Text boxes, combo boxes, check boxes, and option buttons have associated (attached) labels. When you select one of these objects, the label and object are selected as a unit.

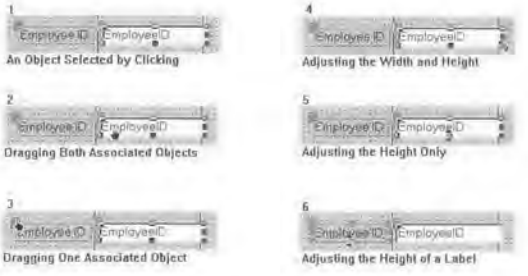


FIG. 12.23 The appearance of a control object selected for relocation and resizing.

The following choices are available for moving or changing the size of a control object (the numbered choices correspond with the numbers in Figure 12.23):

1. *To select a control (and its associated label, if any)*, click anywhere on its surface.
2. *To move the control (and its associated label, if any) to a new position*, move the mouse pointer within the outline of the object at any point other than the small resizing handles or the confines of a text box (where the cursor can become an editing caret). The mouse pointer becomes a hand symbol when it's on an area that you can use to move the entire control. Press and hold down the left mouse button while dragging the hand symbol to the new location for the control. An outline of

Oracle Ex. 1009, p. 441
Oracle v. DeBellis
IPR2024-00221

the control indicates its position as you move the mouse. When the control is where you want it to be, release the mouse button to drop the control in its new position.

Tip

If the control doesn't have an associated label, you can drag the control's anchor handle at the upper-left corner to move the control.

3. *To separately move the elements of a control that has an associated label,* position the mouse pointer on the anchor handle in the upper-left corner of the control that you want to move. The mouse pointer becomes a hand with an extended finger. Click and drag the individual element to its new position, then release the mouse button.
4. *To simultaneously adjust the width and height of a control,* click the small sizing handle at any of the three corners of the outline of the control. The mouse pointer becomes a diagonal two-headed arrow. Click and drag this arrow to a new position, then release the mouse button.
5. *To adjust only the height of the control,* click the sizing handle on one of the horizontal surfaces of the outline. The mouse pointer becomes a vertical, two-headed arrow. Click and drag this arrow to a new position, then release the mouse button.

Selecting and deselecting controls is a *toggling* process. Toggling means repeating an action with the effect of alternating between On and Off conditions. The Properties, Field List, and Toolbox buttons on the toolbar—as well as their corresponding menu choices—are toggles. The Properties window, for example, appears and disappears if you repeatedly click the Properties button.

Aligning Controls to the Grid

The Form Design window includes a grid that consists of one-pixel dots with a default spacing of 24 to the inch horizontally and 24 to the inch vertically. When the grid is visible, you can use the grid dots to assist in maintaining the horizontal and vertical alignment of rows and columns of controls. Even if the grid isn't visible, you can cause controls to "snap to the grid" by choosing Format, Snap to Grid. This menu command is a toggle, and when Snap to Grid is active, the menu choice is checked. Whenever you move a control while Snap to Grid is active, the upper-left corner of the object jumps to the closest grid dot.

You can cause the size of control objects to conform to grid spacing by choosing Format, Size, To Grid. You also can make the size of the control fit its content by choosing Format, Size, To Fit.

Tip

If Snap to Grid is on and you want to locate or size a control without reference to the grid, press and hold the Ctrl key while you move or resize the control.

Oracle Ex. 1009, p. 442
Oracle v. DeBellis
IPR2024-00221

Toggling the View, Grid menu command controls the visibility of the grid; by default, the grid is visible for all new forms. If the grid spacing is set to more than 24 per inch or 10 per centimeter, the dots aren't visible. To change the grid spacing for a form, follow these steps:

1. Choose Edit, Select Form.
2. Click the Properties button on the toolbar to make the form properties appear.



3. Click the Format tab in the Properties window to display the formatting properties, and scroll through the list until the Grid X and Grid Y properties are visible.
4. Change the value of Grid X to 10 dots per inch (dpi) and Grid Y to 12 dpi, or change both values to 16 (if you want controls to align with inch ruler ticks). Users with metric rulers are likely to prefer a value of 10 for both Grid X and Grid Y.

Tip

The default values of grid spacing of Access 97 is 24 dpi. Better values are 10 for Grid X and 12 for Grid Y. This grid dot spacing is optimum for text controls that use the default 8-point MS Sans Serif font. The Form Wizards also use 24 dpi horizontally and vertically.

Selecting and Moving a Group of Controls

You can select and move more than one object at a time by using one of the following methods:

- *Enclose the objects with a rectangle.* Begin by clicking the surface of the form outside the outline of a control object. Press and hold down the mouse button while dragging the mouse pointer to create an enclosing rectangle that includes each of the objects you want to select (see Figure 12.24). Release the mouse button. You may now move the group of objects by clicking and dragging the anchor handle of any one of them.
- *Click to select one object, then hold down the Shift key while you click to select the next object.* You can repeat this step as many times as necessary to select all of the objects you want.
- *To remove a selected object from a group,* hold down the Shift key, and click the object with the mouse to deselect it. To deselect an entire group, click any inactive area of the form. An inactive area is an area outside the outline of a control.



FIG. 12.24 Selecting a group of objects by dragging a selection rectangle.

Oracle Ex. 1009, p. 443
Oracle v. DeBellis
IPR2024-00221

If you select or deselect a control with an associated label, the label is selected or deselected along with the control.

> **Note**
>
> The selection rectangle selects a control if any part of the control is included within the rectangle. This behavior is unlike many drawing applications in which the entire object must be enclosed to be selected. You can change the behavior of Access's selection rectangle to require full enclosure of the object by choosing Tools, Options, selecting the Forms/Reports tab, and changing the value of the Selection Behavior option from Partially Enclosed to Fully Enclosed.

Aligning a Group of Controls

You can align selected individual controls, or groups of controls, to the grid or each other by choosing Format, Align and completing the following actions:

- To align a selected control (or group of controls) to the grid, choose To Grid from the submenu.
- To adjust the positions of controls within a selected columnar group so that their left edges fall into vertical alignment with the far-left control, choose Left from the submenu.
- To adjust the positions of controls within a selected columnar group so that their right edges fall into vertical alignment with the right edge of the far-right control, choose Right from the submenu.
- To align rows of controls at their top edges, choose Top from the submenu.
- To align rows of controls at their bottom edges, choose Bottom from the submenu.

Your forms have a more professional appearance if you take the time to align groups of controls vertically and horizontally.

Using the Windows Clipboard and Deleting Controls

All of the conventional Windows Clipboard commands apply to control objects. You can cut or copy a selected control or group of controls to the Clipboard. After that, you can paste the control or group to the form using Edit menu commands and then relocate the pasted control or group as desired. Access uses the Windows 95 and Windows NT 4.0 keyboard shortcut keys: Ctrl+X to cut, Ctrl+C to copy selected controls to the Clipboard, and Ctrl+V to paste the Clipboard contents. The traditional Shift+Delete, Ctrl+Insert, and Shift+Insert commands perform the same operations.

You can delete a control by selecting it and then pressing Delete. If you accidentally delete a label associated with a control, do the following: select another label, copy it to the Clipboard, select the control with which the label needs to be associated, and paste the label to the control.

Changing the Color and Border Style of a Control

As mentioned earlier in this chapter, the default color for the text and borders of controls is black. Borders are one pixel wide (called *hairline* width). Some objects, such as text

Oracle Ex. 1009, p. 444
Oracle v. DeBellis
IPR2024-00221

boxes, have default borders. Labels have a gray background color by default, but a better choice for the default label color would have been transparent. *Transparent* means that the background color of the object under the control (the form section, in this case) appears within the control except in areas of the control that are occupied by text or pictures.

You control the color and border widths of a control from the Line/Border Color and Line/Border Width buttons on the Formatting toolbar. You must select a border style directly in the Properties window.

To change the color or border width of a selected control or group of controls, follow these steps:

1. Select the control(s) whose color or border width you want to change.
2. Click the arrow of the Fill/Back Color toolbar button to open the color palette popup window. Click the color square you want, or click the Transparent button to make the background transparent.



3. Click the arrow of the Line/Border Color toolbar button to open the color palette popup window, where you change the border color for any selected control with borders.



4. Click the arrow of the Line/Border Width toolbar button to open the border width popup window, where you change the thickness of the border for any selected control whose borders are enabled.



5. Click the arrow of the Font/Fore Color toolbar button to open the color palette popup window, where you change the color of the text of selected controls.



Note

The general practice for Windows database entry forms is to indicate editable elements with borders and clear backgrounds. Still, some popular software (most notably, the DOS versions of dBASE and Paradox) uses reverse video as the default to indicate editable text. You can create the effect of reverse video by choosing black or another dark color for the fill of a text box control and a light color for its text. If you decide to implement reverse text, remember that reverse text is more difficult to read than normal text, and consider using a larger font and adding the bold attribute to ensure legibility.

To set the border style, you must select the Border Style property directly in the Properties window, as explained earlier in this chapter.

Changing the Content of Text Controls

You can edit the content of text controls by using conventional Windows text-editing techniques. When you place the mouse pointer within the confines of a text control and click the mouse button, the mouse pointer becomes the Windows text-editing caret that you use to insert or delete text. You can select text by dragging the mouse over it or by holding down Shift and moving the caret with the arrow keys. All Windows Clipboard operations are applicable to text within controls. Keyboard text selection and editing techniques using the arrow keys in combination with Shift are available, also.

Oracle Ex. 1009, p. 445
Oracle v. DeBellis
IPR2024-00221

If you change the name of a field in a text box and make an error naming the field, you receive a `#Name?` error message in the offending text box when you select Run mode. Following is a better method of changing a text box with an associated label:

1. Delete the existing field control by clicking to select it and then pressing Delete.
2. Click the Field List button in the Properties bar to display the Field List dialog.
3. Scroll through the entries in the list until you find the field name you want.
4. Click the field name to select it, then drag the field name to the location of the deleted control. Release the mouse button to drop the new name.
5. Close the Field List dialog when you are finished.

You can relocate and resize the new field caption and text box (or edit the caption) as necessary.

Using the Format Painter

The Format Painter allows you to quickly copy the format of any control on the form to any other control on the form. The Format Painter copies only those formatting properties that are relevant to the control on which you apply the Format Painter. To use the Format Painter, follow these steps:

1. Select the control with the formatting you want to copy.
2. Click or double-click the Format Painter button on the toolbar; the mouse cursor changes to a pointing arrow with a paintbrush icon attached to it. (Double-clicking "locks" the Format Painter on. Double-click the Format Painter button only if you want to copy the formatting to more than one control.)
3. Click any control that you want to copy the formatting to; the Format Painter copies all relevant formatting properties to this control. If you didn't double-click the Format Painter button, the Format Painter turns itself off after copying the formatting properties to one control.
4. If you locked the Format Painter on by double-clicking its button, you can repeat step 3 as many times as you want. Click the Format Painter button again to turn the Format Painter off.

Typically, you use the Format Painter to quickly set the formatting properties for field text labels, or in any situation where selecting several controls by dragging a selection rectangle seems undesirable. By locking the Format Painter on, it's easy to format several controls one after another.

Rearranging the Personnel Actions Form

The objective of the following set of instructions is to rearrange the controls on the frmPersonnelActions form so that all of the elements on the form (and its subform) are completely visible in the form window. The objective of these instructions is also to optimize the position of the fields for data entry. After you complete the following steps, your main form appears as shown in Figure 12.25, and your subform appears as shown in Figure 12.26.

Oracle Ex. 1009, p. 446
Oracle v. DeBellis
IPR2024-00221

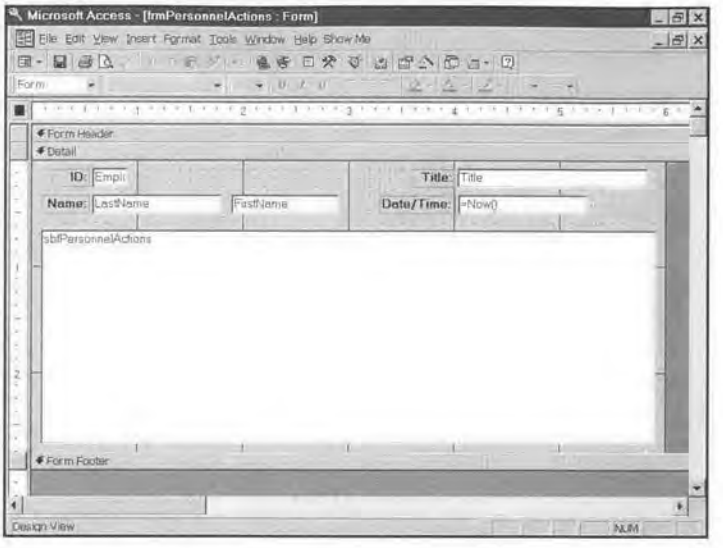


FIG. 12.25 The frmPersonnelActions form after relocating and resizing its control objects.

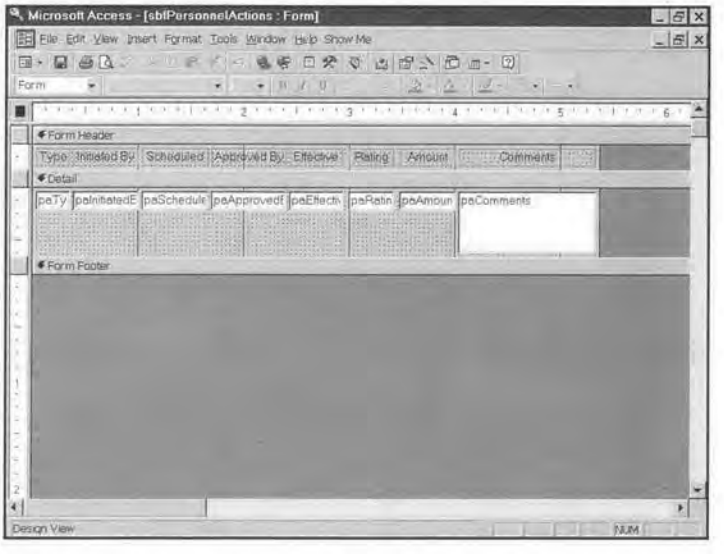


FIG. 12.26 The sbfPersonnelActions subform after modifying its appearance.

Setting Properties of the Main Form

To change the color of form objects and rearrange the controls of the frmPersonnelActions form to correspond with the positions shown in Figure 12.25, follow these steps:

Oracle Ex. 1009, p. 447
Oracle v. DeBellis
IPR2024-00221

1. Close the frmPersonnelActions form by clicking the Close Window button. Do not save any changes you made in the preceding section.



2. Choose frmPersonnelActions from the Forms list in the Database window, and click the Design button.

3. Click the Maximize window button to maximize the Form Design window if it isn't already maximized.



4. Choose Edit, Select Form, and then click the Properties button on the toolbar.

5. Click the Format tab of the Properties window, and then scroll through the properties list until you see Grid X and Grid Y. Change the Grid X property to 10 and the Grid Y property value to 12. (Metric users may prefer a 5-by-5 grid, providing 2 mm resolution.)

6. Close the Properties window by clicking the Properties button on the toolbar again.

7. Drag the right margin of the form from its present position (5.5 inches) to 6 inches.

8. Click the Title field text box to select the text box and its label.

9. Move the mouse pointer onto the selected Title field until the pointer changes to the shape of a hand. Click and drag the Title field to the right of the EmployeeID text box.

10. Delete the FirstName label (click the label, then press Delete). Next, use the technique described in steps 8 and 9 to select the FirstName field and drag it to a position to the right of the LastName field (refer to Figure 12.25).

11. Edit the LastName label to read **Name:**, the EmployeeID label to read **ID:**, and the Title label to read **Title:**.

12. Delete the sbfPersonnelActions field label (the size and content of the subform is sufficient to identify it), and drag the subform control to a position below the FirstName and LastName fields (refer to Figure 12.25).

13. Click and drag the Form Footer bar to approximately 2.75 inches. (Alternatively, you can type the detail section's height directly in the Height property on the Format sheet of the Properties window.) At present, the dimensions of your form are 6×2.7 inches.

14. Resize the sbfPersonnelActions subform control on the form so that its left, right, and bottom edges are one grid mark inside the edges of the form (this makes the sbfPersonnelActions subform control about 5.8×1 7/8 inches).





15. Click the text label of the EmployeeID field to select it, and then click the Bold and Align Right buttons on the Formatting toolbar to make the text label bold and right justified.



16. Double-click the Format Painter button on the toolbar (remember that this locks the Format Painter).

Oracle Ex. 1009, p. 448
Oracle v. DeBellis
IPR2024-00221

17. In turn, click the text labels for all of the remaining controls on the form to apply the formatting with the Format Painter.
18. Click the Format Painter button on the toolbar again to turn off the Format Painter.
19. Adjust the widths of the labels and text boxes to suit their content (refer to Figure 12.25).
20. Click the Save button on the toolbar (or choose File, Save) to save your changes to the frmPersonnelActions form.



You may need to adjust the sizes of some controls individually to make their appearance consistent with other controls. When you complete your rearrangement, click the Form View button. Your form appears as shown in Figure 12.27.

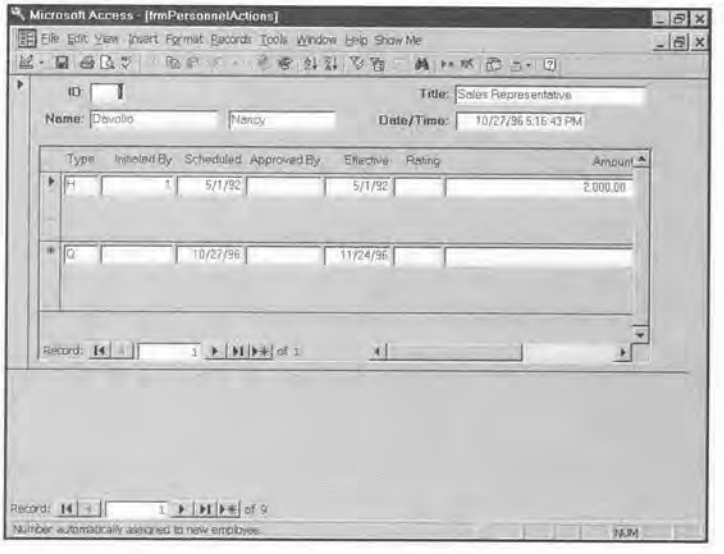


FIG. 12.27 The revised frmPersonnelActions form in Run mode.

Setting the Properties of a Subform

You can learn about modifying the properties of a subform by working with the subform used to create the history of prior Personnel Actions for an employee. In this example, editing or deleting entries using the subform is not allowed, but you can add new entries. The subform needs to be modified so that all of its columns are readable without horizontal scrolling. To change the properties of the Personnel Actions subform, follow these steps:

1. Close the frmPersonnelActions form.
2. Open the sbfPersonnelActions subform from the Database window in Design view.

Oracle Ex. 1009, p. 449
Oracle v. DeBellis
IPR2024-00221



3. Select the form and use the Properties window to make sure that the Grid X and Grid Y properties are both set to 24. Close the Properties window.
4. Using the same techniques you used when working with the main form, resize the label boxes in the Form Header section of the form so that they match what you saw in Figure 12.26. Use the Format Painter to center the text in every text label in the Form Header section.
5. Adjust the field text boxes in the Detail section of the form, if necessary, to line up with the headings in the Form Header section (refer to Figure 12.26).
6. Drag the right edge of the form to the left until the form is 5 3/8 inches in width, and then drag the Form Footer upward so that the Detail section is about 5/8 inches high.



7. Choose Edit, Select Form to select the form, then click the Properties button on the toolbar to display the Properties window for the subform.
8. Click the Data tab in the Properties window so that the Allow Edits, Allow Deletions, and Allow Additions properties are visible.
9. Set the Allow Edits property to **No**; this prevents the user from editing records displayed in this subform.
10. Set the Allow Deletions property to **No**; this prevents the user from deleting records displayed in this subform.
11. Set the Allow Additions property to **Yes**; this enables the user to add new records in this subform.
12. Close the sbfPersonnelActions subform and save your changes.

To see how the new form and subform look, open the frmPersonnelActions form; the adjusted form and subform should appear similar to Figure 12.28. Notice that there's no horizontal scroll bar, and that the appearance and visibility of fields and column headings in the subform has improved. By changing the size of the subform control in the main form and resizing the subform to fit completely within the subform control (allowing room for the vertical scrollbar), the subform now fits completely in the main form. Notice also the tentative append record that is visible as the second record in the subform.

Note

You can set the Data Entry property to Yes to achieve a result that is similar to setting the Allow Edits and Allow Deletions property to No and the Allow Additions property to Yes. When you set the Data Entry property to Yes, however, only the tentative new record appears—no prior entries appear in the subform.

You can change the default view of the subform to a continuous form that you can modify to display the data in another format. When displaying historical data, Datasheet view (a tabular view) is usually the best choice because it is the easiest to implement.

Oracle Ex. 1009, p. 450
Oracle v. DeBellis
IPR2024-00221

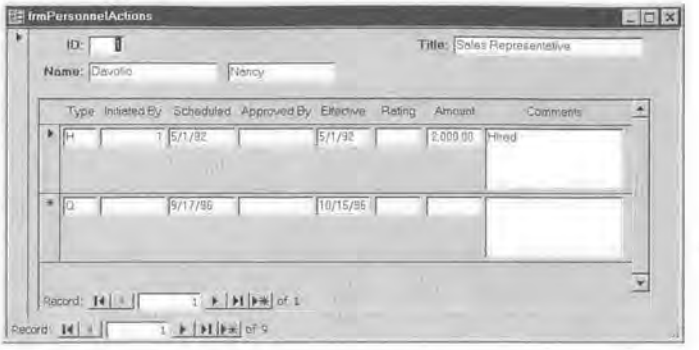


FIG. 12.28 The completed frmPersonnelActions and sbfPersonnelActions forms in Form view displayed in normal window mode.

Using Transaction-Processing Forms

As noted near the beginning of this chapter, the purpose of transaction-processing forms is to add new records to, delete records from, or edit data in one or more tables that underlie the form. This section describes how to add new records to the Personnel Actions table with the frmPersonnelActions form.

Toolbar Buttons in Form View

When you display your form in Run mode (Form view), the toolbar contains the command buttons listed in Table 12.4. This table lists all of the buttons that appear on the toolbar, along with each button's function and the equivalent menu choice.

Table 12.4 Standard Toolbar Buttons in Form Run Mode

Button	Function	Menu Choice
	Selects Form Design mode.	View, Design View
	Saves the form layout.	File, Save
	Prints the form.	File, Print
	Selects Print Preview to display how your form will appear if printed. You can print the form directly from the Print Preview window.	File, Print Preview
	Starts the spelling checker to check the spelling of the current selection or field.	Tools, Spelling
	Cuts the current selection and places it on the Windows Clipboard.	Edit, Cut

(continues)

Oracle Ex. 1009, p. 451
Oracle v. DeBellis
IPR2024-00221

Table 12.4 Continued

Button	Function	Menu Choice
	Copies the current selection to the Windows Clipboard.	Edit, Copy
	Pastes the current contents of the Windows Clipboard onto the form.	Edit, Paste
	Format Painter. This button is always disabled in Form view.	n/a
	Undoes the most recent change to a record.	Edit, Undo
	Inserts a new Hyperlink or allows you to edit an existing Hyperlink.	Insert, Hyperlink
	Toggles the display of the Web Toolbar.	n/a
	Sorts records in ascending order based on the current field.	Records, Sort, Ascending
	Sorts records in descending order based on the current field.	Records, Sort, Descending
	Filters records based on selected text in a field.	Records, Filter, Filter by Selection
	Filters records based on criteria you enter in a form's fields.	Records, Filter, Filter by Form
	Applies a filter. Click this button a second time to show all records.	Records, Apply Filter/Sort or Records, Remove Filter/Sort
	Searches for a value in the selected field or in all fields. Displays the Find dialog.	Edit, Find
	Goes to the tentative append record.	Edit, Go To, New
	Deletes the current record.	Edit, Delete Record
	Gives the Database window the focus.	Window 1 Database

Oracle Ex. 1009, p. 452
Oracle v. DeBellis
IPR2024-00221

Button	Function	Menu Choice
	Displays a drop-down list of new database objects.	n/a
	Displays the Microsoft Office Assistant which, in turn, displays help text related to the actions you are performing.	Help, Microsoft Access Help

The Find button serves the same purpose for forms in Run mode as it does for tables and queries. You type characters in the Find dialog using wild cards, if needed. When you execute the search, Access displays the first record that matches your entry.

The Sort Ascending, Sort Descending, Filter by Selection, and Filter by Form buttons work the same way in Form view as they do in Datasheet view. Using these filter buttons was described in Chapter 6, "Sorting, Finding, and Filtering Data in Tables." Sorting specified in the form overrides the sort criteria of the primary query used as the source of the data (if your form is based on a query, rather than directly on one or more tables). The filter or sort criteria you specify do not take effect until you click the Apply Filter/Sort button or make the equivalent Records, Apply Filter/Sort menu choice.

Using the Personnel Actions Form

Forms you create with the Form Wizard use the standard record-selection buttons located at the bottom of the form. The record-selection buttons perform the same functions with forms as they do with tables and queries. You can select the first or last records in the table or query that is the source of data for your main form, or you can select the next or previous record. Subforms always include their own set of record-selection buttons that operate independently of the set for the main form.

Navigation between the text boxes used for entering or editing data in the form is similar to navigation in queries and tables in Datasheet view, except that the up-arrow and down-arrow keys cause the caret to move between fields rather than between records. Accept the values you've entered by pressing Enter or Tab.

To edit records in a table in Form view, the Allow Editing property must be set to Yes (which is the default). To append new records to a table in Form view, the Allow Additions property must be set to Yes (which is also the default). You can only change these properties by using the Properties window for the form in Form Design view (as you did earlier in this chapter for the subform).

Appending New Records to the Personnel Actions Table

In Datasheet view of a table or query, the last record in the datasheet is provided as a *tentative append record* (indicated by an asterisk on the record-selection button). If you enter data in this record, the data automatically is appended to the table, and Access starts a new tentative append record. Forms also provide a tentative append record, unless you have set the Allow Additions property for the form to No.

Oracle Ex. 1009, p. 453
Oracle v. DeBellis
IPR2024-00221

To append a new record to the Personnel Actions table and enter the required data, follow these steps:



1. Open the frmPersonnelActions form if it is not already open, or click the Form View button if you are in Design view. Data for the first record of the Employees table—with the matching data from the corresponding record(s) in the Personnel Actions table—appears in the text-box controls of your form.

   Because data from the Employees table is included in the main form, the ID number, name, and title of the employee appear in the text boxes on the main form. Access lets you edit the LastName, FirstName, and Title data, although these fields are incorporated in the table (Employees) on the one side of a one-to-many relationship. The editing capability of a form is the same as that for the underlying table or query that serves as its source, unless you change the form's editing capabilities by setting the form's Allow Editing property and other related properties.

   If you added an entry for the chosen employee ID when you created the Personnel Actions table in Chapter 4, "Working with Access Databases and Tables," the entry appears in the subform's fields. The subform's data display is linked to the data in the main form through the one-to-many relationship between the Employees table and the Personnel Actions table. The subform only displays records from the Personnel Actions table whose paID fields match the value of the EmployeeID field of the record currently displayed by the main form.



2. Access places the caret in the first text box of the main form, the ID text box. The first example uses Steven Buchanan, whose employee ID is 5, so you should do the following: click the Find button on the toolbar to open the Find dialog, type **5** in the Find What text box, make sure that the Search Only Current Field option is selected, and click Find First. Access displays the Employees table data for Steven Buchanan in the main form and his Personnel Actions records in the subform. Click Close to close the Find dialog.





3. Click in the Type field of the tentative append record in the subform. If the tentative append record in the subform isn't visible, click in any field in the subform, then click the New Record button on the toolbar to move to the tentative append record at the end of the existing Personnel Actions table entries for Steven Buchanan.

4. The controls of the tentative append record in the subform become empty, except for those fields with default values assigned to them.

5. Access places the caret in the first text box of the subform, the Type (paType field) text box. Type a valid Personnel Action type (H, S, Q, Y, B, or C, because of the field's validation rule) in the Type text box. In this example, you bring Steven Buchanan's Personnel Actions records up to date by adding quarterly performance review information, so you can stick with the default value, **Q**. Press Tab or Enter to accept the Type and move the caret to the next data-entry text box, Initiated By.

Oracle Ex. 1009, p. 454
Oracle v. DeBellis
IPR2024-00221

6. Mr. Buchanan reports to the Vice President of Sales, Andrew Fuller, whose employee ID is 2. Type **2** in the Initiated By text box and press Enter.



The pencil symbol, which indicates that you are editing a record, replaces the triangle at the top of the Record Selector bar to the left of the record that you are entering. The Description property you entered for the field in the table underlying this query appears in the status bar and changes as you move the caret to the next field. (To change a previous entry, press Shift+Tab, or use the up- and down-arrow keys to maneuver to whichever text box contains a value you want to change.)

7. Mr. Buchanan was hired on 10/17/93, so his first quarterly performance review should be dated about three months after that. Therefore, use a date near **1/2/94** for the Scheduled text box. Today's date is the default for the paScheduled date. Edit the date, or press F2 to select the default date, and then replace the default value with a new date.

8. Because Mr. Fuller is a vice president, he has the authority to approve salary increases. Type Mr. Fuller's employee ID, **2**, in the Approved By text box, and press Enter to move the caret to the next field.

9. The effective date for salary adjustments for Northwind Traders is the first or 15th day of the month in which the performance review is scheduled. Type the appropriate date in the Effective text box.

10. You can type any number from **0** (terminated) to **9** (excellent) in the Rating text box, which reflects the employee's performance.

11. You can be as generous as you want with the salary increase that you enter in the New Amount text box. The value of the New Amount is a new monthly salary (or a new commission percentage), not an incremental value.

12. In the Comments multiline text box to the right of the New Amount field, add any comments you care to make concerning how generous or stingy you were with this salary increase. The multiline text box includes a scroll bar that appears when the caret is within the text box.

13. When you complete your entries, Access stores them in a memory buffer but does not add the new record to the Personnel Actions table. You can add the record to the table by doing any of the following: pressing Shift+Enter; choosing Records, Save Record; clicking the New Record button; or changing the position of the record pointer with the Prior or Next record selector button. If you want to cancel the addition of a record, press Esc.

14. Repeat steps 3 through 13 to add a few additional records.

Tip

If you ever click the New Record button on the toolbar (or the Next record selector button) and decide that you don't want to add any more data, simply click the Prior button to make sure this new record is not added to the table.

Oracle Ex. 1009, p. 455
Oracle v. DeBellis
IPR2024-00221

After you add several records, your form appears like the one shown in Figure 12.29. Each record for an employee appears in the subform datasheet in the order of the primary key fields of the Personnel Actions table.

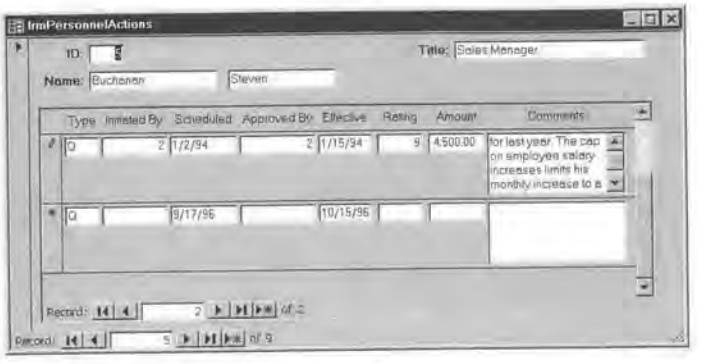


FIG. 12.29 The Personnel Actions form after appending subform records for a single employee.

The key fields of the Personnel Actions table are paID, paType, and paScheduledDate; duplicate values of the combination of the three fields aren't allowed. If you try to enter a duplicate of another record, the message box shown in Figure 12.30 appears. If you still want to add the new record, you must change the Type or Scheduled Date entry to be different from the existing record.

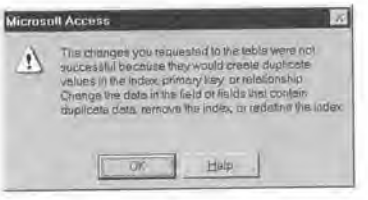


FIG. 12.30 The message box that results when you try to complete a duplicate entry.

Editing Existing Data






You can edit existing records the same way you add new records. Use the Next button to find the record you want to edit, and then make your changes. You can use the toolbar's Find button to locate records by employee ID, by one of the dates in the record, or by a word or phrase contained in the paComments field. If you prefer that the records be ordered by paEffective date to find all records for which an effective date hasn't been entered, use the Filter by Form button and specify an Ascending Sort on the paEffective field. Click the Apply Filter button to apply the sort to the records.

Committing and Rolling Back Changes to Tables

As with tentative append records, Access does not apply record edits to the underlying table until you move the record pointer with the record-selection buttons (or choose Records, Save Record). Either action is the equivalent of the `CommitTrans` instruction in transaction-processing terminology.

Oracle Ex. 1009, p. 456
Oracle v. DeBellis
IPR2024-00221

`Rollback` reverses a `CommitTrans` instruction. You can do the equivalent of rolling back a single transaction by clicking the Undo button on the toolbar immediately after you save the record to the table (or by choosing Edit, Undo Saved Record if that choice is available).



Deleting a Record in Form View

To delete the current record underlying the form (either the main form or a subform), click the Delete Record button on the toolbar (or choose Edit, Delete Record). Access displays the message box, shown in Figure 12.31, and asks you to confirm the permanent removal of the record you selected. Click Yes to delete the record.



Access also allows you to delete records by clicking the vertical record selection bar to the left of the Detail section of the form. The vertical record selection bar becomes a darker shade of gray, and the triangle that is used to identify a selected record appears at the top of the bar. Press Delete, and the confirmation message appears. Click Yes to delete the record, or click No if you've changed your mind.

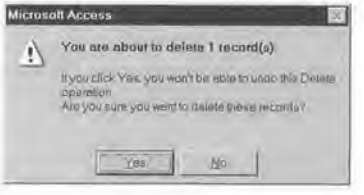


FIG. 12.31 The message box that asks you to confirm or cancel the deletion of a record.

Note

You can't use the sbfPersonnelActions form to delete records from the Personnel Actions table because you set the Allow Deletions property to No when you modified the subform earlier in this chapter. The main form won't permit you to delete records from the Employees table, either, because the Employees table has dependent records in other tables.

Furthermore, the Enforce Referential integrity option for the join between the Employees and Personnel Actions tables is enabled without cascading deletes also being enabled. Access therefore prevents the deletion of records in the Employees table if it would create orphan records in another table. If you want to experiment with deleting records from the Personnel Actions table in the subform, open the sbfPersonnelActions form in Design mode, and change the Allow Deletions property (on the Data tab of the Properties window for the form) to Yes.

Modifying the Properties of a Form or Control After Testing

The entries you added and edited gave you an opportunity to test your form. Testing a form to ensure that it accomplishes the objectives you have in mind usually takes much longer than creating the form and the query that underlies it. During the testing process, you might notice that the order of the fields isn't what you want, or that records in the subform aren't displayed in an appropriate sequence. The following two sections deal with modifying the properties of the form and subform control.

Oracle Ex. 1009, p. 457
Oracle v. DeBellis
IPR2024-00221

Changing the Order of Fields for Data Entry

The order in which the editing caret moves from one field to the next is determined by the Tab Order property of each control. The Form Wizard established the tab order of the controls when you first created the form. The default Tab Order property of each field is assigned, beginning with the value 0, in the sequence in which you add the fields. Because the Form Wizard created a single-column form, the order of the controls in Personnel Actions is top to bottom. The tab order originally assigned doesn't change when you relocate a control.

To change the sequence of entries—for example, to match the pattern of entries on a paper form—follow these steps:



1. Click the Design View button on the toolbar.
2. Choose View, Tab Order to display the Tab Order dialog shown in Figure 12.32. The order of entry is shown by the sequence of field names in the Custom Order list. (In this example, changing the sequence of the entries is unnecessary because the sequence is logical, even after moving the controls to their present locations on the Personnel Actions form.)
3. Click the Auto Order button if you want to reorder the entry sequence going left to right across each row of fields, then top to bottom.
4. Drag any control to a new location by clicking the button at the left of its name and dropping it wherever you want it to be in the sequence.
5. Click OK to implement the changes you made; click Cancel to retain the original entry sequence.

> **Note**
>
> Using the Auto Order button to change the tab order of fields on a form also changes the left-to-right order of the table fields in Datasheet view to correspond to the Auto Order field sequence.



FIG. 12.32 Changing the sequence of data-entry fields in the Tab Order dialog.

Oracle Ex. 1009, p. 458
Oracle v. DeBellis
IPR2024-00221

Removing Fields from the Tab Order

Access 97 allows you to set the value of the Tab Stop property to No in order to prevent controls from receiving the focus in the tab order. To remove a control from the tab order, select the control; open the Properties window; select Other Properties; and change the value of Tab Stop to **No** (see Figure 12.33). You cannot edit the EmployeeID field, so set the Tab Stop property to No for this control.

> **Note**
>
> Setting the Tab Stop property's value to No does not disable a given control, but it removes the control from the tab sequence. This means that the control can't be selected by pressing the Tab key, but can still be selected by clicking it with the mouse.

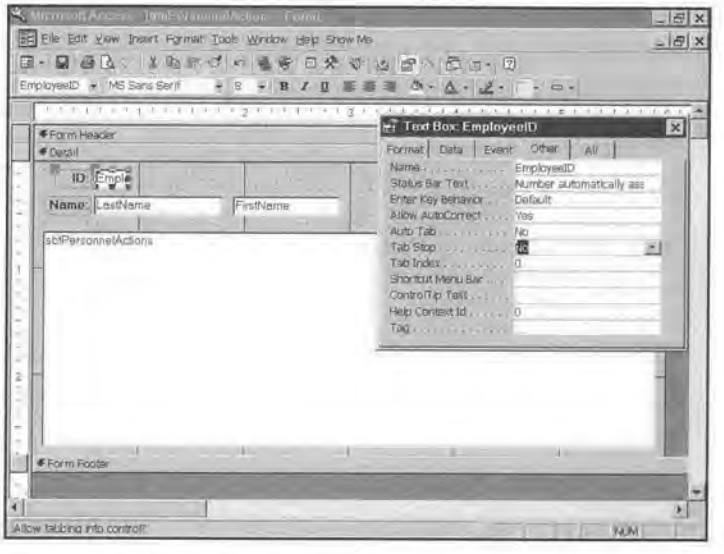


FIG. 12.33 Removing a field from the tab order.

Changing the Source of Data for the Subform and Sorting the Subform Data

The Personnel Actions table is indexed by paType and paScheduledDate, but the sequence of the records appears in the order of the primary key: paID, paType, and paScheduledDate. Because only paID is used to link the records, records for a particular employee appear in the order of the remaining key fields, paType and paScheduledDate. Eventually, the number of records for an employee can become quite large, so having the records appear in type and date order is convenient. Because only a few records can be displayed in the subform datasheet, the latest entries should appear by default. This requires a descending sort on the Personnel Actions table. You can establish a descending sort only by substituting a sorted query that contains all records of the Personnel Actions table, as described in the following two procedures.

Oracle Ex. 1009, p. 459
Oracle v. DeBellis
IPR2024-00221

To create a new sorted query, follow these steps:



1. Close the Personnel Action entry form, and make the Database window active.



2. Click the Queries tab of the Database window, and then click the New button to create a new query. With Design view selected in the New Query dialog, click OK.
3. In the Show Table dialog, select Personnel Actions as the table on which to base the query; click Add and then click Close.
4. Drag the * field to the Query grid, and drop it in the first column.
5. Drag the paScheduledDate field to the Query grid, and drop it in the second column.
6. Click to remove the check mark from the Show box in the new paScheduledDate column so that you don't duplicate a field name in the query. Add a descending sort to the paScheduledDate column (see Figure 12.34).

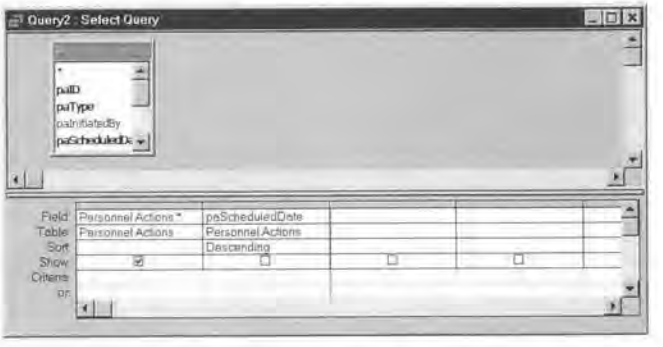


FIG. 12.34 The query design for the subform with all of the personnel actions sorted in descending date sequence.



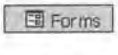
7. Click the Run Query button to check to see that the records are sorted in reverse chronological order.
8. Close your query and name it **qryPersonnelActionsSubform**.

To change the data source for the subform to the new sorted query, complete the following steps:



1. Click the Forms tab in the Database window, and open the sbfPersonnelActions subform in Design mode.





2. Click the Properties button to display the Properties window, then click the Data tab.



3. Click the Record Source box, open the list, and select `qryPersonnelActionsSubform` as the new data source for the subform (see Figure 12.35).

Oracle Ex. 1009, p. 460
Oracle v. DeBellis
IPR2024-00221

4. If you want to review the design of the table or query you select as the value of the Record Source property, click the ellipsis button to display the source object in Design mode.
5. Close the Properties window, and click the Form View button on the toolbar to verify that the datasheet display is correct.



6. Close the sbfPersonnelActions subform, and save your changes.
7. Reopen the Personnel Actions form. Use Find in the main part of the form to locate the employee for whom you added additional records in the Personnel Actions table. The history in the subform is now sorted with the most recent dates first.

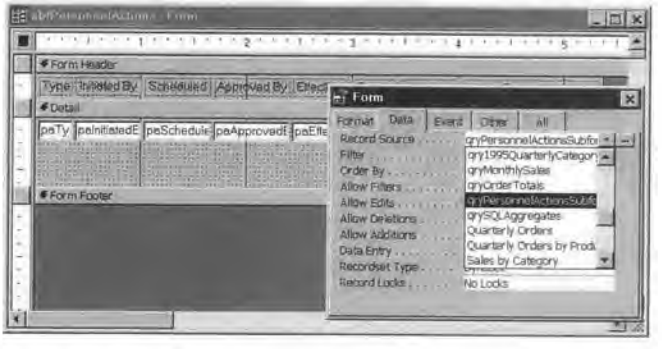


FIG. 12.35 Changing the Record Source property of the subform to change the underlying data source.

8. To verify that the sort stays active when you add new entries, add a new record with today's date, and then click the Prior button to save the new entry. Click Next to display the new entry so that you can be sure the sorting is functional.

You can use the same method described in the preceding steps to change the data source of a main form. For example, you could create a query for the main form so that the records in the form are displayed alphabetically by last name instead of in numerical order by EmployeeID.

Troubleshooting

I receive an error message in the Scheduled column of the subform datasheet every time I try to run the query.

You didn't click the Show box in step 6 of the set of instructions given earlier for creating the new query. When a field is duplicated in a query, Access doesn't know which of the two fields to use if the Show check box is marked for both fields.

Oracle Ex. 1009, p. 461
Oracle v. DeBellis
IPR2024-00221

Note

Pressing Shift+Enter (or choosing Records, Save Record) to save a record does not update the subform, and the new entry is not displayed in the subform. You must choose Records, Refresh to update the subform's display. The Refresh command doesn't requery the database but does cause any updates that have occurred since the last automatic refreshing operation to appear. The best approach, therefore, is to use the record position buttons to commit the new or edited record. The Refresh command is discussed in Chapter 25, "Securing Multiuser Network Applications."

Oracle Ex. 1009, p. 462
Oracle v. DeBellis
IPR2024-00221

Chapter 13

Designing Custom Multitable Forms

The controls that the Form Wizard adds to the forms it creates are only a sampling of the 20 native control objects offered by Access 97. Until now, you used the Form Wizard to create the labels, text boxes, and subform controls for displaying and editing data in the Personnel Actions table. These three kinds of controls are sufficient for creating a conventional transaction processing form.

The remaining 17 controls described in this chapter allow you to take full advantage of the Windows graphical user environment. You add controls to the form by using the Access Toolbox. List and combo boxes increase data entry productivity and accuracy by allowing you to choose from a list of predefined values rather than requiring you to type the value. Option buttons, toggle buttons, and check boxes supply values to Yes/No fields. If you place option buttons, toggle buttons, and check boxes in an option frame, these controls can supply the numeric values you specify. The Image control (which was a new feature of Access 95) supplements the Bound and Unbound Object Frame controls for adding pictures to your forms. Page breaks control how forms print. Access 97's new Tab control allows you to create tabbed forms to display related data on forms and subforms in a space-saving and more clearly organized fashion. Command buttons allow you to execute Access VBA procedures. ActiveX controls, which greatly expand the versatility of Access forms, are one of the subjects of Chapter 19, "Using 32-Bit OLE Components."

Understanding the Access Toolbox



The Access 97 Toolbox is based on the Toolbox created for Microsoft Visual Basic. Essentially, the Access Toolbox is a variation of toolbar. You choose one of the 20 buttons that appear in the Toolbox to add a control, represented by that tool's symbol, to the form. When you create a report, the Toolbox serves the same purpose—although tools that require user input, such as combo boxes, are seldom used in reports.

Oracle Ex. 1009, p. 463
Oracle v. DeBellis
IPR2024-00221

Control Categories

Three control object categories exist in Access forms and reports:

- *Bound controls* are associated with a field in the data source for the form or subform. The data source can be a table or query. Bound controls display and update values of the data cell in the associated field of the currently selected record. Text boxes are the most common bound control. You can display the content of graphic objects or play a waveform audio file with a bound OLE object. You can bind toggles, check boxes, and option buttons to Yes/No fields. All bound controls have associated labels that display the Caption property of the field; you can edit or delete these labels without affecting the bound control.
- *Unbound controls* display data you provide that is independent of the data source of the form or subform. You use the unbound OLE object to add a drawing or bitmapped image to a form. You can use lines and rectangles to divide a form into logical groups, or simulate boxes used on the paper form. Unbound text boxes are used to enter data that is not intended to update a field in the data source but is intended for other purposes, such as establishing a value used in an expression. Some unbound controls, such as unbound text boxes, include labels; others, such as unbound OLE objects, don't include labels.
- *Calculated controls* use expressions as their source of data. Usually, the expression includes the value of a field, but you also can use values created by unbound text boxes in calculated control expressions.

The Toolbox



You use the Access Toolbox to add control objects to forms and reports. The Toolbox appears only in Design mode for forms and reports, and it appears only if you click the Toolbox button on the toolbar or toggle the View, Toolbox menu command. When the Toolbox is visible, the Toolbox menu choice is checked; the Toolbox is shown in Figure 13.1. You can choose from the 20 controls and one wizard button, whose names and functions are listed in Table 13.1.



FIG. 13.1 The Access 97 Toolbox in its floating window.

Oracle Ex. 1009, p. 464
Oracle v. DeBellis
IPR2024-00221

Table 13.1 Control Objects of the Access Toolbox

Tool	Name	Function
	Select Objects	Changes mouse pointer to object selection tool. Deselects a previously selected tool and returns the mouse pointer to normal selection function. Select Objects is the default tool when you display the Toolbox.
	Control Wizards	Turns the Control Wizards on and off. Control Wizards aid you in designing complex controls, such as option groups, list boxes, and combo boxes.
	Label	Creates a box that contains fixed descriptive or instructional text.
	Text Box	Creates a box to display and allow editing of text data.
	Option Group	Creates a frame of adjustable size in which you can place toggle buttons, option buttons, or check boxes. Only one of the objects within an object group frame may be selected. When you select an object within an option group, the previously selected object is deselected.
	Toggle Button	Creates a button that changes from On to Off when clicked. The On state corresponds to Yes (–1), and the Off state corresponds to No (0). When used within an option group, toggling one button On toggles a previously selected button Off. You can use toggle buttons to let the user select one value from a set of values.
	Option Button	Creates a round button (originally called a *radio button*) that behaves identically to a toggle button. Option buttons are most commonly used within option groups to select between values in a set where the choices are mutually exclusive.
	Check Box	Creates a check box that toggles On and Off. Multiple check boxes should be used outside of option groups so that you can select more than one check box at a time.
	Combo Box	Creates a combo box with an editable text box where you can enter a value, as well as a list, from which you can select a value from a set of choices.
	List Box	Creates a drop-down list box from which you can select a value. A list box is simply the list portion of a combo box.
	Command Button	Creates a command button that, when clicked, triggers an event that can execute an Access VBA event-handling procedure.
	Image	Displays a static graphic on a form or report. This is not an OLE-type picture, so you can't edit it after placing it on the form.

(continues)

Oracle Ex. 1009, p. 465
Oracle v. DeBellis
IPR2024-00221

Table 13.1 Continued

Tool	Name	Function
	Unbound Object	Adds an OLE object created by an OLE server application, such as Microsoft Graph or Microsoft Draw, to a form or report. The ActiveX Control object is a special version of the Unbound Object that contains ActiveX Controls.
	Bound Object	Displays the content of an OLE field of a record if the field contains a graphic object. If the field contains no graphic object, the icon that represents the object appears, such as the Sound Recorder's icon for a linked or embedded .wav file. Data-bound ActiveX Controls are stored in a special version of the Bound Object.
	Page Break	Causes the printer to start a new page at the location of the page break on the form or report. Page breaks don't appear in form or report Run mode.
	Tab Control	Inserts a tab control to create tabbed forms. (The tab control looks like the tabbed pages you've seen in the various Properties windows throughout this book.) Pages of a tab control can contain other bound or unbound controls, including subform/subreport controls.
	Subform	Adds a subform or subreport to a main form or report, respectively. The subform or subreport you intend to add must exist before you use this control.
	Line	Creates a straight line that you can size and relocate. The color and width of the line can be changed by using the Formatting toolbar buttons or the Properties window.
	Rectangle	Creates a rectangle that you can size and relocate. The border color, width, and fill color of the rectangle are determined by selections from the palette.
	More Controls	Clicking this tool opens a scrolling list of additional ActiveX controls that you can use in your forms and reports. The ActiveX controls available through the More Controls list aren't part of Access; ActiveX controls are supplied as .ocx code libraries with Office 97, Visual Basic, and various third-party tool libraries.

Using controls in the design of reports is discussed in the following two chapters, which are devoted entirely to the subject of Access reports. The use of bound and unbound OLE objects is described in Chapter 20, "Adding Graphics to Forms and Reports." Using command buttons to execute Access VBA code is covered in Part VII of this book, which deals with Access VBA programming. You learn how to use the remaining 16 controls on your forms in the following sections.

Access's Control Wizards, Builders, and Toolbars

Access provides a number of features to aid you in designing and using more complex forms. Three of these features—Control Wizards, Builders, and customizable toolbars—are described in the three sections that follow.

Oracle Ex. 1009, p. 466
Oracle v. DeBellis
IPR2024-00221

Access Control Wizards. Much of the success of Access is attributable to the Form Wizard, Report Wizard, and Chart Wizard that simplify the process of creating database objects. The first wizard appeared in Microsoft Publisher, and most of Microsoft's productivity applications now include a variety of wizards. Chapter 12, "Creating and Using Forms," introduced the Form Wizard; the Report Wizard is discussed in Chapter 14, "Printing Basic Reports and Mailing Labels"; and the Chart Wizard is described in Chapters 19, "Using 32-Bit OLE Components," and 20, "Adding Graphics to Forms and Reports." Developers can create custom wizards to perform a variety of duties. You can expect a wide range of wizards to become available from independent software vendors (ISVs) as Access continues to gain adherents.

Access's repertoire of wizards includes Control Wizards that lead you step-by-step through the design of more complex control objects, such as option groups, list boxes, and combo boxes. Designing and populating list and combo boxes requires several steps. In this chapter, you are introduced to a Control Wizard each time you add a control for which a wizard is available.

Access Builders. Builders are another feature that makes Access easy to use. You use the Expression Builder, introduced in Chapter 4, "Working with Access Databases and Tables," to create expressions that supply values to calculated controls on a form or report. The Query Builder creates the SQL statements you need when you create list boxes or combo boxes whose Row Source property is a SQL statement that executes a select query. Using the Query Builder to insert SQL statements created by graphical QBE is much simpler than the method originally used with Access 1.x: You had to create a query, open the SQL window, copy the SQL statement to the Clipboard, and then paste the SQL statement to the Row Source property's text box. Using the Query Builder is described in the section "Using the Query Builder to Populate a Combo Box" near the end of this chapter.



Customizable Toolbars. The preceding chapters demonstrated that the toolbars of Access include many shortcut buttons to expedite the design and use of Access database objects. Access 97, like most other contemporary Microsoft applications, lets you customize the toolbars to your own set of preferences. Access 1.x stored toolbars as forms in UTILITY.MDA. Access 2.0 used the 16-bit common toolbar Dynamic Link Library, COMMTB.DLL, to manipulate toolbars. Access 95 used the 32-bit version of the common toolbar dynamic link library, Commtb32.dll, and (like Access 2.0) stored definitions of your customized versions of standard toolbars in the MSysToolbars table of System.mdw. (The significance of storing your preferences in System.mdw is discussed in Chapter 25, "Securing Multiuser Network Applications.") Access 97 also uses the Commtb32.dll to manipulate toolbars, and it stores customized toolbars in System.mdw. Toolbars that you create yourself are stored in a hidden system table—MSysCmdbars—in each database.

You can convert conventional floating design tools, such as the Toolbox, to conventional toolbars by the drag-and-drop method. To anchor the Toolbox as a toolbar, also called *docking the toolbar*, follow these steps:

Oracle Ex. 1009, p. 467
Oracle v. DeBellis
IPR2024-00221

1. Press and hold down the mouse button while the mouse pointer is on the title bar of the Toolbox, and then drag the Toolbox toward the top of Access's parent window.

   When the Toolbox reaches the toolbar area, the dotted outline changes from a rectangle that is approximately the size of the Toolbox into a wider rectangle with only as much height as a toolbar.

2. Release the mouse button to change the Toolbox to an anchored toolbar positioned below the standard Form Design toolbar.

Note

You can anchor or dock a toolbar to any edge of Access's parent window. Press and hold down the mouse button on an empty area of the toolbar (not covered by a button), and then drag the toolbar until its outline appears along the left, right, or bottom edge of the window. If you drop the toolbar within the confines of Access's main window, it becomes a floating toolbar.

You can add or delete buttons from toolbars with the Customize Toolbars dialog. If you are using the conventional VGA display format (640×480 pixels), there is very little room to add new buttons to the Form Design toolbar, and no room exists for adding buttons to the Formatting toolbar. However, the Toolbox toolbar has room to add seven or eight additional buttons when Access's main window is maximized. To add form design utility buttons to the Toolbox toolbar (whether it's docked or floating), do the following:



1. Choose View, Toolbars, Customize to display the Customize dialog.
2. Click the Toolbars tab to display the list of built-in and custom toolbars, and then select Toolbox in the list of toolbars to both display the Toolbox (if it isn't already visible) and select the Toolbox toolbar for customization.

Tip

You can also open the Customize dialog by right-clicking any part of a toolbar and then choosing Customize on the resulting popup menu.

3. Click the Commands tab, and then select Form/Report Design from the Categories list. The optional buttons applicable to form design operations appear in the Commands list, as shown in Figure 13.2.
4. The most useful optional buttons for form design are control alignment and sizing buttons. Press and hold down the mouse button on the Align to Grid command, and then drag this button to the Toolbox toolbar and drop it to the right of the Rectangle button. The right margin of the Toolbox toolbar expands to accommodate the new button (if you customize the Toolbox while it's floating, the window expands to accommodate the new button). You can drag the Snap to Grid button slightly to the right to create a gap between the new button and the Rectangle button.

Oracle Ex. 1009, p. 468
Oracle v. DeBellis
IPR2024-00221

5. Repeat step 4 for the Size to Fit, Size to Grid, and Align Left commands, dropping each button to the right of the preceding button. You now have four new buttons available in your Toolbox.

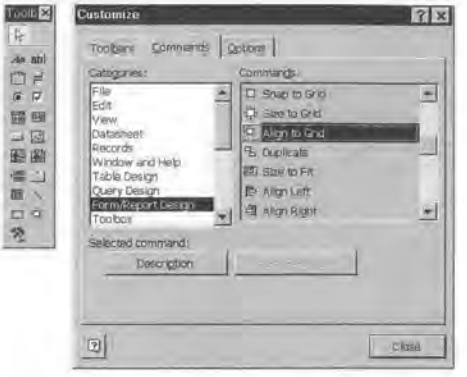


FIG. 13.2 The Toolbox with the Align to Grid command added and the open Customize dialog used to add commands.

The Customize dialog for toolbars provides the following additional capabilities:

- To remove buttons from the toolbar, open the Customize dialog; click and drag the buttons you don't want, and drop them anywhere off the toolbar.
- To reset the toolbar to its default design, open the Customize dialog. In the Toolbars list, select the toolbar you want to reset, then click the Reset button. A message box asks you to confirm that you want to abandon any changes you made to the toolbar.
- To create a button that opens or runs a database object, open the Customize dialog, display the Commands page, and scroll the Categories list to display the All *Objects* items. When, for example, you select All Tables, the tables of the current database appear in the Commands list. Select a table name, such as Employees, and drag the selected item to an empty spot on a toolbar. The ScreenTip for the new button displays `Open Table 'Employees'`. (If you select All Macros and drag a macro object to the toolbar, the button you add runs the macro when clicked.)
- To substitute text or a different image for the picture on the buttons you add to a toolbar, open the Customize dialog. Then click the button you want to change with the right mouse button to display the button shortcut menu. Click Choose Button Face to display the Choose Button Face dialog. Click one of the images offered, or click the Text check box and type the text you want to display in the text box. To edit the image of the button, click Edit Button Image.
- To create a new empty toolbar that you can customize with any set of the supplied buttons you want, open the Customize dialog and select Utility 1 or Utility 2 on the Toolbars page. If space exists to the right of an existing toolbar, the empty toolbar appears in this space. Otherwise, Access creates a new toolbar row for the

Oracle Ex. 1009, p. 469
Oracle v. DeBellis
IPR2024-00221

empty toolbar. The Utility 1 and Utility 2 toolbars are available in any Access database; changes you make to these toolbars will be universally available.

- To create a custom toolbar that becomes part of your database, open the Customize dialog and click New on the Toolbars page of the dialog. The New Toolbar dialog appears, requesting a name for the new toolbar (Custom 1 is the default). Access creates a new floating tool window to which you add buttons from the Commands page of the Customize dialog. You can anchor the custom tool window to the toolbar if you want.
- To delete a custom toolbar, open the Customize dialog, select the custom toolbar on the Toolbars page, and click the Delete button. You are requested to confirm the deletion. The Delete button is disabled when you select one of Access's standard toolbars in the list.

Custom toolbars to which you assign names become part of your database application and are stored in the current database file; they are available only when the database in which they are stored is open. Built-in Access toolbars that you customize are stored in System.mdw and are available in any Access work session.

The Appearance of Controls in Design and Run Modes

The appearance in Form Design mode of the 17 different controls you can create with the Toolbox is shown in Figure 13.3. Labels were added to the page break, image, unbound object, line, and rectangle controls to identify them in the illustration. The labels aren't actually components of the controls.

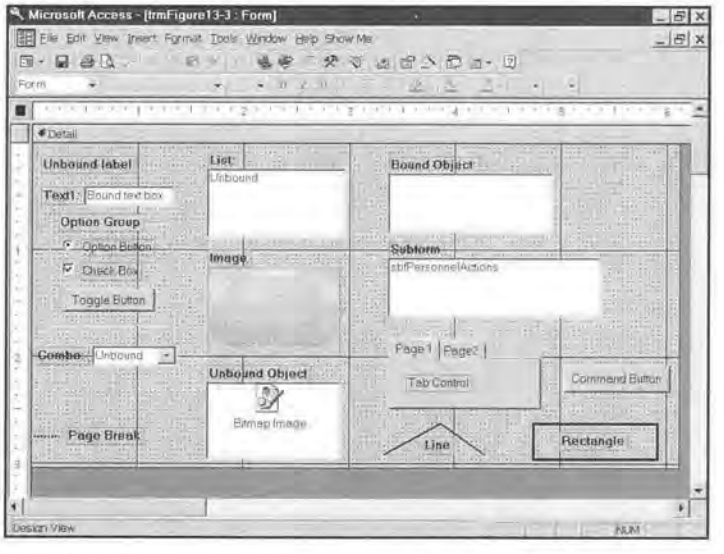


FIG. 13.3 The 17 controls you can create by using the Toolbox, shown in Design mode (with four custom buttons added to the Toolbox toolbar).

Oracle Ex. 1009, p. 470
Oracle v. DeBellis
IPR2024-00221

When you click the Form View button, the controls appear as shown in Figure 13.4. The text box displays a #Name? error message because no value is assigned to the content of the text box. The list and combo boxes show some test values, which are provided only to show how the control appears in Form view. The image field displays a bitmap copied from the Globe.wmf graphic with the Size Mode property set to Zoom; the contents of the image field aren't editable. The unbound object field contains the Stone.bmp graphics file. This file is an embedded OLE object displayed as an icon. The bound object field is empty because no value for its content is assigned to it. The subform field displays a portion of the sbfPersonnelActions subform (the field is too small to display a significant portion of the subform). You can't create a subform control unless you enter the name of an existing form as the value. No fields have been placed on the tab control, which shows the default number of tab pages (2). An unbound label was added to the first page to help identify the control.

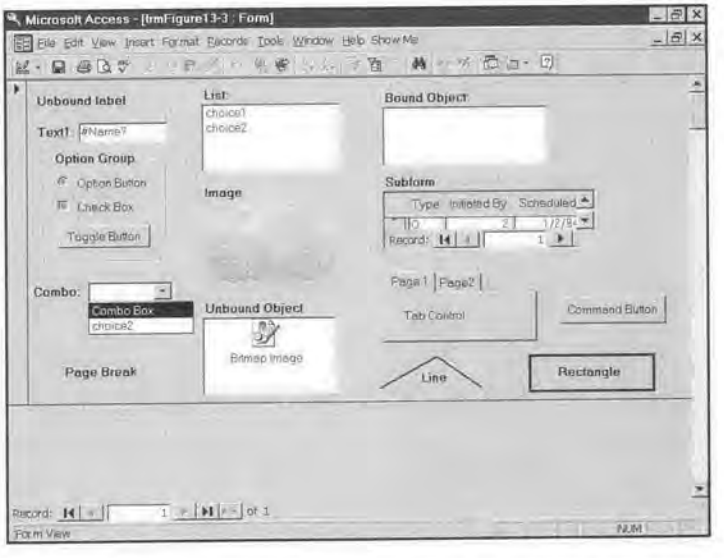


FIG. 13.4 The 17 controls you can create by using the Toolbox, shown in Form view. Notice that the list in the combo box has been dropped down.

Using the Toolbox to Add Controls

Experimenting is the best way of learning how to use a new computer application. No matter how well the product's documentation—or a book such as this—describes a process, no substitute exists for trying the methods. This axiom holds true whether you are designing a form or writing program code. The Microsoft programmers who created Access cleverly designed the user interface for creating custom forms so that the interface is intuitive and flexible. After you complete the examples in this chapter, you probably will agree that this statement is true.

Oracle Ex. 1009, p. 471
Oracle v. DeBellis
IPR2024-00221

The examples in this chapter use the Personnel Actions table that you created in Chapter 4, "Working with Access Databases and Tables," and two queries: qryPersonnelActions (which you create in the next section of this chapter, "Creating the Query on Which to Base the Main Form") and qryPersonnelActionsSubform (which you created in Chapter 12, "Creating and Using Forms"). The data dictionary needed to create the Personnel Actions table appears in Appendix C.



◀◀ See "Creating the Personnel Actions Table," p. 141
◀◀ See "Changing the Source of Data for the Subform and Sorting the Subform Data," p. 439

Creating the Query on Which to Base the Main Form

The Personnel Actions table identifies employees only by their ID numbers, located in the paID field. As before, you need to display the employee's name and title on the form to avoid entering records for the wrong person. To obtain the employee's name and title data for the form, you need to create a one-to-many query that joins the Employees table, which has only one entry per employee, with the Personnel Actions table, which can have many entries for one employee.

> **Note**
>
> In the form you created in Chapter 12, you used the Form Wizard to specify the tables from which you wanted to display data on the form. When you create a multitable form from scratch, you need to create a query as a data source for the form. The query joins the data from the tables into a unified source for use by the form.

To create the Personnel Actions query that serves as the data source for your main form, follow these steps:



1. Close any open forms, click the arrow to the right of the New Object button on the toolbar to open the drop-down menu, and click Query. Next, select Design View in the list of the New Query dialog and click OK. Double-click the Personnel Actions table in the list of tables of the Show Table dialog to add the Personnel Actions table to your query.

   Alternatively, you can bypass the Show Table dialog step by clicking the Table button of the Database window and then selecting Personnel Actions in the table list. When you click the New Query button with a table selected, the table is added automatically to the new query.

2. Double-click the Employees table in the table list, and then click Close. The field list windows for the Personnel Actions and Employees tables appear in the upper pane of the Query Design window.



   If you used the alternative method to add the Personnel Actions table to the query described in step 1, you need to click the Show Table button on the toolbar to open the Show Table dialog to add the Employees table to your query.

Oracle Ex. 1009, p. 472
Oracle v. DeBellis
IPR2024-00221

3. Choose View, Table Names to add the Tables row to your Query Design grid. (If the Table Names command already has a check mark next to it, you don't need to choose it.)

4. If you defined relationships for the Personnel Action table as described in Chapter 4, "Working with Access Databases and Tables," the upper pane of the query window appears as shown in Figure 13.5. The line connecting the two tables indicates that a many-to-one relationship exists between the paID field in the Personnel Action table and the EmployeeID field of the Employees table.



FIG. 13.5 The upper pane of the Query Design window for the Personnel Actions query.

If you didn't define any relationships, the join line won't appear. In this case, you need to drag the paID field from the Personnel Actions field list to the EmployeeID field of the Employees field list to create a join between these two fields.

5. Click the * field of the Personnel Actions table, drag it to the first column of the Query Design grid, and drop it in the Personnel Actions column. This adds all of the fields to the Personnel Actions table to your query.

6. Click the LastName field of the Employees table, drag it to the Query grid, and drop it in the second column.

7. From the Employees table, click and drag the FirstName, Title, HireDate, Extension, Notes, ReportsTo, and Photo fields. Drop them in columns 3, 4, 5, 6, 7, 8, and 9 of the Query grid, respectively, as shown in Figure 13.6.

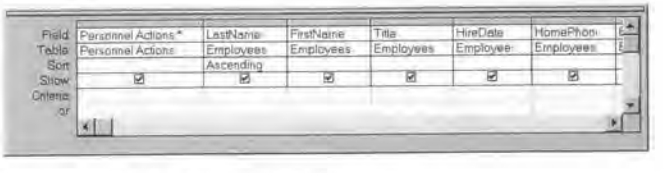


FIG. 13.6 The Query grid for the Personnel Actions query.

8. To simplify finding an employee, click the Sort row of the LastName column and select an Ascending sort.

9. Close the new query. Click Yes when the message box asks if you want to save the query.

10. In the Save As dialog, name the query **qryPersonnelActions** and click OK (see Figure 13.7).

Oracle Ex. 1009, p. 473
Oracle v. DeBellis
IPR2024-00221



FIG. 13.7 The Save As dialog for the Personnel Actions query.

Now that you've created the query that will provide a unified record source for the main form, you're ready to begin creating your custom multitable form.

Creating a Blank Form with a Header and Footer

When you create a form without using the Form Wizard, Access provides a default blank form to which you add controls that you choose from the Toolbox. To create a blank form with which to experiment with Access controls, perform the following steps:



1. With the Database window active, click the Forms tab in the Database window, and then click the New button. Access displays the New Form dialog.



2. Even an experimental form requires a data source, so choose qryPersonnelActions from the drop-down list, choose Design View from the list in the upper-right corner of the New Form dialog, and click OK.

3. Access creates a new blank form with the default title Form1. Click the Maximize button to expand the form to fill the document window.



4. If the Toolbox isn't visible, click the Toolbox button on the toolbar (or choose View, Toolbox) to display the Toolbox. Drag the Toolbox to the top or bottom of the form to anchor it there as a toolbar (do this if the Toolbox isn't already in a docked position—docking the toolbar usually makes working on a form easier because the floating Toolbox window obscures the form underneath it).

5. Choose View, Form Header/Footer. The blank form appears as shown in Figure 13.8. If the grid doesn't appear on the form, choose View, Grid.

   The default width of blank forms is 5 inches. The default height of the Form Header and Footer sections is 0.25 inch, and the height of the Detail section is 2 inches.

6. To adjust the height of the Detail section of the form, place the mouse pointer on the top line of the Form Footer bar. The mouse pointer becomes a double-headed arrow with a line between the heads. Hold down the left mouse button and drag the bar to create a height of about 3.5 inches, measured by the left vertical ruler. The active surface of the form, which is gray with the default 24×24 grid dots, expands vertically as you move the Form Footer bar, as shown in Figure 13.9.

7. Minimize the Form Footer section by dragging the bottom margin of the form to the bottom of the Form Footer bar.

8. Drag the right margin of the form to 6 inches as measured by the horizontal ruler at the top of the form.

Oracle Ex. 1009, p. 474
Oracle v. DeBellis
IPR2024-00221

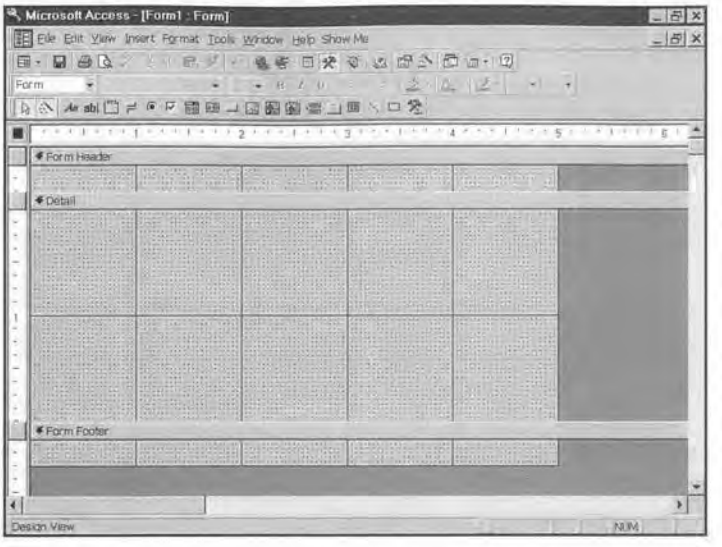


FIG. 13.8 Access's default blank form with Form Header and Form Footer sections added.

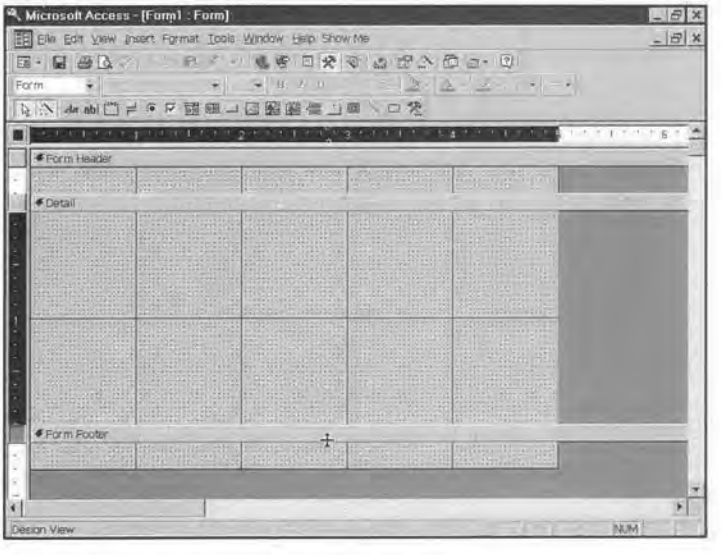


FIG. 13.9 Expanding the Detail section of the blank form.

You are using the blank form to create a form similar to the frmPersonnelActions form that you created in Chapter 12, "Creating and Using Forms."

Adding a Label to the Form Header

The label is the simplest control in the Toolbox to use. Labels are unbound and static, and they display only the text you enter. *Static* means that the label retains the value you

Oracle Ex. 1009, p. 475
Oracle v. DeBellis
IPR2024-00221

originally assigned as long as the form is displayed. To add a label to the Form Header section, complete the following steps:



1. Click the Label button in the Toolbox. When you move the mouse pointer to the active area of the form, the pointer becomes the symbol for the Label button, combined with a crosshair—the center point of the crosshair defines the position of the upper-left corner of the control.

2. Locate the crosshair at the upper-left of the Form Header section. Press and hold down the left mouse button while you drag the crosshair to the position for the lower-right corner of the label (see Figure 13.10).

   As you drag the crosshair, the outline of the container for the label follows your movement. The number of lines and characters that the text box can display in the currently selected font is shown in the status bar.

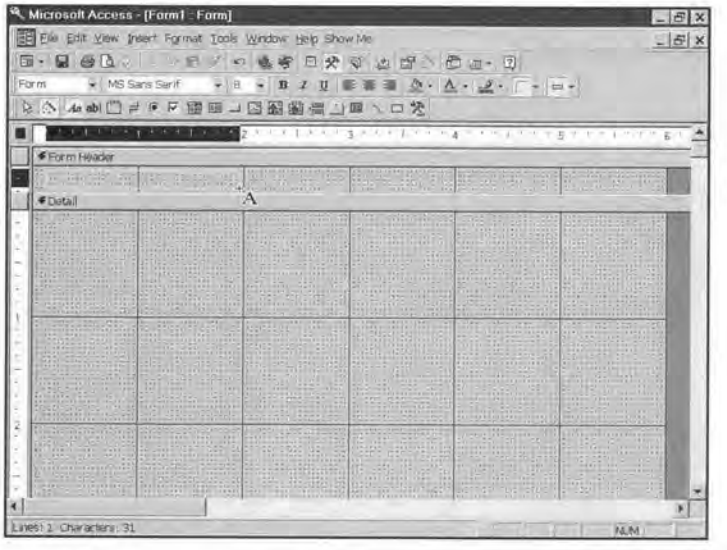


FIG. 13.10 Adding a label control to the Form Header.

3. If you move the crosshair beyond the bottom of the Form Header section, the Form Header bar expands to accommodate the size of the label after you release the left mouse button. When the label is the size you want, release the mouse button.

4. The mouse pointer becomes the text editing caret inside the outline of the label. Enter **Personnel Action Entry** as the text for the label, and then click anywhere outside the label to finish its creation. If you don't type at least one text character in a label after creating it, the box disappears the next time you click the mouse.



◀◀ See "Selecting, Moving, and Sizing a Single Control," p. 421

Oracle Ex. 1009, p. 476
Oracle v. DeBellis
IPR2024-00221

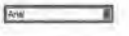
You use the basic process described in the preceding steps to add most of the other types of controls to a form. (Some Toolbox buttons, such as the graph and command buttons, launch a Control Wizard to help you create the control if the Control Wizards button is activated.) After you add the control, you use the anchor and sizing handles described in Chapter 12, "Creating and Using Forms," to move the control to the desired position and to size the control to accommodate the content. The location of the anchor handle determines the Left (horizontal) and Top (vertical) properties of the control. The sizing handles establish the control's Width and Height properties.



Formatting Text and Adjusting Text Control Sizes

When a control is selected that accepts text as the value, the typeface and font size combo boxes appear on the toolbar. To format the text that appears in a label or text box, complete the following steps:

1. Click the Personnel Action Entry label you created in the preceding section to select the label.
2. Double-click the label (or click the Properties button on the toolbar) to display the Properties window. (You don't actually need to use the Properties window to make these formatting changes. You open the Properties window to learn about how the changes you make to a control with toolbar buttons and menu commands affect the control's properties.)
3. Open the Font list on the Formatting toolbar and select the typeface family you want. MS Sans Serif, the default, is recommended because all users of Windows 95 and Windows NT 4.0 have this bitmapped font. (MS Sans Serif is quite similar to Linotype Company's Helvetica typeface or the Arial TrueType typeface also supplied with Windows.) Sans serif faces are easier to read on forms than faces with serifs, such as MS Serif or Times New Roman. (Serif faces are easier to read when a large amount of text is involved, such as in newspapers or the body text of this book.)



4. Open the Font Size list and select 14 points.



5. Click the Bold attribute button on the toolbar.



6. The size of the label you created isn't large enough to display the larger font. To adjust the size of the label to accommodate the content of the label, click the Size to Fit button—if you added it to the Toolbox—or choose Format, Size, To Fit. Access resizes the label's text box to display the entire label; if necessary, Access also increases the size of the Form Header section.

Note

The two sizing commands (Size, To Grid and Size, To Fit) work slightly differently, depending on whether one or more controls are selected. If one or more controls are selected when you execute

(continues)

Oracle Ex. 1009, p. 477
Oracle v. DeBellis
IPR2024-00221

(continued)

one of the sizing commands, the command is applied to the selected control(s). If no controls are selected, the chosen sizing command applies as the default to all objects you subsequently create, move, or resize.

When you change the properties of a control, the new values are reflected in the Properties window for the control, as shown in Figure 13.11. If you move or resize the label, you see the label's Left, Top, Width, and Height values change in the Properties window. You usually use the Properties window to actually change the characteristics of a control only if a toolbar button or menu choice isn't available.

You can choose different fonts and the bold, italic, and underline attributes (or a combination) for any label or caption for a control. You can assign the text content of list boxes and combo boxes to a typeface or size other than the default, but this practice is uncommon in Windows applications.

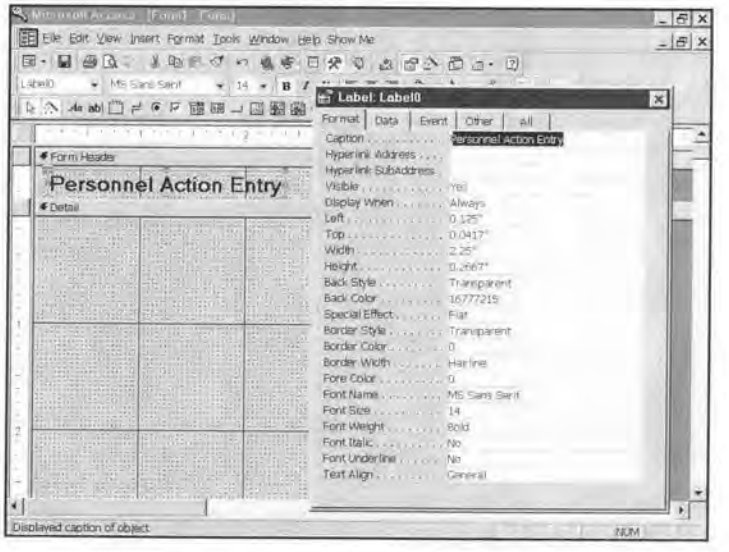


FIG. 13.11 The form title label and its Properties window.

Creating Bound, Multiline, and Calculated Text Boxes

Access uses the following four basic kinds of text boxes:

- *Single-line text boxes* usually are bound to controls on the form or to fields in a table or query.
- *Multiline text boxes* usually are bound to Memo field types and include a vertical scroll bar to allow access to text that doesn't fit within the dimensions of the box.

Oracle Ex. 1009, p. 478
Oracle v. DeBellis
IPR2024-00221

- *Calculated text boxes* obtain values from expressions that begin with = (equal sign) and are usually a single line. If you include a field value, such as `[paScheduledDate]`, in the expression for a calculated text box, the text box is bound to that field. Otherwise, calculated text boxes are unbound. You cannot edit the value of a calculated text box.
- *Unbound text boxes* can be used to supply values—such as limiting dates—to Access VBA procedures. An unbound text box that doesn't contain a calculation expression can be edited.

The following sections show you how to create the first three types of text boxes.

Adding a Text Box Bound to a Field. The most common text box used in Access forms is the single-line bound text box that comprises the majority of the controls of the frmPersonnelActions form you created in Chapter 12. To add a text box that is bound to a field of the form's data source with the field list window, complete the following steps:

1. Click the Field List button on the toolbar. The field list window appears.
2. Click the paID field in the field list window. Hold down the mouse button and drag the field to the upper-left corner of the form's Detail section. When you move the mouse pointer to the active area of the form, the pointer becomes a field symbol, but no crosshair appears. The position of the field symbol indicates the upper-left corner of the text box, not the label, so drop the symbol in the approximate position of the text box anchor handle, as shown in Figure 13.12.

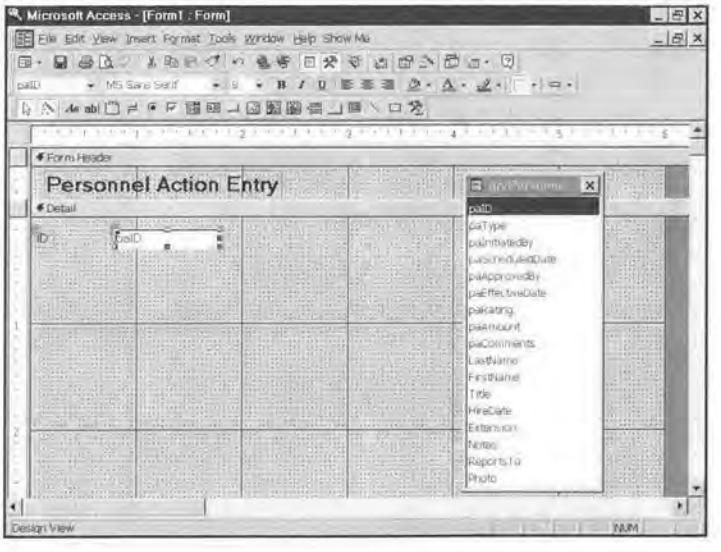


FIG. 13.12 Adding a text box control that is bound to the paID field.

Oracle Ex. 1009, p. 479
Oracle v. DeBellis
IPR2024-00221

3. Drag the text box by the anchor handle closer to the ID label, and decrease the box's width.



4. Small type sizes outside of a field text box are more readable when you turn the bold attribute on. Choose the ID label and click the Bold button.

5. Choose File, Save, and then type the name **frmPersonnelActionsEntry** in the Form Name text box of the Save As dialog. Click OK.

> **Note**
>
> When Access creates a text label that is associated with a form control, it uses the bound object's name as the value for the text label. If the form control is bound to a table object, such as a field, that has a Caption property (and the Caption property isn't blank), then Access uses the value of the Caption property as the default value for the text label of the bound form control. When you created the Personnel Actions table in Chapter 4, "Working with Access Databases and Tables," you set the Caption property for each field name. The paID field has a Caption property set to ID, so the label for the text box bound to the paID field is also ID.

Steps 3 and 4 in the preceding example are included to show how to make minor design adjustments to controls that improve the appearance of forms. Step 5 was included because you've already put some effort into this sample form (which you'll continue to experiment with and eventually complete in the following sections of this chapter). Furthermore, it's good working practice to save your documents frequently.

Adding a Multiline Text Box with Scroll Bars. Although you can use a conventional text box to display comments or other text fields with lengthy content, you must then scroll the caret through the text box to read the content. Multiline text boxes allow you to display long strings of text as a series of lines whose width is determined by the width of the multiline text box. To create a multiline text box, perform the following steps:

1. Click and drag paComments from the field list window to about the middle of the Detail section and drop it there.

2. Delete the Comments label, and size the text box as shown in Figure 13.13.



3. Click the Properties button on the toolbar, and click the Format tab in the Properties window. Scroll the Format Properties list for the text box until the Scroll Bars property appears.

4. Open the drop-down list for the Scroll Bars property, and choose Vertical to add a vertical scroll bar to the Comments text box.

5. If you plan to print the form, change the Can Grow and Can Shrink properties from No to Yes (for the height of the printed version of the form to vary with the number of lines of text in the box). The Can Grow and Can Shrink properties don't affect the appearance of the form in Run mode.

Oracle Ex. 1009, p. 480
Oracle v. DeBellis
IPR2024-00221

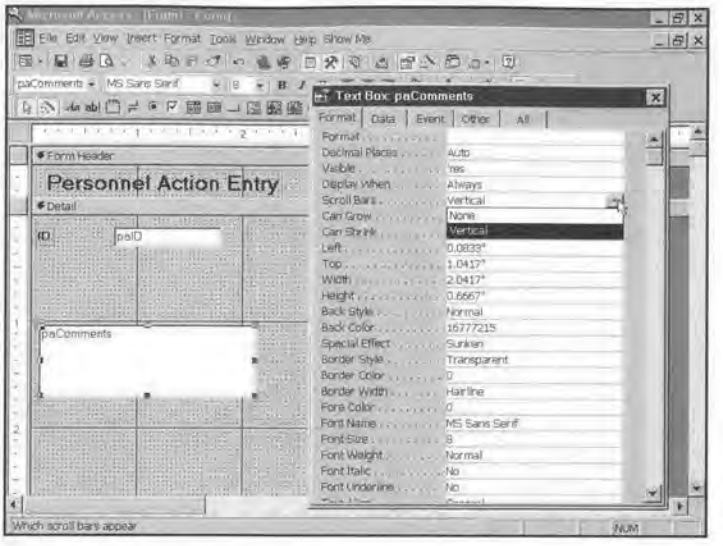


FIG. 13.13 Adding the multiline paComments text box.

> **Note**
>
> The vertical scroll bar of a multiline text box is visible only in the form's Run mode, and then only when the multiline text box has the focus (when the caret is within the text box).

Creating a Calculated Text Box. You can display the result of all valid Access expressions in a calculated text box. An expression must begin with = and may use Access functions to return values. As mentioned in the introduction to this section, you can use calculated text boxes to display calculations based on the values of fields. To create a calculated text box that displays the current date and time, complete the following steps:

1. Close the field list and Properties windows. Click the Text Box tool in the Toolbox to add an unbound text box at the right of the Form Header section of the form.
2. Edit the label of the new text box to read `Date/Time:`, and relocate the label so that it is adjacent to the text box.
3. Type **=Now()** in the text box to display the current date and time from your computer's clock. (In Design view, the form displays the calculation formula; it displays the actual date and time only in Form view.)
4. Adjust the length of the text box to accommodate the number of characters—20—in the default DD/MM/YY HH:MM:SS PM format used for dates and times. The entry appears as shown in the Date/Time text box of Figure 13.14. You add the other two text boxes in the following section.

Oracle Ex. 1009, p. 481
Oracle v. DeBellis
IPR2024-00221

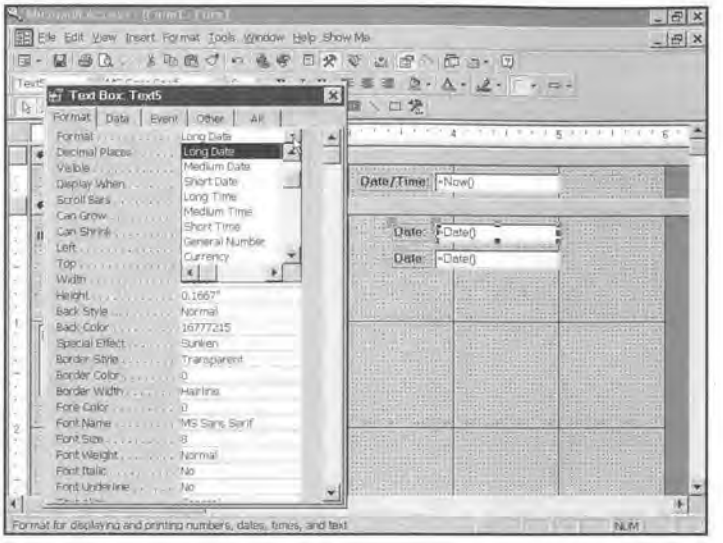


FIG. 13.14 Creating and formatting a calculated text box to display the date and time.

Formatting Values. You can use the Format property you learned about in Chapter 4 to determine how dates, times, and numbers are displayed in a text box on a form. To format a date entry, perform the following steps:

1. Using the Text Box tool, add a second unbound text box in the Detail section of the form under the first text box in the Form Header section. Adjust the new box's dimensions to correspond to the other text box.
2. Edit the label to read `Date:` and enter **=Date()** in the text box.
3. Select the text box, then display the Properties window in one of two ways: click the right mouse button to display a popup menu, and choose Properties; or click the Properties button on the toolbar. Click the Format tab in the Properties window.
4. Click the Format property and open the drop-down list. Select Long Date from the list.

A Format property applied to a bound text box on a form or report overrides the format assigned to the field in the table design that supplies the value to the text box.

> **Note**
>
> When you display a form in Run mode, the value displayed in the Date/Time text box is the time that you open the form. To update the time, choose Records, Refresh. The refreshing process that occurs at an interval determined by the Refresh Interval property of the Multiuser Options (the default value is 15 seconds) doesn't update unbound text boxes.

Oracle Ex. 1009, p. 482
Oracle v. DeBellis
IPR2024-00221

Using the Clipboard with Controls. You can use the Windows Clipboard to easily make copies of controls and their properties. As an example, create a copy of one of the Date/Time controls using the Clipboard by performing the following steps:

1. Select the unbound control and its label by clicking the field text box of the second date text box you added in the preceding section. Both the label and the text box are selected, as indicated by the selection handles on both controls.
2. To copy the selected control to the Clipboard, do one of the following: press Ctrl+C, click the Copy button on the toolbar, or choose Edit, Copy.



3. To paste the copy of the control below the original version, do one of the following: press Ctrl+V, click the Paste button on the toolbar, or choose Edit, Paste.



4. Click the Format property in the Properties window for the copied control, and select Short Date from the drop-down list.
5. To display the controls you've created, click the Form View button on the toolbar, and then return to Design view.



6. Delete the two Date text boxes and labels. To do so, enclose both with a selection boundary created by dragging the mouse pointer across the text boxes from the upper-left to the lower-right corner. Then press Delete. (You only need the Date/Time text box in the Form Header section for this form.)
7. Click the Form View button to view the form (see Figure 13.15).



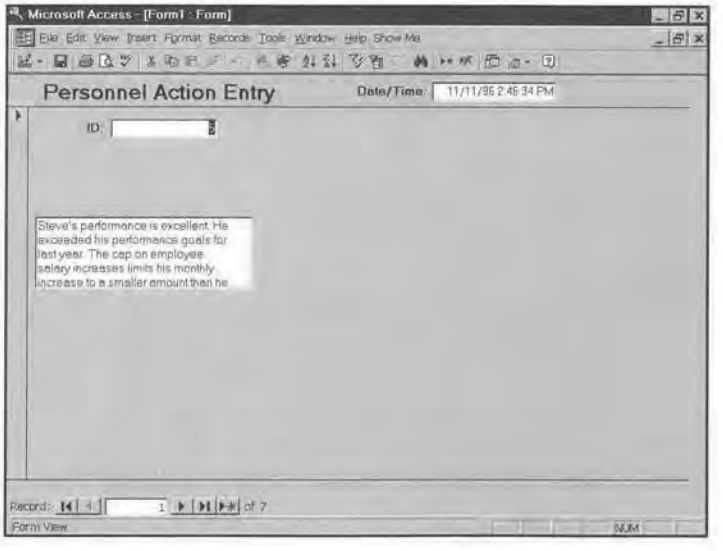


FIG. 13.15 The form title and text boxes displayed in Form view.

Text boxes (and their associated labels) are the most commonly used control objects on Access forms.

Oracle Ex. 1009, p. 483
Oracle v. DeBellis
IPR2024-00221

Changing the Default View and Obtaining Help for Properties

A form that fills Access's Design window might not necessarily fill the window in Run mode. Run mode may allow the beginning of a second copy of the form to appear. The second copy is created because the Default View property has a value of Continuous Forms. (In Access 2.0, the default property value for Default view was Continuous Forms; in Access 95 and 97, the default value of Default view is Single Form—your test form won't show the second form at the bottom of the screen.) Forms have the following three Default View property values from which you can choose:

- *Single Form* displays one record at a time in one form.
- *Continuous Forms* displays multiple records, each record having a copy of the Detail section of the form. You can use the vertical scroll bar or the record selection buttons to select which record to display. Continuous Forms view is the default value for subforms created by the Form Wizard.
- *Datasheet* displays the form fields arranged in rows and columns.

To change the Default View property of the form, complete the following steps:



1. Click the Design View button on the toolbar.
2. Choose Edit, Select Form.



3. Click the Properties button on the toolbar if the Properties window isn't visible. Click the Format tab in the Properties window.
4. Click the Default View property to open the list.
5. Select the value you want for this property for the current form. For this exercise, select Single Form (the default) from the list.
6. While Default view is selected, press F1. The Help window for the Default View property appears. This Help window also explains how the Default view and Views Allowed properties relate to each other.

   The vertical scroll bar disappears from the form in Run mode if a single form fits within its MDI child window.



You can verify your changes to the Default View property by clicking the Form View button to review the form's appearance.

Adding Option Groups, Binding Controls, and Locking Tools

Option buttons, toggle buttons, and check boxes can return only Yes/No (—1/0 or True/False) values when used by themselves on a form. Here, their use as bound controls is limited to providing values to Yes/No fields in a table. When you place any of these controls within an option group, the buttons or check boxes can return a number you specify for the Option Value property of the control.

The capability of assigning numbers to the Option Value property allows you to use one of the preceding three controls inside an option group frame for assigning values to the

Oracle Ex. 1009, p. 484
Oracle v. DeBellis
IPR2024-00221

paRating field of the Personnel Actions table. Option buttons are most commonly employed in Windows applications to select one value from a limited number of values.

Caution

Placing check boxes within option groups violates the Windows user interface design guidelines. According to the guidelines, a group of check boxes provides multiple additive choices. Thus, if you have more than one check box in a group, any or all of the check boxes can be marked. Use the shape control to create a frame around check boxes. Only option buttons and toggle buttons should be used in option groups.

By default, all controls you add with the Toolbox are unbound controls. You can bind a control to a field by choosing the control you want to use and then clicking the field name in the Field List window to which you want the control bound. Another way of binding a control is to create an unbound control with a tool and then type the name of a field in the Control Source property text box (reach the Control Source text box by clicking the Data tab in the Properties window for the control).

Access 97 offers two means of creating an option group: using the Option Group Wizard or manually adding option buttons or toggle buttons to the option group. The following two sections describe these methods.

Using the Option Group Wizard. The Option Group Wizard is one of three Control Wizards that take you step by step through the creation of complex controls. To create an option group for the paRating field of the Personnel Actions table with the Option Group Wizard, follow these steps:

1. Click the Control Wizards tool to turn on the wizards if the toggle button is not On (the default value). Toggle buttons indicate the On (True) state with a sunken appearance.



2. Click the Option Group tool, position the pointer where you want the upper-left corner of the option group, and then click the mouse button to display the first dialog of the Option Group Wizard, shown in Figure 13.16.





FIG. 13.16 The opening dialog of the Option Group Wizard.

Oracle Ex. 1009, p. 485
Oracle v. DeBellis
IPR2024-00221

3. Type five of the nine ratings in the Label Names datasheet: **Excellent**, **Good**, **Acceptable**, **Fair**, and **Poor** (see Figure 13.16). Click the Next button to display the second dialog of the Option Group Wizard, shown in Figure 13.17.

Tip

You can create accelerator keys in the captions of your option buttons by placing an ampersand (**&**) before the letter to be used as an accelerator key. Thereafter, pressing Alt in combination with that letter key selects the option when your form is in Run mode.



FIG. 13.17 Choosing a default value for the options group.

4. The second dialog lets you set an optional default value for the option group. Select the option named Yes, the Default Choice Is, and then open the drop-down list. Select Good, as shown in Figure 13.17, and then click Next. If you need to, you can always return to the prior step by clicking Back.

5. The third dialog of the Option Group Wizard provides for the assignment of option values to each option button of the group. Type **9**, **7**, **5**, **3**, and **1** in the five text boxes, as illustrated by Figure 13.18, and then click the Next button.



FIG. 13.18 Assigning the numeric OptionValue property to the option buttons.

Oracle Ex. 1009, p. 486
Oracle v. DeBellis
IPR2024-00221

The domain integrity rule for the paRating field provides for nine different ratings. Nine option buttons, however, occupy too much space on a form. Thus, only five of the nine ratings are provided here. In the section "Creating an Option Group Manually" later in this chapter, you add to this form a drop-down combo list with all nine ratings.

6. The fourth Option Group Wizard dialog enables you to bind the option frame to a field of a table or a column of a query that acts as the Record Source of the bound form. Select the paRating column of the qryPersonnelActions query to which your form is bound (see Figure 13.19). Click Next to continue with the next stage of the wizard.

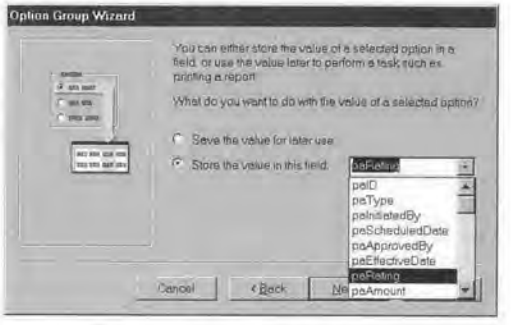


FIG. 13.19 Binding the option group to a column of the form's Record Source.

7. The fifth dialog lets you determine the style of the option group, as well as the type of controls (option buttons, check boxes, or toggle buttons) to add to the option group. You can preview the appearance of your option group and button style choices in the Sample pane. For this example, accept the defaults, Option Buttons and Etched (see Figure 13.20).



FIG. 13.20 Selecting a style for the option group and determining the type of button to add.

Oracle Ex. 1009, p. 487
Oracle v. DeBellis
IPR2024-00221

The sunken and raised styles of option groups, option buttons, and check boxes are applicable only to control objects on forms with a Back Color property other than white. Light gray is used to aid in the three-dimensional simulation, and neither option buttons nor check boxes have a Back Color property. Thus, option buttons and check boxes with special effects are best suited for light gray backgrounds (Back Color = 12632256).

8. The last dialog provides a text box for entering the value of the Caption property of the label for the option group. Type **Rating**, as shown in Figure 13.21, and then click Finish to let the wizard complete its work. Your completed Rating option group appears as shown in Figure 13.22.



FIG. 13.21 Assigning the value of the Caption property for the option group's label.

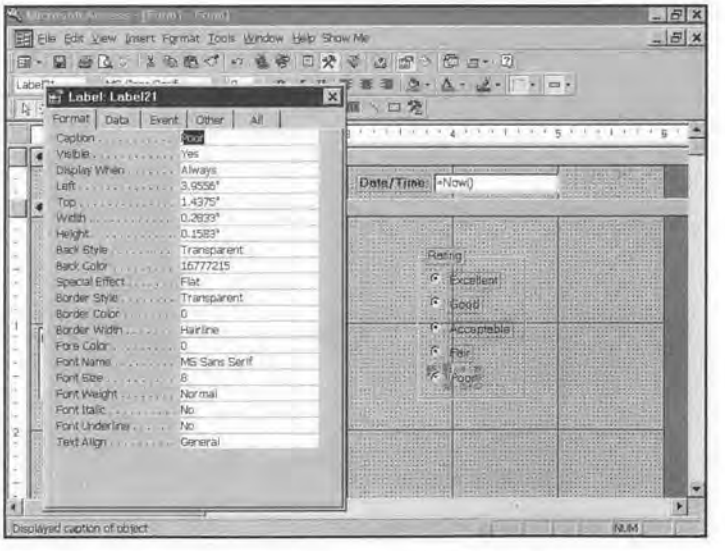


FIG. 13.22 The option group created by the Option Group Wizard.

Oracle Ex. 1009, p. 488
Oracle v. DeBellis
IPR2024-00221

To test your new bound option group, add a text box that is bound to the paRating column of the query underlying the form. Figure 13.23 shows the option group in Form view with the space between the buttons closed up, the bold attribute applied to the option group label, and the Rating text box added. Click the option buttons to display the rating value in the text box.

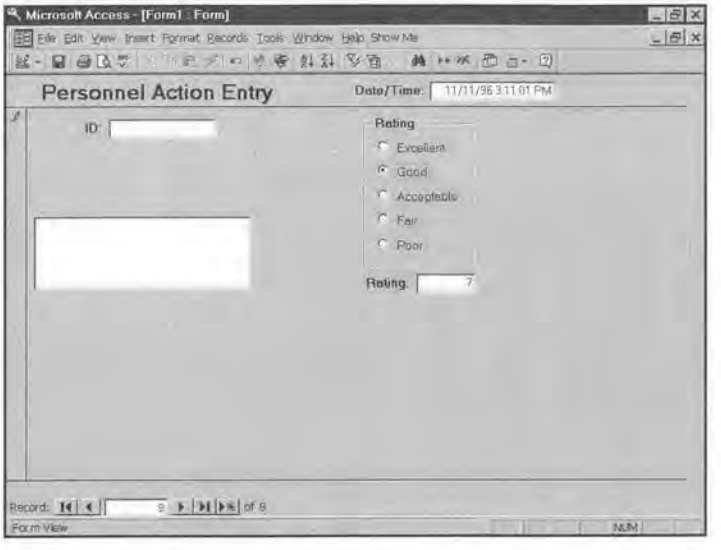


FIG. 13.23 The new option group in Form view with a text box for the Rating field added to show the effect of selecting different options.

Creating an Option Group Manually. Although the Option Group Wizard does a good job of creating option groups, it's useful to know how to create a bound option group on your own. To bind an option group frame to the paRating field of the Personnel Actions query without taking advantage of the Control Wizard, complete the following steps:

1. Deactivate the Control Wizard's toggle button in the Toolbox (make sure the button has a raised appearance), so you won't get any help from a wizard, and then click the Option Group tool in the Toolbox.

2. Click the Field List button on the toolbar to display the Field List window and choose the paRating column of your query.

3. Hold down the mouse button and drag the field pointer to a position to the right of the Rating option group you created in the preceding section. Then release the mouse button to create an option group of the default size.

   When you create a bound option group by dragging a field from the Field List window, the option group name is automatically assigned to the Caption property of the associated label.

Oracle Ex. 1009, p. 489
Oracle v. DeBellis
IPR2024-00221

4. Resize the option group frame so that it is the same size as the other Rating option group. Apply the bold attribute to the option group's label.

Option buttons, toggle buttons, and check boxes within bound frames inherit many of their properties, such as Control Source, from the frame. The option frame provides the binding of these tools when they are inside of a frame. Therefore, you don't use the field list with these controls. Adding multiple copies of a control is easier if you double-click the tool's button in the Toolbox to lock the tool on. (Click the tool at any time to unlock it.) To add five option buttons to assign values to the paRating field, perform the following steps:



1. Double-click the Option Button tool.
2. Using the crosshair as a reference to the upper-left corner of an imaginary rectangle that surrounds the option button, drop the option button at the appropriate location in the option group frame. When the option button symbol enters the option group frame, the button, frame, and contents appear in reverse video as shown in Figure 13.24.

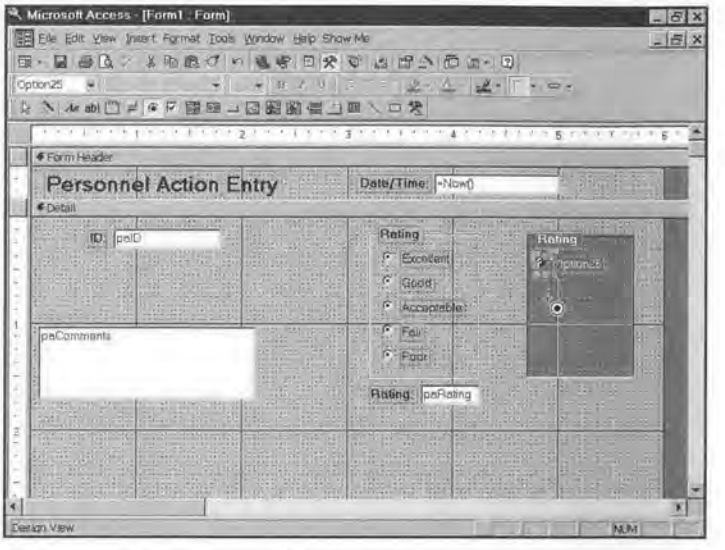


FIG. 13.24 Manually adding a second option button to an option frame.

3. Repeat step 2 four times to include a total of five Rating option buttons inside of the Rating option group frame. The labels of the buttons are assigned numbers in the sequence in which they were added.



4. Click the Option Button tool in the Toolbox again to unlock this tool; Access automatically changes back to the pointer tool.
5. Edit the labels to read the following from top to bottom: **Excellent**, **Good**, **Acceptable**, **Fair**, and **Poor**, corresponding to the option values of 9, 7, 5, 3, and 1, respectively.

Oracle Ex. 1009, p. 490
Oracle v. DeBellis
IPR2024-00221

6. Double-click the option button at the top to display its Properties window. Click the Data tab, and then replace 1 with **9** as the Option Value in the Data Properties list. A default Option Value is assigned in sequence from 1 to the number of buttons in the frame.

7. Repeat step 6 for the four remaining buttons, replacing the default values 2, 3, 4, and 5 with **7**, **5**, **3**, and **1**, respectively. No two buttons in an option frame can have the same value.

8. To test the entries, click the Form View button on the toolbar. The form appears as shown in Figure 13.25.

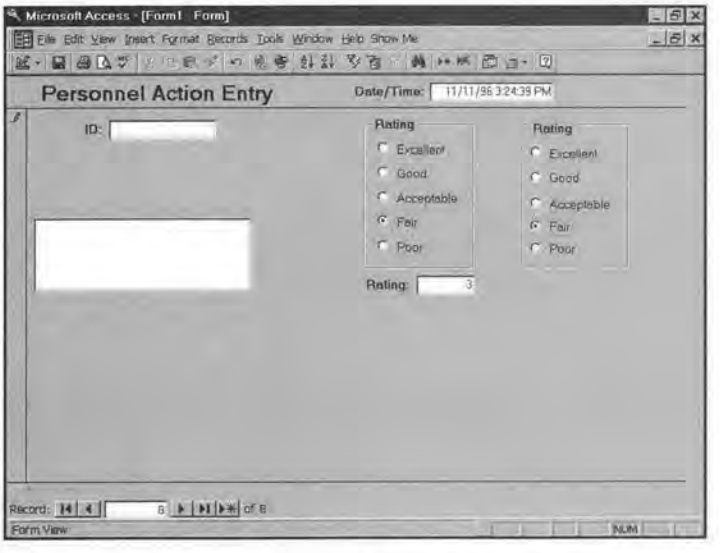


FIG. 13.25 The option group frame and option buttons displayed in Form view.

9. Using the record selection buttons, choose a record to edit. If you previously assigned ratings with odd-numbered values, the option button that corresponds to the value is selected.

10. Click the option buttons in sequence to verify that the proper numeric values appear in the Rating text box.

11. Click the Design View toolbar button to return to Design view.

12. You won't need either of the Rating option group boxes currently on the form, so delete both.

    The drop-down lists and combo boxes you learn about in the following sections of this chapter are a better control type to use than an option group for the relatively large number of choices available in the ratings field. Option groups are best for choosing one of only three or four choices.

Oracle Ex. 1009, p. 491
Oracle v. DeBellis
IPR2024-00221

If you have a Yes/No field in the table, you can use a single option button bound to a field (not inside an option frame) to create the Yes/No values for the user.

Note

If you add a button or a check box within a frame with the Field List drag-and-drop method, the button is independently bound to the selected field rather than to the field through the option frame. In this case, the button's Properties window doesn't include the Option Value property, and the button assigns Yes/No values to the field.

An independently bound button inside an option frame doesn't follow the rules of the option frame; you can choose this button and another button simultaneously.

Adding independently bound buttons within option frames results in the assignment of inconsistent values to fields.

Using the Clipboard to Copy Controls to Another Form

Access's capability of copying controls and their properties to the Windows Clipboard allows you to create controls on one form and copy them to another. If you use a standard header style, you can copy the controls in the header of a previously designed form to a new form and edit the content as necessary. The form that contains the controls to be copied need not be in the same database as the destination form in which the copy is pasted. You can create a library of standard controls in a dedicated form that is used only for holding standard controls.

The Time/Date calculated text box is a candidate to add to the frmPersonnelActions form you created in Chapter 12, "Creating and Using Forms." You may want to add a Time/Date text box to the Form Header or Detail section of all your transaction forms. To add the Time/Date control to the frmPersonnelActions form, perform the following steps:



1. Click the Design View button, and select the Time/Date control and its label by clicking the field text box.



2. To copy the selected control(s) to the Clipboard, do one of the following: press Ctrl+C; click the Copy button on the toolbar; or choose Edit, Copy.



3. Click the Show Database Window button on the toolbar. Then open the frmPersonnelActions form from the Database window in Design mode.



4. Click the Detail section selection bar, and then do one of the following: press Ctrl+V; click the Paste button on the toolbar; or choose Edit, Paste. A copy of the control appears at the upper-left corner of the Detail section.

   Controls are pasted to the section of the form that is presently selected. You cannot drag controls between sections of a form.



5. Position the mouse pointer over the copied option group so that the pointer becomes a hand symbol.

Oracle Ex. 1009, p. 492
Oracle v. DeBellis
IPR2024-00221

6. Hold down the mouse button and drag the option group to the position shown in Figure 13.26, and then release the mouse button.

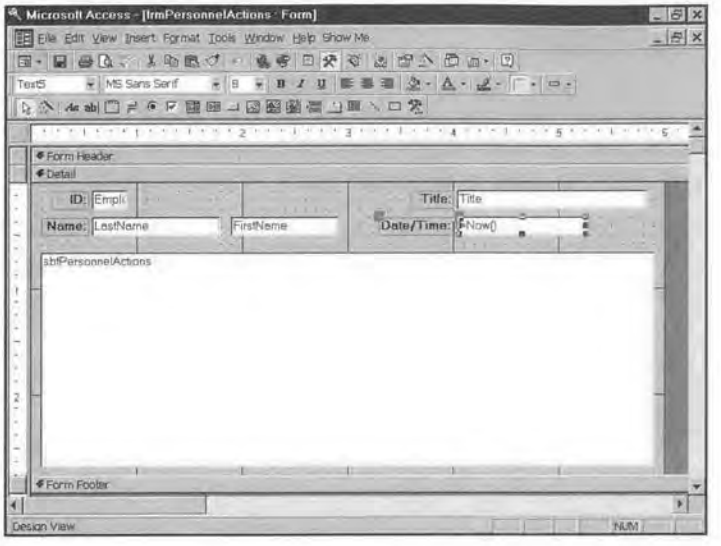


FIG. 13.26 Copying the Date/Time calculated field to the frmPersonnelActions form by using the Clipboard.

7. Click the Form View button on the toolbar. The Personnel Action Entry form appears as shown in Figure 13.27.



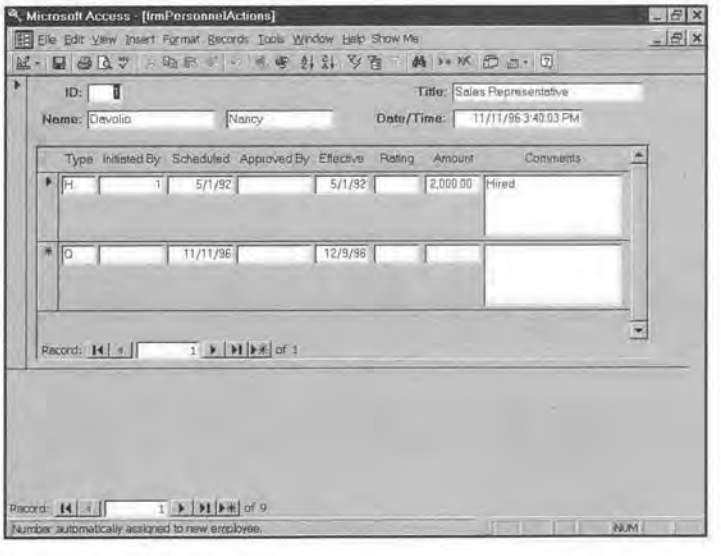


FIG. 13.27 The Date/Time text box displayed in Form view.

Oracle Ex. 1009, p. 493
Oracle v. DeBellis
IPR2024-00221



8. Return to Design mode, click the Save button to save your changes, and then click the Close window button to close the frmPersonnelActions form.



Troubleshooting

I've copied a control to another form, but when I attempt to use the form, I get an error message whenever that control gets the focus.

When you copy a control to a form that uses a data source that is different from the one used to create the original control, you need to change the Control Source property to correspond with the field to which the new control is to be bound. Changing the Control Source property doesn't change the Status Bar Text, Validation Rule, or Validation Text properties for the new control source. You must enter the appropriate values manually.

Using List Boxes and Combo Boxes

List boxes and combo boxes both serve the same basic purpose by enabling you to pick a value from a list, rather than type the value in a text box. These two kinds of list boxes are especially useful when you need to enter a code that represents the name of a person, firm, or product. You don't need to refer to a paper list of the codes and names to make the entry. The differences between list boxes and combo boxes are shown in the following list:

- *List boxes* don't need to be opened to display their content; the portion of the list that fits within the size of the list box you assign is visible at all times. Your choice is limited to values included in the list.
- *Drop-down combo boxes and drop-down lists* consume less space than list boxes in the form, but you must open these controls to select a value. Combo boxes in Access are drop-down lists plus a text box, not traditional combo boxes that display the list at all times. You can allow the user to enter a value in the text box element of the drop-down combo list or limit the selection to just the members in the drop-down list. If you limit the choice to members of the drop-down list (sometimes called a *pick list*), the user can still use the edit box to type the beginning of the list value—Access searches for a matching entry. This feature reduces the time needed to locate a choice in a long list.

Drop-down lists and combo boxes are two of the most powerful controls that Microsoft programmers developed for Access. The data source for these controls may be a table, a query, a list of values you supply, or the names of Access VBA functions. The boxes may have as many columns as you need to display the data for making the correct choice.

Adding a Combo Box with a Table or Query as the Data Source. In the majority of cases, you bind the drop-down list or combo box to a field so that the choice updates the value of this field. Two-column controls are most commonly used. The first column contains the code that updates the value of the field to which the control is bound, and the second column contains the name associated with the code. An example of where a limit-to-list, multiple-column, drop-down list is most useful is the assignment of supervisor and manager employee ID numbers to the paInitiatedBy and paApprovedBy fields

Oracle Ex. 1009, p. 494
Oracle v. DeBellis
IPR2024-00221

in the frmPersonnelActionsEntry form. The Combo Box Wizard is used to add the paInitiatedBy drop-down list, and then you employ manual methods to add the paApprovedBy drop-down list in the two sections that follow.

Using the Combo Box Wizard. Designing combo boxes is a more complex process than creating an option group, so you're likely to use the Combo Box Wizard for most of the combo boxes you add to forms. Follow these steps to use the Combo Box Wizard to create the paInitiatedBy drop-down list that lets you select from a list of Northwind Traders' employees:

1. Open the frmPersonnelActionsEntry form (that you created and saved earlier in this chapter) from the Database window in Design mode if it is not presently open.



2. Click the Control Wizards button, if necessary, so that the wizards are turned on.



3. Click the Combo Box tool in the Toolbox. The mouse pointer turns into a combo box symbol while on the active surface of the form.



4. Click the Field List button to display the Field List window.



5. Drag the paInitiatedBy field to a position at the top and extreme right-hand edge of the Detail section of the form, opposite the paID field (look ahead to Figure 13.34). The first Combo Box Wizard dialog appears.
6. You want the combo box to look up values in the Employees table, so accept the default option button and then click Next (see Figure 13.28). Your selection specifies Table/Query as the value of the Record Source property of the combo box. The second Combo Box Wizard dialog appears.

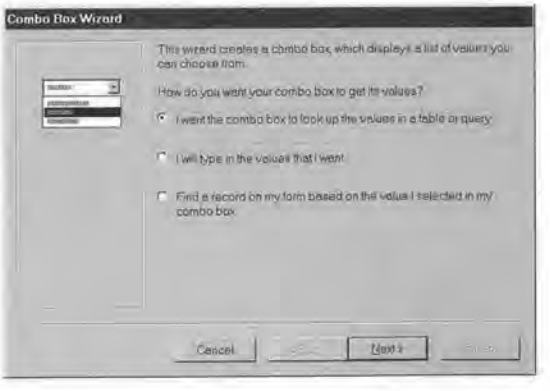


FIG. 13.28 Selecting the source of list values in the opening dialog of the Combo Box Wizard.

Oracle Ex. 1009, p. 495
Oracle v. DeBellis
IPR2024-00221

7. Select Employees from the list of tables in the list (see Figure 13.29). Click Next to reach the third dialog.



FIG. 13.29 Selecting the Record Source property of the combo box.

8. You need the EmployeeID and LastName fields of the Employees table for your combo box. EmployeeID serves as the bound field, and your combo box displays the LastName field. EmployeeID is selected in the Available Fields list by default, so click the > button to move EmployeeID to the Selected Fields list. LastName is selected automatically, so click the > button again to move LastName to the Selected Fields list. Your Combo Box Wizard dialog appears as shown in Figure 13.30. This selection specifies the SQL SELECT query that serves as the value of the Row Source property and populates the combo box's list. Click Next to reach the fourth dialog.

9. The fourth dialog (see Figure 13.31) displays the value list for the combo box. Access has successfully determined that the EmployeeID field is the key field of the Employees table and has assumed (correctly) that the EmployeeID field is the bound field for the combo box. The Hide key column check box is selected by default; this option causes Access to hide the bound column of the combo box. The result is that, although you've selected two columns for the combo box, only one column (the LastName field) is displayed in the combo box's list. The EmployeeID column is hidden and used only to supply the data value for the paInitiatedBy field. Resize the LastName column by dragging the right edge of the column leftward—you want the column wide enough to display everyone's last name but not any wider than absolutely necessary. Click Next to continue to the fifth Combo Box Wizard dialog.

10. Your combo box supplies the EmployeeID value corresponding to the name you select to the paInitiatedBy field. You previously specified that the Control Source property is paInitiatedBy when you dragged the field symbol to the form in step 5. The Combo Box Wizard uses your prior selection as the default value of the Control Source property (see Figure 13.32), so accept the default value by clicking the Next button to display the sixth and final dialog.

Oracle Ex. 1009, p. 496
Oracle v. DeBellis
IPR2024-00221

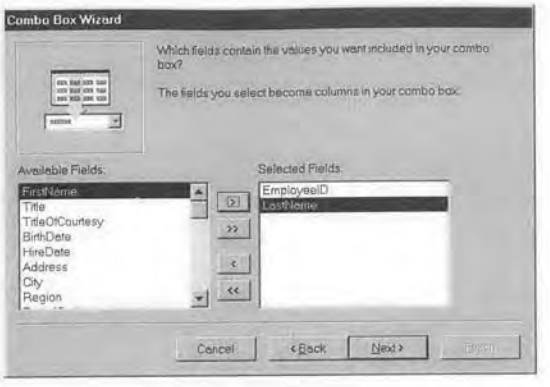


FIG. 13.30 Selecting the fields of the table with which to populate the combo box.

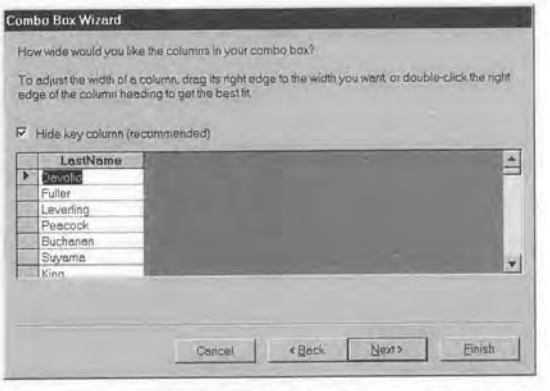


FIG. 13.31 Selecting the column width for the combo box and a hidden key field column.



FIG. 13.32 Assigning the Control Source property value.

Oracle Ex. 1009, p. 497
Oracle v. DeBellis
IPR2024-00221

11. The last dialog lets you edit the label associated with the combo box (see Figure 13.33). Type **Initiated By:** and click Finish to add the combo box to your form. Your combo box in Design mode appears as shown in Figure 13.34.



FIG. 13.33 The final Combo Box Wizard dialog allows you to edit the control's label.

12. Click the Form View button on the toolbar to test your combo box (see Figure 13.35). Change the Initiated By value to another person, such as Mr. Fuller, the Vice President of Sales, and then move the record pointer to make the change permanent. Return to the original record, and open the combo box to verify that the combo box is bound to the paInitiatedBy field.

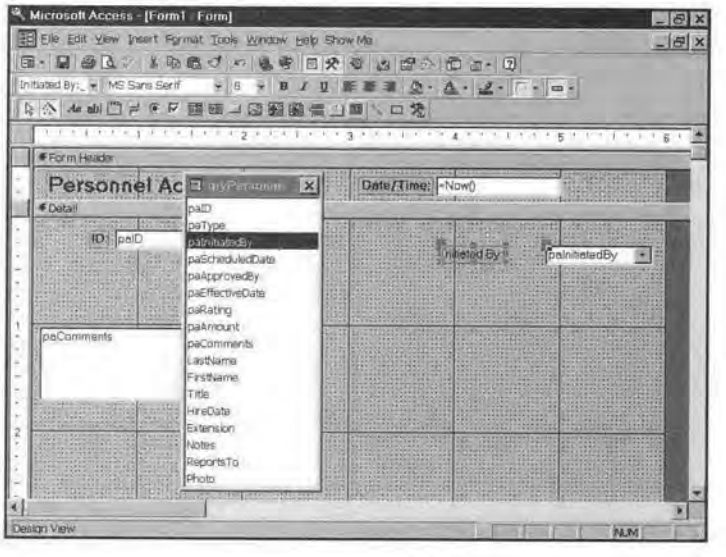


FIG. 13.34 The new combo box in Design mode.

Oracle Ex. 1009, p. 498
Oracle v. DeBellis
IPR2024-00221

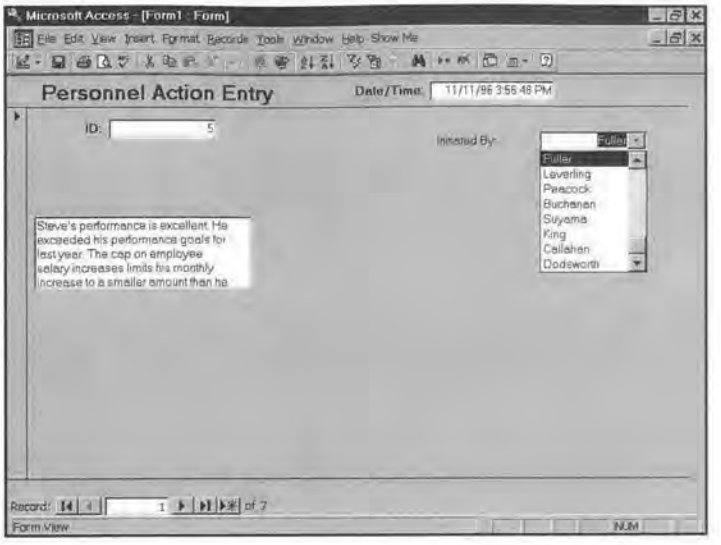
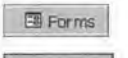


FIG. 13.35 The Initiated By combo box in Run mode.

Adding a Combo Box Manually. As mentioned previously in this chapter, it is good practice to create control objects manually so that you learn the properties associated with each control. To substitute a two-column combo box that you create yourself for the paApprovedBy text box in the frmPersonnelActionsEntry form, complete the following steps:

1. Open the frmPersonnelActionsEntry form from the Database window in Design mode if it is not presently open.

2. If necessary, click the Control Wizards button in the Toolbox to disable the wizards (make sure the button is raised).



3. Click the Field List button on the toolbar to open the Field List window if it isn't already open.



4. Click the Combo Box tool in the Toolbox. Then click paApprovedBy in the Field List window, and drag and drop the field symbol underneath the Initiated By combo box you created previously. Size the combo box and its label to match the Initiated By combo box above it.



5. Double-click the combo box to display the Properties window, and then click the Data tab.

6. The source of the data for the combo box is the Employees table, so the default value of the Row Source Type property, Table/Query, is correct. Place the caret in the Row Source property box, open the list, and select Employees as the value of the Row Source property.

Oracle Ex. 1009, p. 499
Oracle v. DeBellis
IPR2024-00221

7. When you choose the name of a table or query as the Row Source property, all fields of the table or columns of the query are included automatically as combo box columns. The first two columns of the Employees table provide EmployeeID to be assigned as the value of the paApprovedBy field and LastName to identify the supervisor. Click the Format tab in the Properties window, and type **2** as the Column count to create a two-column combo box.

8. The default width of each column of the combo box is 1 inch. The EmployeeID column can be less than 1 inch wide because it consists of only one digit. In the Column Width box, type **0.2** as the width of the first column followed by a semicolon (or comma) separator, and type **0.8** as the width of the second column. Access adds inch units (") for you.

9. Click the Data tab to return to the data properties. The first column of the Employees table—EmployeeID—contains the value to assign to the paApprovedBy field, so the default value of the Bound Column property—column 1—is correct. You can choose any column by its number (in left-to-right sequence) as the value to be assigned to the field to which the combo box is bound.

10. Only an employee included in the Employees table can initiate or approve a Personnel Action, so open the Limit to List drop-down list and select Yes. (If you want to allow the user to add a value not included in the list, accept the default No value. Adding a user-defined value is not applicable in this case.) The Personnel Actions form appears as shown in Figure 13.36.

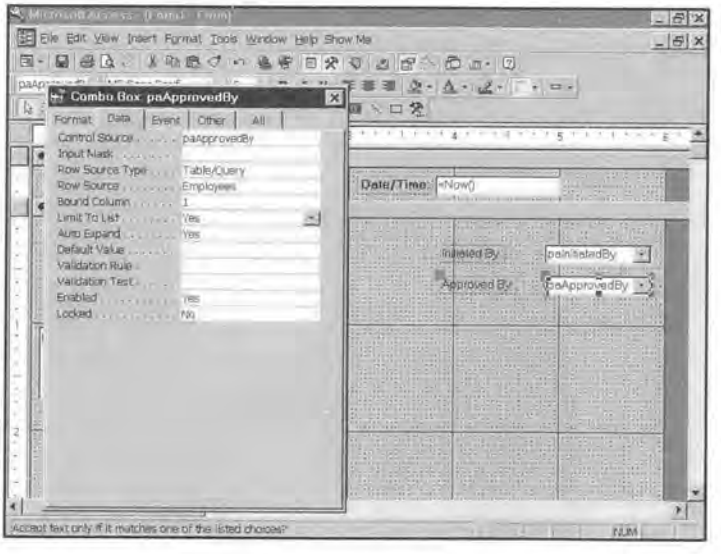


FIG. 13.36 Setting the values of the data properties of the combo box.



11. Click the Form View button on the toolbar to test the combo boxes. When you open the Approved By combo box, the display appears as shown in Figure 13.37.

Oracle Ex. 1009, p. 500
Oracle v. DeBellis
IPR2024-00221

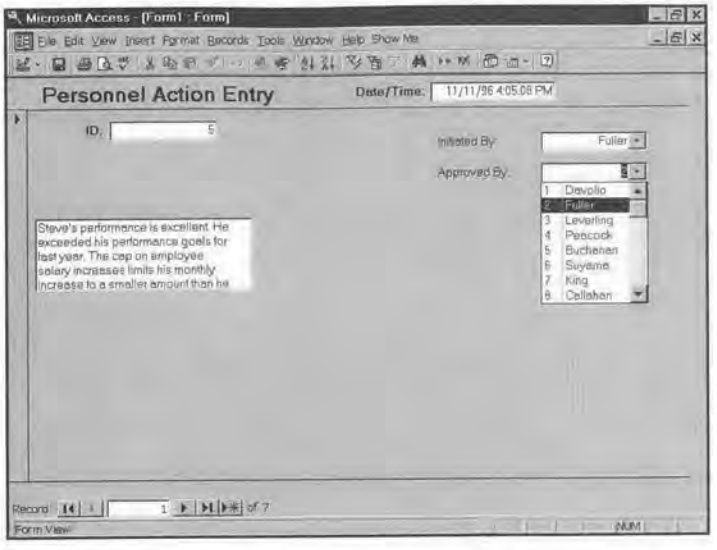


FIG. 13.37 The Approved By multiple-column combo box in Form view.

Notice that the EmployeeID field value appears in the text element of the combo box rather than the LastName field value, as in the Initiated By combo box. If the bound column appears in the list element, the value of the bound column appears in the text element.

12. To display only the name of the supervisor or manager in the list and text boxes, return to Design mode and change the value of the Column Widths property (on the Format tab of the Properties window) of the first column to 0 from 0.2 inches. This action causes only the second column to appear in the text box and list elements of the combo box, making the two combo boxes of your form consistent.

    As an example, if the fourth column of the table or query is the column you want to display in the combo box, type three zero-width columns preceding the width you want for the column that you want to display (0,0,0, and 1).

List and combo boxes are a boon to developers because Access does all of the work. Users who, in early versions of Clipper or later versions of dBASE, wrote the code necessary to create a popup window that contains a drop-down list, will appreciate the ease of creating a combo box in Access.

Using the Query Builder to Populate a Combo Box. If the Row Source Type property for a combo box is Table/Query, you can substitute a SQL statement for a named table or query as the value of the Row Source property. In the case of queries, the advantage of the substitution is that this process prevents the list of queries in the Database window from becoming cluttered with named queries used to create a multitude of combo boxes. For either tables or queries, you can choose only the fields or columns you want for the text box, eliminating the need to hide columns. In addition, you can specify a sort order for the list element of your combo box.

Oracle Ex. 1009, p. 501
Oracle v. DeBellis
IPR2024-00221

To invoke Access 97's Query Builder to create a SQL statement for populating the Approved By combo box, follow these steps:



1. Return to or open frmPersonnelActions in Design mode, and double-click the paApprovedBy combo box to open the Properties window. Click the Data tab of the Properties window, if necessary.
2. Select the Row Source property, and click the Build button (...) to launch the Query Builder. You previously selected the Employees table as the value of the Row Source property, so the message box shown in Figure 13.38 appears. Click Yes to confirm the replacement and open the Query Builder window.

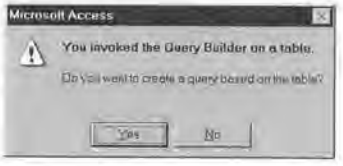


FIG. 13.38 Confirming you want to build a Row Source query based on the Employees table.

3. The Query Builder's window is identical in most respects to the Query Design window, but its title and behavior differ. The Employees table automatically appears in the upper pane. Drag the EmployeeID and LastName fields to columns 1 and 2 of the Query Design grid.
4. You want an ascending sort on the LastName field, so select Ascending in the Sort list box. Your query design appears as shown in Figure 13.39.



When you use the Query Builder, you can test the results of your query by clicking the Run button on the toolbar. Access executes the query and displays a Datasheet view of the query's results.

5. Click the Close window button to close the Query Builder. The message box shown in Figure 13.40 appears for confirmation of your change to the Row Source property value, instead of asking if you want to save your query. Click Yes and the SQL statement derived from the graphical QBE design appears as the value of the Row Source property.
6. SQL statements, especially those created by Access, have a tendency to be lengthy. With the caret in the Row Source property text box, press Shift+F2 to display the SQL statement in the Zoom box, as shown in Figure 13.41.

   In this case, the field name prefix is applied in each field reference, although it is really only necessary in the `FROM` clause because just one table is involved in the query.

Switch to Form view to test the effect of adding the sort (the `ORDER BY` clause) to the query. Writing your own SQL statements to fill combo boxes with values is discussed in Chapter 24, "Working with Structured Query Language."

Oracle Ex. 1009, p. 502
Oracle v. DeBellis
IPR2024-00221

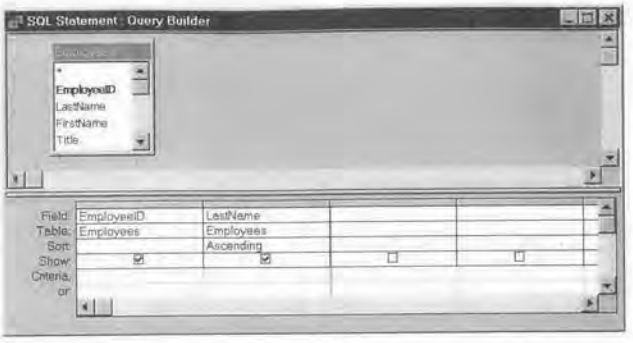


FIG. 13.39 The design of the query to create the SQL statement for the Approved By combo box.

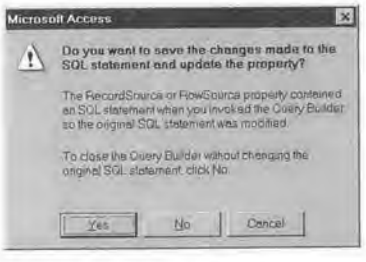


FIG. 13.40 Confirming your change to the Row Source property value.

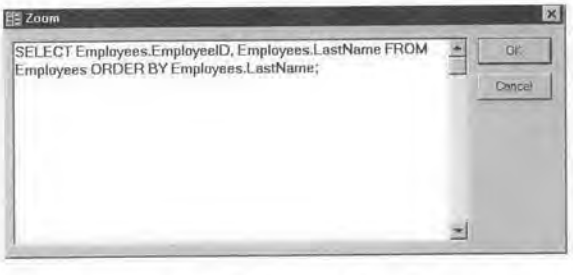


FIG. 13.41 The new SQL statement for the Approved By combo box's Row Source property displayed in the Zoom box.

Creating a Combo Box with a List of Static Values. Another application for list boxes and combo boxes is picking values from a static list of options that you create. A drop-down list to choose a Rating value saves space in a form compared with the equivalent control created with option buttons within an option frame. As you design more complex forms, you find that display "real estate" becomes increasingly valuable.

The option frame you added to the frmPersonnelActionsEntry form provides a choice of only five of the possible 10 ratings. To add a drop-down list with the Combo Box Wizard to allow entry of all possible values, perform the following steps:

Oracle Ex. 1009, p. 503
Oracle v. DeBellis
IPR2024-00221




1. Click the Design View button on the toolbar (if the form isn't already in Design view). Click the Control Wizards button in the Toolbox, if necessary, to enable the Combo Box Wizard (the button should have a sunken appearance).




2. Open the Field List window, then click the Combo Box tool in the Toolbox. Drag the paRating field symbol to a position underneath the Approved By combo box you added previously. The first Combo Box Wizard dialog appears.
3. Select the I Will Type in the Values That I Want option, and then click Next to reach the second dialog.
4. The Rating combo box requires two columns: the first column contains the allowable values of paRating, 0–9, and the second column contains the corresponding description of each rating code. Enter **2** as the number of columns.
5. Access assigns Row Source property values in column-row sequence; you enter each of the values for the columns in the first row, and then do the same for the remaining rows. Type **9 Excellent**, **8 Very Good**, **7 Good**, **6 Average**, **5 Acceptable**, **4 Marginal**, **3 Fair**, **2 Sub-par**, **1 Poor**, **0 Terminated**, as shown in Figure 13.42 (don't type the commas). Click Next to reach the third dialog.

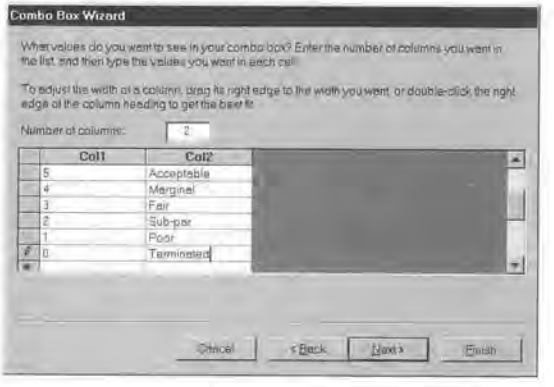


FIG. 13.42 Entering Static values in the Combo Box Wizard's datasheet.

6. Set the widths of the columns you want by dragging the edge of each column header button to the left, as shown in Figure 13.43. If you don't want the rating number to appear, drag the left edge of column 1 fully to the left to reduce its width to 0. When you've adjusted the column widths, click Next to reach the fourth dialog.
7. Select Col1, the rating number code, as the bound column for your value list—that is, the column containing the value you want to store or use later (see Figure 13.44); this column must contain unique values. Click Next to reach the fifth dialog.

Oracle Ex. 1009, p. 504
Oracle v. DeBellis
IPR2024-00221

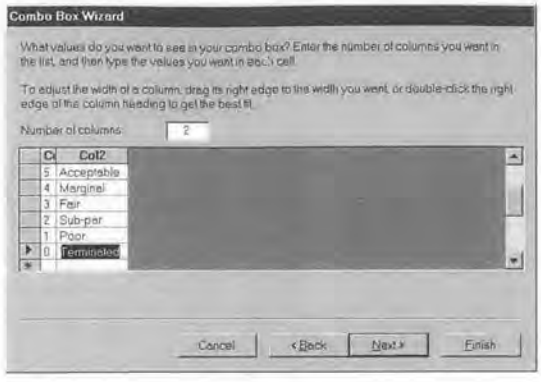


FIG. 13.43 Setting the column widths of the combo box.



FIG. 13.44 Choosing a column to bind to a field of a table or a column of a query.

8. Accept the default value (the paRating field) in this dialog by clicking Next to go to the final dialog of the Combo Box Wizard.
9. Type **Rating:** as the label for the new combo box control, and click Finish to complete the combo box specification and return to Design mode.
10. Open the Properties window for the combo box, then click the Data tab in the Properties window. Set Limit to List to Yes to convert the drop-down combo to a drop-down list. The frmPersonnelActionsEntry form's Data properties appear as shown in Figure 13.45. Notice that Access has added commas after the numbers, semicolons between the row entries, and quotation marks to surround the text values in the Row Source property. You use this format when you enter list values manually.
11. Click the Form View button on the toolbar to display the form. The open Rating static-value combo box appears as shown in Figure 13.46.



Another opportunity to use a static-value combo box is as a substitute for the Type text box. Several kinds of performance reviews exist: Quarterly, Yearly, Bonus, Salary, Commission, and so on, each represented by an initial letter code.

Oracle Ex. 1009, p. 505
Oracle v. DeBellis
IPR2024-00221

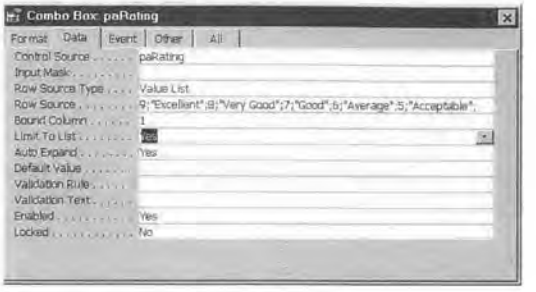


FIG. 13.45 The data properties for the value list combo box.



FIG. 13.46 The Rating static-value combo box opened in Form view.



Note

You can improve the appearance of columns of labels and associated text, list, and combo boxes by right-aligning the text of the labels and left-aligning the text of the boxes. Select all of the labels in a column with the mouse, and click the Align Right button on the toolbar. Then select all of the boxes and click the Align Left button.

Creating a Combo Box to Find Specific Records

The Combo Box Wizard in Access 97 includes a third type of combo list box that you can create—a combo list that locates a record on the form based on a value you select from the list. You can use this type of combo box, for example, to create a Find Last Name box on the frmPersonnelActionsEntry form that contains a drop-down list of all last names from the Employees table. Thus, you can quickly find Personnel Actions records for those employees.

Oracle Ex. 1009, p. 506
Oracle v. DeBellis
IPR2024-00221

To create a combo box that finds records on the form based on a value you select in the combo box, follow these steps:

1. Click the Design View button on the toolbar (if the form isn't already in Design view). Click the Control Wizards button in the Toolbox, if necessary, to enable the Combo Box Wizard (the button should have a sunken appearance).




2. Click the Combo Box tool in the Toolbox, and then click and drag on the surface of the form's Detail section to create the new combo box in a position underneath the Rating drop-down box you created previously. Release the mouse, and the first Combo Box Wizard dialog appears.



3. Click the Find a Record on My Form Based On the Value I Selected in My Combo Box option. Click Next to reach the second dialog.
4. Scroll the Available Fields list until the LastName field is visible. Click to select this field, and then click the > button to move it to the Selected Fields list. Click Next to reach the third dialog.

> **Note**
>
> When creating a combo box to find records, select only one field. The combo box won't work for finding records if you select more than one field for the combo box's lists.

5. The Combo Box Wizard now displays a list of the field values from the column you just selected. Double-click the right edge of the LastName column to get the best column-width fit for the data values in the column, and then click Next to go to the fourth and final step of the wizard.
6. Type **Zoom to:** as the label for the new combo box, and then click Finish to complete the new combo box control. Your form should appear as shown in Figure 13.47.
7. Click the Form View button on the toolbar to display the form. The open Zoom To combo box appears as shown in Figure 13.48.



When you create this type of combo box, the Combo Box Wizard automatically creates an Access VBA event procedure for the After Update property of the combo box (refer to the Property window in Figure 13.47). An event procedure is a VBA procedure that Access executes automatically whenever a particular event occurs—in this case, updating the combo box. Chapter 28, "Writing Visual Basic for Applications Code," describes how to write Access VBA code and Chapter 30, "Responding to Events with VBA 5.0" describes how to write event-handling procedures.

To view the event procedure code that the wizard created for your new combo box, open the Properties window for the Zoom to: combo box, click the Events tab in the window, select the After Update property text box, and then click .... Access opens the Code Builder window shown in Figure 13.49. After you've looked at the code, click the Close window button to close the Code Builder window and return to Design mode.

Oracle Ex. 1009, p. 507
Oracle v. DeBellis
IPR2024-00221

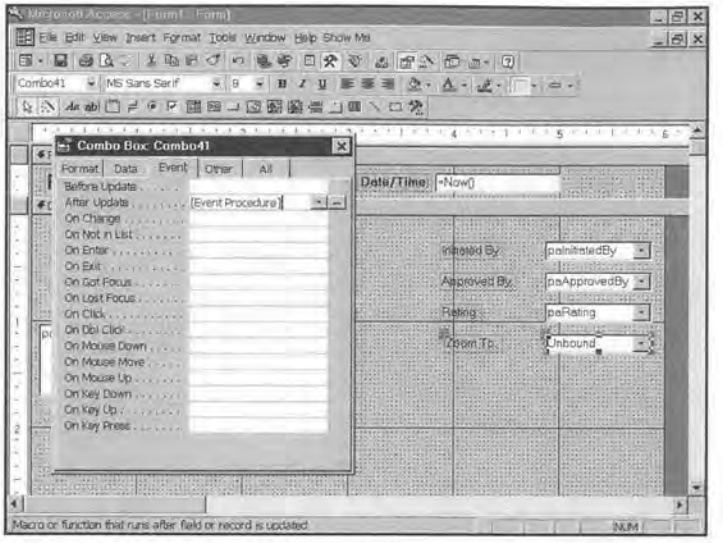


FIG. 13.47 The new combo box that will find a record on the form based on a value selected in the combo box.

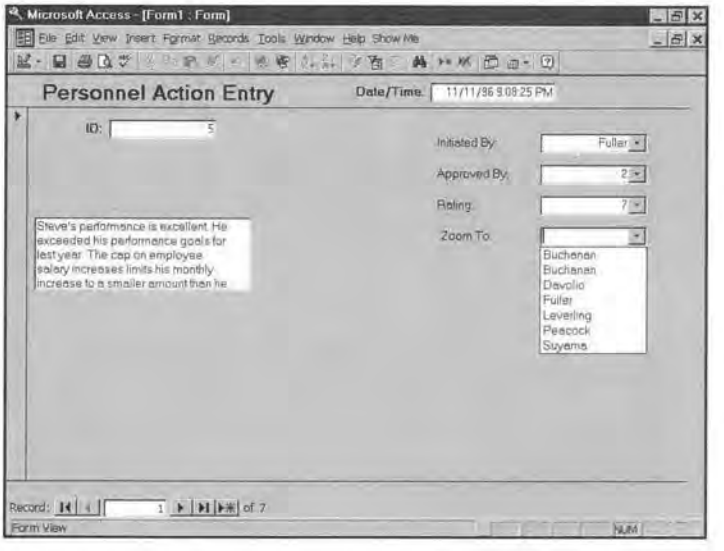


FIG. 13.48 The combo box for finding a record on the form in Form view.

To use a combo box of this type, select a value from the list. As soon as you select the new value, Access updates the combo box's text box, which then invokes the Access VBA code for the After Update event procedure. The VBA code in the After Update procedure finds the first record in the form's record set with a matching value and displays it. You can only use this type of combo box to find the first matching record in a record set.

Oracle Ex. 1009, p. 508
Oracle v. DeBellis
IPR2024-00221

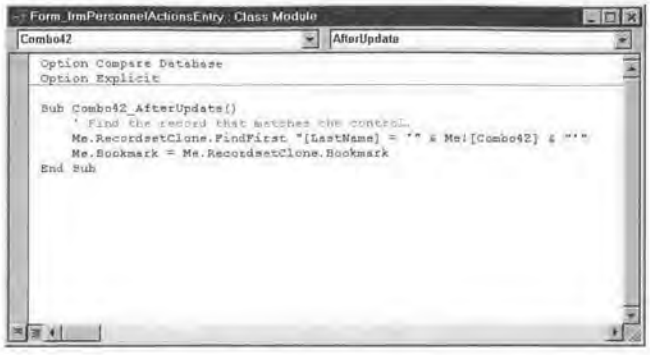


FIG. 13.49 The Module window for the After Update event property of the locating combo box.

Because the field on the form is based on the LastName column of the form's underlying query, you'll see an entry in the list for each and every last name entry in the record set produced by the qryPersonnelActions query. If, let's say, more than one Personnel Action record exists for Steve Buchanan, then Buchanan appears in the combo list as many times as there are records for him. To display a unique list of last names to be located on the form, change the Row Source property to obtain the LastName field values for the combo box list through a SQL statement based on a query from the Employees table.

To change the Row Source property, follow the procedure you learned in the "Using the Query Builder to Populate a Combo Box" section, earlier in this chapter: open the Properties window of the Zoom To combo box, click the Data tab, select the Row Source text box, and then open the Query Builder. Change the query so that it uses the Employees table, as shown in Figure 13.50. You should also change the Limit to List property value to Yes.

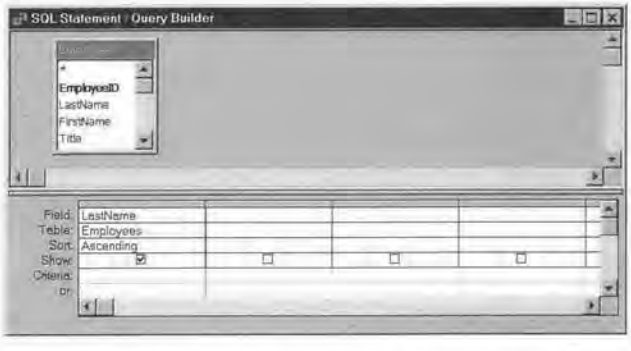


FIG. 13.50 The Query Builder window for the new combo box, showing the Employees table selected as the new Row Source property value for the combo box.

Creating a Tab Control

The Tab control is an important addition to Access 97's Toolbox. With the Tab control, you can easily create multipage forms in a tabbed dialog, similar to the tabbed pages you've seen in the Properties window, in the Options dialog, and elsewhere in Access.



Oracle Ex. 1009, p. 509
Oracle v. DeBellis
IPR2024-00221

The Tab control is a relatively easy and very efficient alternative to creating multipage forms with the Page Break control. You can use the Tab control to conserve space on-screen, and to show information from one or more tables. This section shows you how to add a Tab control to a form. You also learn to set the important properties of the Tab control as a whole, as well as the properties of individual pages of the Tab control.

To add a Tab control to the frmPersonnelActionsEntry form, follow these steps:



1. Click the Design View button on the toolbar if the frmPersonnelActionsEntry form isn't already in Design view. No wizard for the Tab control exists, so the status of the Control Wizards button doesn't matter.



2. Click the Tab Control tool in the Toolbox; the mouse cursor changes to the Tab Control icon while it is over the active surface of the form.

3. Click and drag on the surface of the Detail section of the form to create the new tab control near the bottom center of the form (see Figure 13.51).

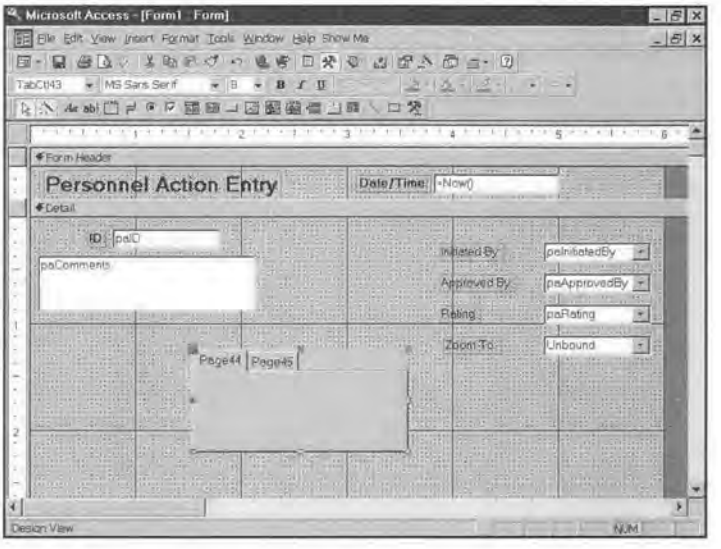


FIG. 13.51 A new Tab control showing the default two pages and default page captions.

Access creates the new Tab control when you release the mouse button, as shown in Figure 13.51. By default, Access creates a Tab control with two pages. Each page's tab displays the name of the page combined with a sequential number corresponding to the number of controls you placed on your form in this work session. The next few sections describe how to change the page tab's caption, add or delete pages in the Tab control, add controls to the pages, and set the page and Tab control properties.

Oracle Ex. 1009, p. 510
Oracle v. DeBellis
IPR2024-00221

Adding Tab Control Pages. Depending on the data you want to display and how you want to organize that data, you may want to include more than two pages in your Tab control. To add a page to a tab control, follow these steps:

1. Click the tab of the page you want the new page inserted in front of. Access brings the page you select to the front of the Tab control.
2. Using the right mouse button, click the edge of the Tab control (the blank space along the top of the tab rows is easiest). Access displays the popup menu shown in Figure 13.52.
3. Choose Insert Page; Access inserts a new page in the Tab control in front of whichever existing page is currently selected.

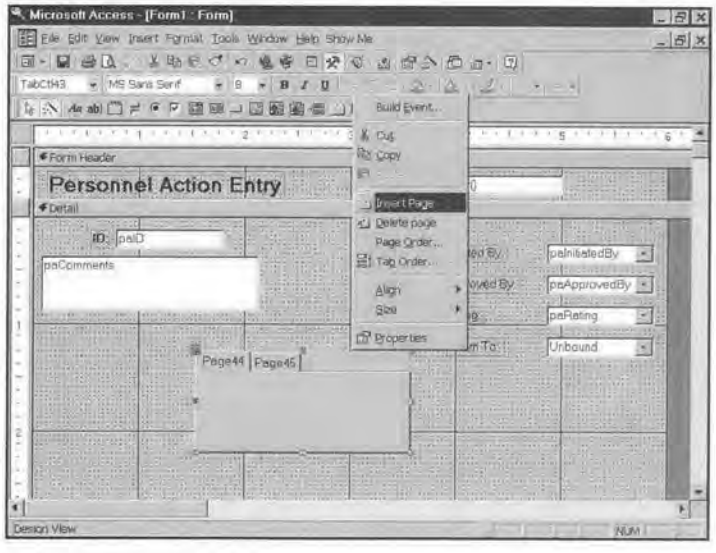


FIG. 13.52 The popup menu for adding, deleting, or changing the order of pages in a Tab control.

Changing the Page Order. Because Access adds new pages to a Tab control in front of the currently selected page, it isn't possible to add a new page at the end of the existing tab pages. As a result, if you want the new Tab control page to appear as the last page in the Tab control, you'll need to change the order of pages in the Tab control. You may also want to change the order of Tab control pages as you work with and test your forms—in general, you should place the most frequently used (or most important) page at the front of the Tab control. To change the order of pages in a Tab control, follow these steps:

1. Using the right mouse button, click the edge of the Tab control. Access displays the Tab control's popup menu (see Figure 13.52).

Oracle Ex. 1009, p. 511
Oracle v. DeBellis
IPR2024-00221

2. Choose Page Order; Access displays the Page Order dialog shown in Figure 13.53.
3. In the Page Order list, select the page whose position you want to change.
4. Click the Move Up or Move Down buttons, as appropriate, until the page is in the position you want.
5. Repeat steps 3 and 4 until you have arranged the Tab control pages in the order you want, and then click OK to close the Page Order dialog and apply the new page order to the Tab control.



FIG. 13.53 Changing the page order of the tab control in the Page Order dialog.

Deleting a Tab Control Page. At some point, you may decide that you don't want or need a page in a Tab control. The frmPersonnelActionsEntry form only needs two pages in its Tab control. If you added a page to the Tab control by following the steps at the beginning of this section, you can delete a page from the Tab control by following this procedure:

1. Click the page tab of the page you want to delete; Access brings that page to the front of the Tab control.
2. Right-click the edge of the Tab control; Access displays the Tab control's popup menu (refer to Figure 13.52).
3. Choose Delete Page; Access deletes the currently selected Tab control page.

Setting the Tab Control's Properties. Two sets of properties govern the appearance and behavior of a tab control. A set of properties exists for the entire Tab control, and a separate set of properties exists for each page in the Tab control. The following list summarizes the important properties of the Tab control and its pages; the remaining property settings for the Tab control and its pages are similar to those you've seen for other controls (height, width, color, and so on):

- *Caption.* This text property controls the text that appears on the page's tab and applies to individual Tab control pages only. If this property is blank (the default), then the page's Name property is displayed on the page's tab.
- *MultiRow.* This Yes/No property applies to the Tab control as a whole and controls whether the Tab control can display more than one row of tabs. (The Options

Oracle Ex. 1009, p. 512
Oracle v. DeBellis
IPR2024-00221

dialog, reached by choosing Tools, Options, is an example of a multirow tabbed dialog.) The default setting is No; in this case, if there are more tabs than fit in the width of the Tab control, Access displays a scroll button in the Tab control. If you change this property to Yes and there are more page tabs than will fit in the width of the Tab control, Access displays multiple rows of tabs.

- *Picture*. You can display an icon in any or all of the page tabs in a Tab control using this property, which applies to pages only. You can use any of Access's built-in icons or insert any bitmapped (.bmp) graphic file as the page's tab icon.
- *Style*. The Style property applies to the Tab control as a whole and controls the style in which the Tab control's page tabs are displayed. The default setting, `Tabs`, produces the standard page tabs you're accustomed to seeing in the Properties window and in various dialogs in Access and Windows. Two other settings are available: `Buttons` and `None`. The `Buttons` setting causes the page tabs to display as command buttons in a row across the top of the Tab control. The `None` setting causes the Tab control to omit the page tabs altogether. Use the `None` setting if you want to control which page of the Tab control has the focus with command buttons or option buttons located outside of the Tab control. However, using command buttons external to the Tab control to change pages requires writing Access VBA program code. You should use the default `Tabs` setting unless you have a very specific reason for doing otherwise—using the `Tabs` setting ensures that the appearance of your Tab controls is consistent with other portions of the Access user-interface. Using this setting also saves you the effort of writing VBA program code.
- *TabFixedHeight* and *TabFixedWidth*. These two properties, which apply to the Tab control as a whole, govern the height and width of the page tabs in the control, respectively. The default setting for these properties is `0`. When these properties are set to 0, the Tab control sizes the page tabs to accommodate the size of the Caption for the page. If you want all of the page tabs to have the same height or width, enter a value (in inches or centimeters, depending on your specific version of Access) in the corresponding property text box.

To display the Properties window for the entire Tab control, right-click the edge of the Tab control, and select Properties from the resulting popup menu (see Figure 13.52). Alternatively, click the edge of the Tab control to select it (clicking the blank area to the right of the page tabs is easiest), and then click the Properties button on the toolbar to display the Properties window.



To display the Properties window for an individual page in the Tab control, click the page's tab to select it, and then click the Properties button on the toolbar to display the page's Properties window.



The Tab control in the frmPersonnelActionsEntry form uses one page to display company information about an employee: the employee's job title, supervisor, company telephone extension, hire date, and photo. The second Tab control page displays a history of that employee's personnel actions. Follow these steps to set the Caption property for the frmPersonnelActionsEntry form's Tab control:

Oracle Ex. 1009, p. 513
Oracle v. DeBellis
IPR2024-00221



1. Click the Design View button on the toolbar if the frmPersonnelActionsEntry form isn't already in Design view.



2. Click the first page of the Tab control to select it, and then click the Properties button on the toolbar to display that page's Properties window (see Figure 13.54).
3. Click the Format tab, if necessary, to display the Format properties for the Tab control page.
4. Type **Company Info** in the Caption property's text box.
5. Click the second page of the Tab control to select it; the contents of the Property dialog change to show the properties of the second Tab control page.
6. Type **History** in the Caption property text box for the second page of the Tab control, and close the Properties window.

Figure 13.54 shows the Tab control with both page captions set and the first page of the Tab control selected. Notice that the sizing handles visible in the Tab control are *inside* of the control—this position indicates that the page, not the entire control, is currently selected. When the entire Tab control is selected, the sizing handles appear at the edge of the Tab control.

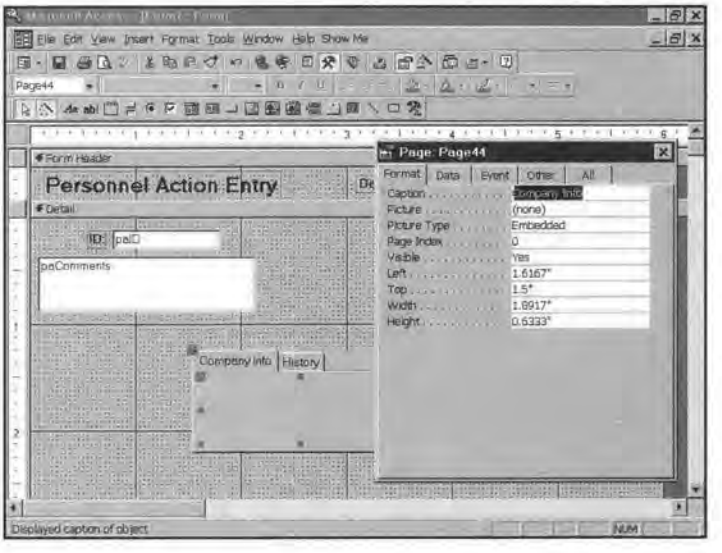


FIG. 13.54 Setting the Caption property of a page in the Tab control.

Placing Other Controls on Tab Pages. You can place any of Access's 16 other types of controls on the pages of a Tab control—labels, text boxes, list boxes, even subforms. To add a control of any type to a Tab control's page, follow this procedure:

1. In Design view, click the page tab you want to add the control to; Access selects the page and brings it to the front of the Tab control.

Oracle Ex. 1009, p. 514
Oracle v. DeBellis
IPR2024-00221

2. Add the desired control to the Tab control's page using the techniques presented earlier in this chapter for creating controls on the Detail or Header/Footer sections of a form.

   Alternatively, you can copy controls from the same or another form and paste them into the Tab control's pages by using the same techniques you learned for copying and pasting controls on a form's Detail and Header/Footer sections. You cannot drag controls from the form's Detail or Header/Footer sections onto the Tab control's page.

As you proceed with the examples in this chapter and complete the frmPersonnelActionsEntry form, you'll place various bound and unbound controls on the pages of the Tab control.

Changing One Control Type to Another

If you made a mistake in selecting the type of control in earlier versions of Access, you would have to delete the control and start over. Access 97 allows you to "morph" a control of one type to become a control of a compatible type. You can change an option button to a check box, for example, or you can change a toggle button to an option button. You can't, however, change a text box to an object frame or other control with a different field data type. To change a control to a different type, follow these steps:

1. In the form's Design view, select the control whose type you want to change.
2. Choose Format, Change To to see a submenu of form control types. Only the submenu choices for control types that are compatible with the selected control are enabled.
3. Select the control type you want from the submenu. Access changes the control type.

Completing the Main Personnel Actions Entry Form

In the following sections of this chapter, you learn how to use the Control Wizards to help you add a subform to a form. Before you add the subform, however, you should complete the main frmPersonnelActionsEntry form. Like the form that you created with the Form Wizard in Chapter 12, the purpose of this form is to display records from the Personnel Actions table so that a user can view the history of an employee's personnel actions. The form also conveniently provides a means of adding new personnel action records.

In the form you created in Chapter 12, you viewed the history of personnel records and also added new records in a tabular subform, while information from the Employees table was displayed only on the main form. In this custom form, you place fields from

Oracle Ex. 1009, p. 515
Oracle v. DeBellis
IPR2024-00221

the Personnel Actions table on the main form to make adding new records to the Personnel Actions table easier, and the Tab control is used to contain the subform as well as additional information from the Employees table. The subform on the second page of the Tab control displays only historic personnel action records. The frmPersonnelActionsEntry form has fields from the Personnel Actions table on both the main form and the subform, and it uses the qryPersonnelActions form you created at the beginning of this chapter as the form's data source. If you were creating a full-scale human resources database application, you might choose the Employees table as the data source for the form and design a subform for editing the Personnel Actions table. You might then put this subform on a third Tab control page or display the history in a subform of the subform.

Note

Creating a form and subform that are both based on the same underlying table—in this example, the Personnel Actions table—is a somewhat unconventional but totally acceptable database application design method. Most forms that employ subforms employ a base table (such as Employees) or a query whose data source is a base table as the record source of the main form. A related table or query based on a related table serves as the record source of the subform. Many of the forms of Northwind.mdb demonstrate the conventional form-subform design. Our form example, frmPersonnelActionsEntry, uses a common underlying table for both the form and subform to illustrate some of the unique characteristics of this approach to one-to-many form design.

To complete the main form, follow these steps (refer to Figure 13.55 for field placement):



1. Click the Design View button on the toolbar (if the frmPersonnelActionsEntry form isn't already in Design view).





2. If necessary, click the Toolbox button to enable the Control Wizards (make sure the button has a sunken appearance), and then click the Field List button on the toolbar to open the Field List dialog if it isn't already open.
3. Drag the LastName field from the Field List to a position to the right of the ID field text box; when you release the mouse, Access creates a text box for the field. Edit the field's label to read **Name:**.
4. Drag the FirstName field from the Field List to a position to the right of the LastName field; delete the FirstName field's label.
5. Drag the paType field from the Field List to a position at the right of the FirstName field.
6. Repeat step 5 for the paScheduledDate, paEffectiveDate, and paAmount fields (refer to Figure 13.55 for field positioning and sizing). You'll need to move the Approved By, Rating, and Zoom To fields that you placed on the form earlier in this chapter.

Oracle Ex. 1009, p. 516
Oracle v. DeBellis
IPR2024-00221

7. Resize the paComments field so that it is underneath the paID and name fields (see Figure 13.55). Next, resize the Tab control so that it fills the width of the form and extends from an area below the paComments field to the bottom of the form. The Tab control needs to be as large as possible in order to display the most data in the subform that you'll later add to its second page.
8. Click the first tab of the Tab control to bring it to the front, and then drag the Title field from the Field List to a position near the top left corner of the Company Info page.
9. Repeat step 8 for the ReportsTo, Extension, and HireDate fields (refer to Figure 13.55 for field placement).
10. Drag the Photo field onto the right side of the Tab control's first page and delete its label (the fact that this field displays a photo of the employee is enough to identify the field). Size and position the Photo field at the right edge of the Tab control's page; you may need to resize the Tab control and the form after inserting the Photo field.
11. Double-click the Photo field to display its Properties window, click the Format tab, and select the Size Mode property's text box (see Figure 13.56). Select Zoom from the drop-down list to have the employee photo scaled down to fit the photo field's size.

Tip

Use the Format Painter to format the text labels of the fields. Using the Format Painter is described in Chapter 12, "Creating and Using Forms."

12. Drag the Notes field onto the bottom left side of the Tab control's first page and delete its label. Refer to Figure 13.55 for placement and sizing.
13. Use the techniques you learned in Chapter 12 to move, rearrange, and change the label formats to match the appearance of Figure 13.55. (All labels are bold and right-aligned.)
14. Test your new fields by clicking the Form View button on the toolbar. Your form appears as shown in Figure 13.57.

Note

In Figure 13.55, all of the toolbars and the Form Design window's rulers have been turned off so that you can see the entire form in Design view.

Oracle Ex. 1009, p. 517
Oracle v. DeBellis
IPR2024-00221

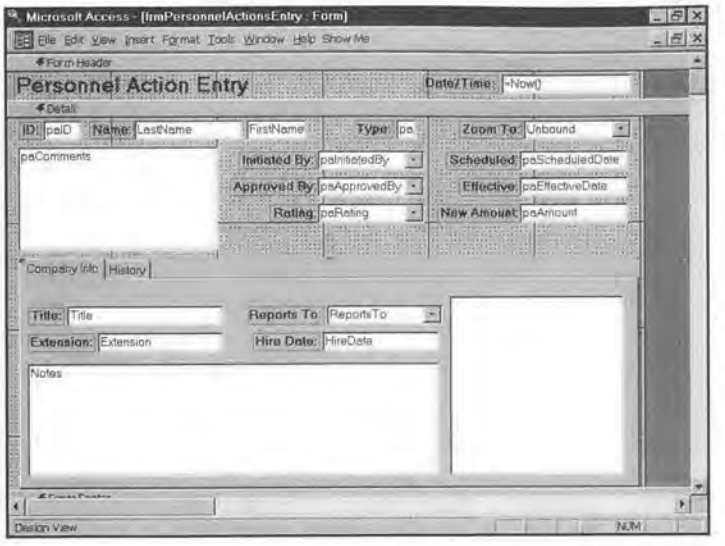


FIG. 13.55 The frmPersonnelActionsEntry form in Design view, showing the final placement and formatting of the main form fields and the first page of the tab control.



FIG. 13.56 Setting the Photo field's Size Mode property so that photos are scaled to fit the size of the field on the form.

Oracle Ex. 1009, p. 518
Oracle v. DeBellis
IPR2024-00221



FIG. 13.57 The frmPersonnelActionsEntry form of Figure 13.55 displayed in Form view.

Creating a Subform Using the Subform/Subreport Wizard

The frmPersonnelActionsEntry form needs a subform in which to view the history of personnel actions for the employee displayed in the main part of the form. Access 97's new Subform/Subreport Wizard makes it possible for you to create a new subform at the same time that you add the subform field to the main form or, as in this example, a page in a Tab control. To do so, follow these steps:

1. Click the Design View button on the toolbar if the frmPersonnelActionsEntry form isn't already in Design view.



2. Click the Control Wizards button to enable the Control Wizards if the button isn't already down.



3. Click the second tab of the Tab control to bring its second page to the front. Click the Subform button in the Toolbox, and then click at the top-left corner of the second tab control page. Access displays the first dialog of the Subform/Subreport Wizard.



4. You can use this wizard either to create a new subform based on a table or query or to insert an existing subform (see Figure 13.58). (You learn how to insert an existing form as a subform in the section "Creating and Using Continuous Forms" later in this chapter.) For this exercise, select the Table/Query option, and click Next to reach the second dialog.

5. The Subform/Subreport Wizard asks you to indicate which table or query the new subform is based on and which fields appear in the subform (see Figure 13.59).

Oracle Ex. 1009, p. 519
Oracle v. DeBellis
IPR2024-00221

Select Query: qryPersonnelActionsSubform in the Tables and Queries drop-down list. To expedite field selection, click the >> button to copy all of the fields to the Selected Fields list. Select the paComments field in the Selected Fields list, and click the < button to remove this field from the list. Click Next to reach the third dialog.



FIG. 13.58 Choosing whether to use an existing form as the subform or to use a table or query to create a new subform.



FIG. 13.59 Selecting the data source and fields for the new subform.

6. The wizard now asks you to specify the link between the main form and the subform. You may select from a list of possible relationships that Access has determined, or define your own link. Click the Define my own option, and the Wizard dialog changes to show four drop-down list text boxes (see Figure 13.60).

7. In the upper Form/Report Fields list, select paID as the linking field; in the upper Subform/Subreport Fields list, also select paID as the linking field. Click Next to go to the fourth and final dialog of the wizard.

8. Type **sbfTest** as the name of this new subform, and click Finish to complete the subform's specifications (see Figure 13.61). Access creates and saves the new form; it inserts the completed subform into the subform field on the main form and sizes

Oracle Ex. 1009, p. 520
Oracle v. DeBellis
IPR2024-00221

the subform field to accommodate the new subform (see Figure 13.62). The text you entered for the subform's name is assigned to the label for the subform. The subform itself is saved under the same name.

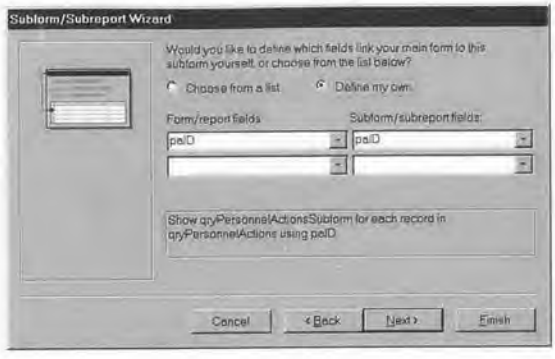


FIG. 13.60 Defining the link between the main form and the subform.



FIG. 13.61 Entering a name for your new subform.

9. Click the Form View button on the toolbar to check the appearance of the new subform. Your form appears as shown in Figure 13.63.



As you can see from Figure 13.63, the Subform/Subreport Wizard always creates new subforms with Datasheet view. In many cases, this is acceptable—or even desirable. For the frmPersonnelActionsEntry form, however, a better view of the data can be achieved with a tabular continuous form view of the data. The following sections of this chapter explain the advantages of a tabular continuous form and guide you through the steps necessary to create such a form and insert it as a subform into the main form.

Oracle Ex. 1009, p. 521
Oracle v. DeBellis
IPR2024-00221

10. Close the frmPersonnelActionsEntry form by clicking the Close window button. Click No when Access asks if you want to save changes to the form's design.

You'll create a much more useful form in the next section.

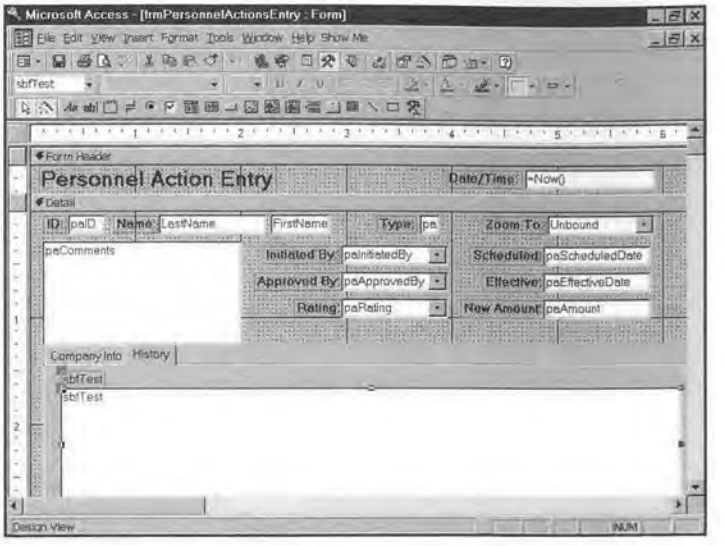


FIG. 13.62 The completed subform field in Design view.

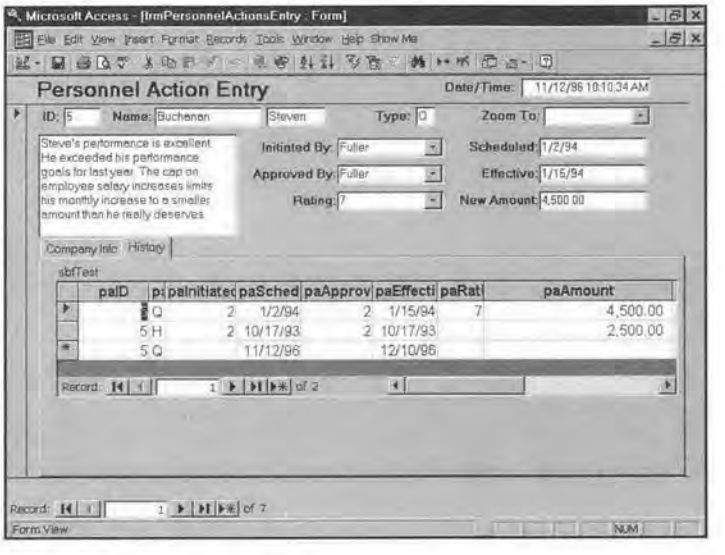


FIG. 13.63 The new subform in Form view.

Oracle Ex. 1009, p. 522
Oracle v. DeBellis
IPR2024-00221

Creating and Using Continuous Forms

Continuous forms are useful for displaying data contained in multiple records of a table or query in a format other than Datasheet view. The sbfPersonnelActions subform you created in Chapter 12, for example, is designed only to display the most recent Personnel Action records for an employee. Editing isn't allowed in the subform, so you don't need the field headers, record selection buttons, and scroll bars associated with Datasheet view. These graphic elements focus more attention on the subform than is deserved. You need a plain vanilla display of the history for the employee; this basic display requires a continuous form.

The Form Wizard offers the choice of creating a tabular continuous form, so using the Form Wizard is the quickest method of creating a plain, vanilla subform. To create a tabular continuous form with the Form Wizard, perform the following steps:

1. Click the Forms tab, then click New in the Database window to create a new form. Access displays the New Form dialog.



2. Select qryPersonnelActionsSubform as the source of data for the new form.
3. Select Autoform: Tabular from the list at the top of the New Form dialog, then click OK. Access immediately creates a tabular form based on the fields in the query qryPersonnelActionsSubform and then displays the form in Form view (see Figure 13.64).

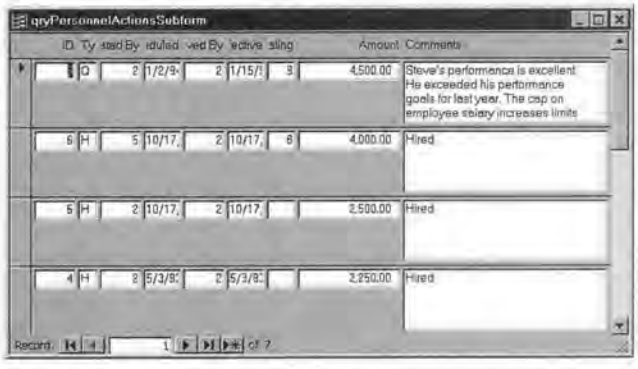


FIG. 13.64 The tabular form produced by the Form Wizard from qryPersonnelActionsSubform.

4. Click the Save button on the toolbar; Access displays the Save As dialog.



5. Type **sbfTest1** as the name of the form, and click OK.

The Form Wizard created the tabular form using the qryPersonnelActionsSubform query as the data source. To customize the form to make the size and appearance compatible with the frmPersonnelActionsEntry form, follow these steps:

Oracle Ex. 1009, p. 523
Oracle v. DeBellis
IPR2024-00221



1. Click the Design View button to switch the sbfTest1 form to Design view, and then maximize the form window.
2. Delete the paID and paComments fields from the Detail section of the form. Delete the paID and paComments labels from the Form Header section of the form.



3. Click the Form Header bar, and use the Fill/Back Color button on the Formatting toolbar to change the background color (Back Color property) to white. Change the background color of the Detail section to white, also.
4. Drag a selection box around all of the labels in the Form Header section. Click the Bold button on the Formatting toolbar, and then click the Center button.
5. Choose Format, Size, to Fit; then choose Format, Size, To Grid. This sequence of commands makes the label text boxes large enough to display their entire contents and then sizes them to the nearest regular grid mark. This sizing makes it easier to position the labels into columns with the fields in the Detail section of the form.
6. Choose Format, Horizontal Spacing, Make Equal to spread the field labels apart evenly so that they are touching each other without overlapping.
7. Drag the Type label to the upper-left corner of the Form Header section; position the remaining labels along the top edge of the same section, one grid mark apart from each other (see Figure 13.65). Then drag the Detail section header bar to the bottom of the labels.

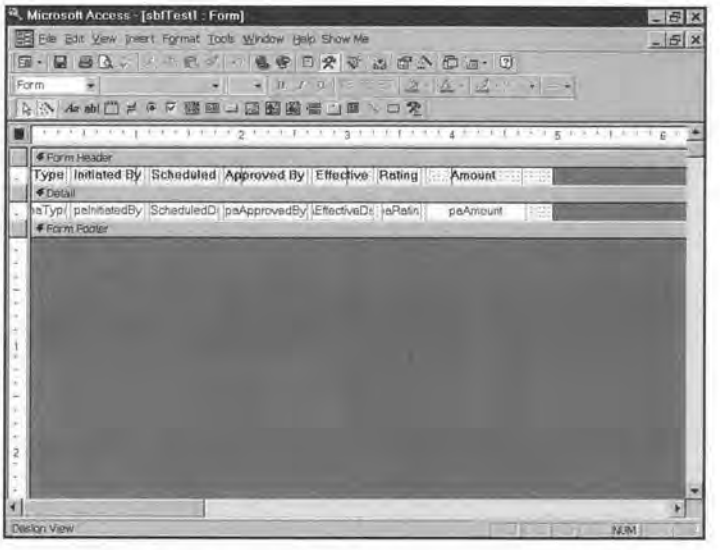


FIG. 13.65 The modified tabular form in Design view.

Oracle Ex. 1009, p. 524
Oracle v. DeBellis
IPR2024-00221

8. Drag the paType field to the upper-left corner of the Detail section, and resize it to match the Type label in the Form Header section. Position each of the remaining fields from left to right under the corresponding label in the Form Header section, sizing each field to match its label. Drag the Form Footer section header bar to the bottom of the text boxes.

9. Drag a selection box around all of the fields in the Detail section of the form, and then click the Center button on the formatting toolbar to center the data in each field. Click the Line/Border Color button, and then click Transparent to make all of the fields' borders transparent.




10. Choose Edit, Select Form, and then click the Properties button on the toolbar.



11. Click the Data tab, and set the value of the Allow Edits property to No, the Allow Deletions property to No, and the Allow Additions property to No. These changes prevent records displayed by this form from being edited, added to, or deleted.

12. Click the Format tab, set the value of the Scroll Bars property to Neither, and then set the Record Selectors and Navigation Buttons properties to No. The default GridX and GridY property values of 24 correspond to the grid spacing of the frmPersonnelActionsEntry form.

13. Close the Properties window, and drag the right edge of the form leftward until the form is just under 5 inches wide. Your form should look like the one you saw in Figure 13.65.

14. Click the Form View button on the toolbar. The continuous form displays all records in the Personnel Actions subquery, as shown in Figure 13.66.



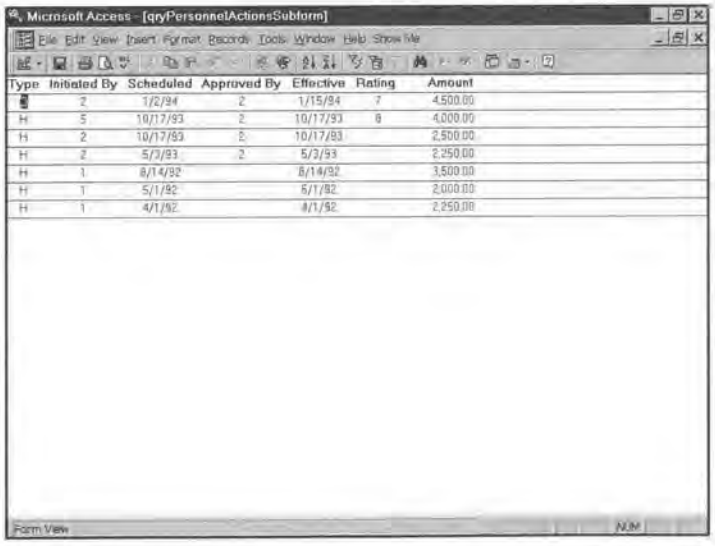


FIG. 13.66 The appearance of the continuous form of Figure 13.65 in Form view.

Oracle Ex. 1009, p. 525
Oracle v. DeBellis
IPR2024-00221

15. Choose File, Save As/Export. Select Within the Current Database As, and type **sbfPersonnelActionsTab** in the New Name text box. Click OK to save the form under its new name. Click the Close window button to close the form.

Now, you need to add the tabular form you just created as a subform in the Tab control of the frmPersonnelActionsEntry form. You'll use the Subform/Subreport Wizard to insert this form into the Tab control on your main form. (Whether you add the subform directly on a form or onto a page of a Tab control, the procedure is the same.) To complete this procedure, perform the following steps:



1. Open the frmPersonnelActionsEntry form, and click the Design View button on the toolbar.



2. If necessary, click the Control Wizards button on the toolbar to enable the Control Wizards (the button position should be down).

3. On the Tab control, click the tab of the second page to bring it to the front. (This is the History tab you created earlier in this chapter.)



4. Click the Subform button on the Toolbox, and then click at the upper-left corner of the Tab control's second page. Access displays the first dialog of the Subform/Subreport Wizard.

5. To insert an existing form, select the Forms option, and then select sbfPersonnelActionsTab in the drop-down list (see Figure 13.67). Click Next to reach the second dialog.



FIG. 13.67 Using the Subform/Subreport Wizard to insert an existing form as a subform.

6. The wizard asks you to specify the link between the main form and the subform. You may select from a list of possible relationships that Access has determined or define your own link. Click the Define my own option. Select paID in the upper Form/Report Fields list, and select paID in the upper Subform/Subreport Fields list. Click Next to go to the third dialog.

Oracle Ex. 1009, p. 526
Oracle v. DeBellis
IPR2024-00221

7. Accept the default label name for the new subform field, and click Finish to complete the new subform field. Access now inserts the form you specified as a subform on the second page of the Tab control and sizes the subform field to accommodate the new subform, as shown in Figure 13.68.

8. Delete the label from the subform field, and resize the field so that it fills the width of the main form. Scroll the Form Design window downward, and resize the subform field so that it fills the Tab control page (see Figure 13.69). At this point, your form should be approximately 3.5 inches high, and the Tab control should be about 2 inches high.

9. Click the Form View button on the toolbar to check the appearance of the new subform. Your form appears as shown in Figure 13.70 after clicking the History tab to display the second page of the Tab control. (Refer to Figure 13.57 for a view of the completed form showing the Company Info page of the Tab control.) If the subform displays scroll bars or a record selector, close the Personnel Action Entry form and reopen it from the Database window to remove these features (called *adornments*).

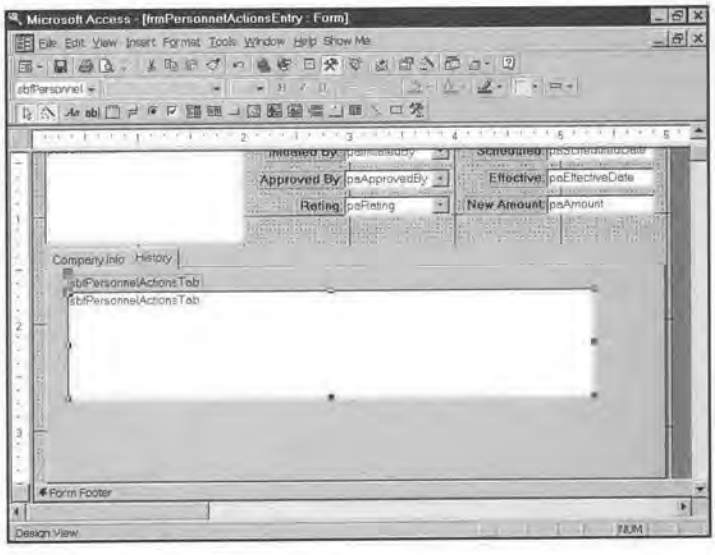


FIG. 13.68 The new subform field for the sbfPersonnelActionsTab form, inserted by the Subform/Subreport Wizard.

Oracle Ex. 1009, p. 527
Oracle v. DeBellis
IPR2024-00221

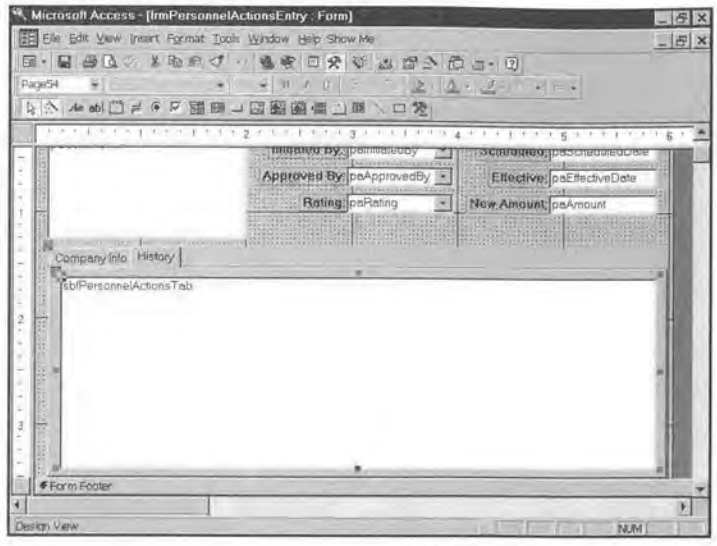


FIG. 13.69 The modified sbfPersonnelActionsTab form in Design view.

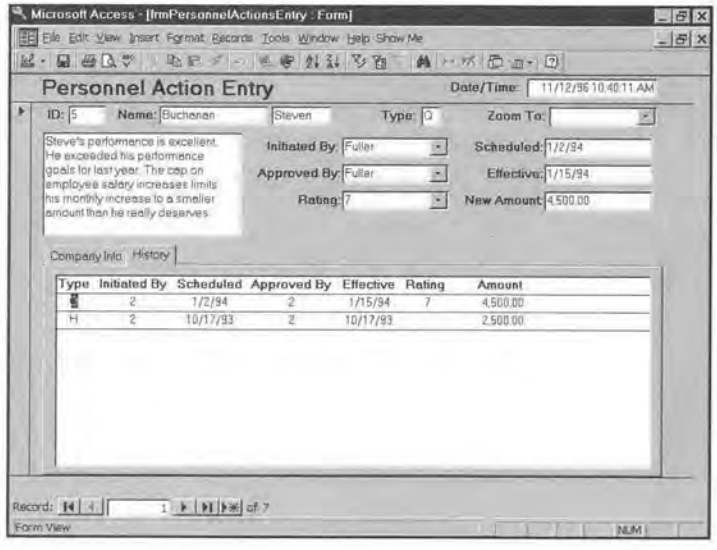


FIG. 13.70 The completed subform and main form in Form view.

Oracle Ex. 1009, p. 528
Oracle v. DeBellis
IPR2024-00221

> **Note**
>
> Delete the vertical scroll bar when all control objects of the noncontinuous form fit in a maximized form window. The horizontal scroll bar need not be present when record selectors are used. Unnecessary graphic elements are distracting and have a negative influence on the overall appearance of a form.

10. Choose File, Save, and then click the Close window button to close the frmPersonnelActionsEntry form.

The frmPersonnelActionsEntry form's fields and field formatting are essentially complete. In a real-world application, you would now adjust the tab order of the form and test the form by entering and editing records—you learned to do these steps in Chapter 12, "Creating and Using Forms."

Overriding the Field Properties of Tables

Access uses the table's property values assigned to the fields as defaults. The form or subform inherits these properties from the table or query on which the form is based. You can override the inherited properties, except for the Validation Rule property, by assigning a different set of values in the Properties window for the control. Properties of controls bound to fields of tables, or queries that are inherited from the table's field properties are shown in the following list:

- Format
- Decimal Places
- Status Bar Text
- Validation Rule
- Validation Text
- Default Value
- Typeface characteristics (such as Font Name, Font Size, Font Bold, Font Italic, and Font Underline)

Values of field properties that you override with properties in a form apply only when the data is displayed and edited with the form. You can establish validation rules for controls bound to fields that differ from properties of the field established by the table, but you can only narrow the rule. The table-level validation rule for the content of the paType field, for example, limits entries to the letters H, S, Q, Y, B, and C. The validation rule you establish in a form cannot broaden the allowable entries; if you add T as a valid choice by editing the validation rule for the paType field to `InStr("HSQYBCT",[PA Type])>0`, you receive an error when you type **T**.

Oracle Ex. 1009, p. 529
Oracle v. DeBellis
IPR2024-00221

However, you can narrow the range of allowable entries by substituting `InStr("SQYB",[PA Type])>0`. Notice that you can use expressions that refer to the field name in validation rule expressions in forms; such expressions are not permitted in table validation rule expressions in Access 97.

Adding Page Headers and Footers for Printing Forms

Access allows you to add a separate pair of sections, Page Header and Page Footer, that appear only when the form prints. You add both of these sections to the form at once by choosing View, Page Header/Footer. The following list shows the purposes of Page Headers and Footers:

- *Page Header* sections enable you to use a different title for the printed version. The depth of the Page Header can be adjusted to control the location where the Detail section of the form is printed on the page.
- *Page Footer* sections enable you to add dates and page numbers to the printed form.

Page Header and Page Footer sections appear only in the printed form, not when you display the form on-screen in Form view. The frmPersonnelActionsEntry form in Design mode with Page Header and Page Footer sections added is shown in Figure 13.71. The Tab control has been deleted so that both the Form Footer and Page Footer sections appear in the window. Usually, you need to use the vertical scroll bar to display these sections in Design mode.

With the Display When (Format) property of the Properties window for the Form Header and Form Footer sections, you can control whether these sections appear in the printed form. In Figure 13.71, the Form Header duplicates the information in the Page Header (except for the Date/Time label and text box), so you don't want to print both. To control when a section of the form prints or is displayed, perform the following steps:

1. Double-click the title bar of whichever section of the form you want to change; this opens the related Properties window. (The Page Header and Page Footer sections don't have a Display When property; these sections only appear when printing.)
2. Click the Format tab if the formatting properties aren't already showing. Click to drop down the Display When list.
3. To display but not print this section in Form view, select Screen Only.
4. To print but not display this section, select Print Only.

Oracle Ex. 1009, p. 530
Oracle v. DeBellis
IPR2024-00221

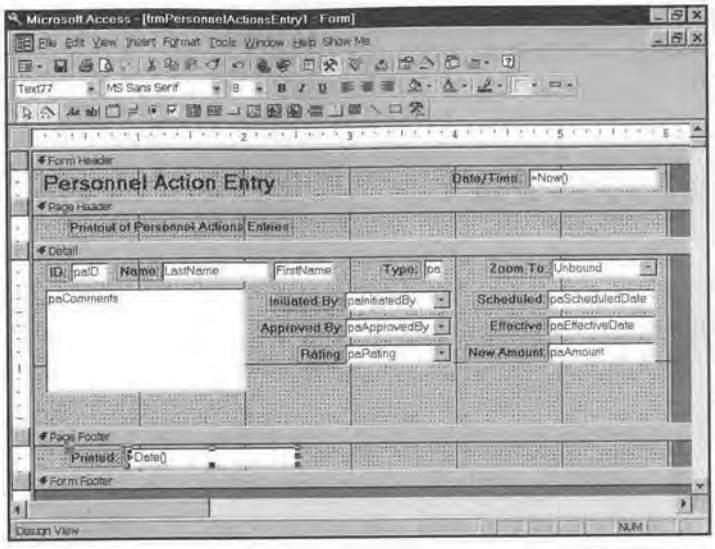


FIG. 13.71 The frmPersonnelActionsEntry form with Page Header and Page Footer sections added.

Oracle Ex. 1009, p. 531
Oracle v. DeBellis
IPR2024-00221

Chapter 14

Printing Basic Reports and Mailing Labels

The final product of most database applications is a report. In Access, a report is a special kind of continuous form that is designed specifically for printing. Access combines data in tables, queries, and even forms to produce a report that you can print and distribute to people who need or request it. A printed version of a form can serve as a report, which often is the case for reports designed for decision support, one of the topics of Chapter 20, "Adding Graphics to Forms and Reports." By printing a continuous form, you can create a report that displays some or all of the values of fields in a table or query.

This chapter describes how you create relatively simple reports, including multicolumn mailing labels, using the Report Wizards. The chapter also tells how you modify the design of the wizard's reports to suit your particular needs. The next chapter describes how you design a report from scratch without using the Report Wizards.

Differences and Similarities Between Forms and Reports

Most methods of creating transaction-processing forms, which you learned about in Chapter 11, "Using Action Queries," and Chapter 12, "Creating and Using Forms," also apply to reports. The following list details the principal differences between reports and forms:

- Reports are intended for printing only and, unlike forms, aren't designed for display in a window. When you view an 8 1/2×11-inch report in Print Preview, its content is not legible. In the zoomed (full-page) view, only a portion of the report is visible in the Print Preview window.
- You cannot change the value of the underlying data for a report with a control object from the toolbox as you can with forms. With reports, Access disregards user input from option buttons, check boxes, and the like. You can use these controls, however, to indicate the status of Yes/No option buttons, check boxes, and fields with values derived from multiple-choice lists.

Oracle Ex. 1009, p. 532
Oracle v. DeBellis
IPR2024-00221

- Reports do not provide a Datasheet view. Only Print Preview and Report Design views are available.
- You can create an unbound report that isn't linked to a source of data. Unbound reports are used as "containers" for individual subreports that use unrelated data sources.
- The Printer Setup dialog controls the minimum left, right, top, and bottom printing margins of reports. If a report is less than the printable page width, the report's design determines the right margin. You can increase the left margin over the default setting by positioning the print fields to the right of the display's left margin.
- In multicolumn reports, the number of columns, the column width, and the column spacing are controlled by settings in the Printer Setup dialog, not by controls that you add or properties that you set in Design mode.

Access reports share many characteristics of forms, including the following:

- *Report Wizards* can create the three basic kinds of reports: single-column, groups/totals, and mailing labels. You can modify as necessary the reports that the Report Wizard creates. The function of the Report Wizard is similar to that of the Form Wizard discussed in Chapter 12, "Creating and Using Forms," and Chapter 13, "Designing Custom Multitable Forms."
- *Sections* include report headers and footers that appear once at the beginning and at the end of the report, and page headers and footers that print at the top and bottom of each page. The report footer often is used to print grand totals. Report sections correspond to similarly named form sections.
- *Group sections* of reports, as a whole, comprise the equivalent of the Detail section of forms. Groups often are referred to as *bands*, and the process of grouping records is known as *banding*. You can add Group Headers that include a title for each group, and Group Footers to print group subtotals. You can place static (unbound) graphics in header and footer sections and bound graphics within group sections.
- *Controls* are added to reports from the Access toolbox and then moved and sized with their handles.
- *Subreports* can be incorporated into reports the same way you add subform controls within main forms.

Types of Access Reports

Reports created by Access fall into six basic types, also called *layouts*, that are detailed in the following list:

- *Single-column reports* list in one long column of text boxes the values of each field in each record of a table or query. A label indicates the name of a field, and a text box to the right of the label provides the values. Access 97's AutoReport feature creates a single-column report with a single click of the toolbar's AutoReport button. Single-column reports are seldom used because the format wastes paper.

Oracle Ex. 1009, p. 533
Oracle v. DeBellis
IPR2024-00221