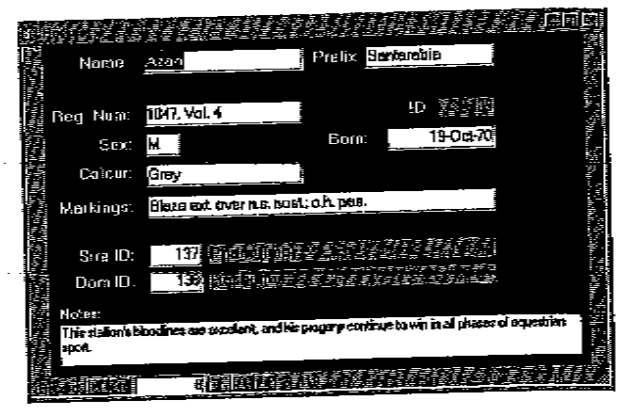

Figure 13.2.
The name of the sire and dam are displayed by binding text boxes to a `DLookup()` expression.

This technique is quite simple when the ID field is a number, but when the field is text, Access requires that it be enclosed in quotation marks. Say, for example, you need to look up the CategoryName field in tblCat where the CategoryID is `C`. The simplest method of providing the quotes is to use single quotes inside the double quotes:

```
=DLookup('CategoryName', "tblCat", 'Category = 'C'")
```

Again, to use this expression in a form, you need the current value of the CategoryID field in the form, rather than `C` for every record. Remember to include the closing single quote mark. This becomes

```
=DLookup("CategoryName', 'tblCat', "Category = '" & [Category] & "'")
```

Using this technique, it is possible to retrieve values from several fields at once. For example, you could look up both the "Writer" and "SongTitle" from tblSongs, with the following:

```
=DLookup("[Writer] & ', ' & [SongTitle]", 'tblSongs',
➥"SongID = '" & [SongID] & "'")
```

If the field you are looking up might contain an apostrophe, such as the name D'Sylva, avoid using the single quotes technique. Although it looks quite clumsy, you can generate a quotation mark within quotation marks by doubling them up. Using this technique, the last example becomes

```
=DLookup('[Writer] & '', '' & [SongTitle]', 'tblSongs',
➥"SongID = """ & [SongID] & """")
```

Access provides an Expression Builder that can help in building simple or complex expressions. In the Properties window, beside `ControlSource` where you are entering these expressions, is an ellipsis button. Clicking this button invokes the Expression Builder. The builder can reduce typing, prevent spelling errors, and provide clues on functions such as `DLookup()`. However, you still need to understand what you are aiming for in order to build a valid expression.

Essential Example

For this example, create a form to be used as a dialog box to request information. The form will request limiting dates to be used in a report. Both the report itself and the query it is based on will refer to these dates. The form will be used for other reports in the following chapters as well.

Create a new unbound form, and add two textboxes. In the Properties window, change their names to `StartDate` and `EndDate`. Because they are unbound, the `ControlSource` for both is blank. Set the `Format` property for both to Short Date, indicating the kind of contents you expect. Type appropriate labels such as `From:` and `To:`, and save the form as `frmWhichRpt`.

Next, create a query to select only those clients who joined you within the specified date range. Create the query based on table tblClients, and add the fields Company, Surname, FirstName, Phone, Fax, and Entered. Sort on Company. In the Criteria row for Entered, you need to refer to the dates in the form to limit the values in the query. Enter the following:

```
Between Forms![frmWhichRpt]![StartDate] and Forms![frmWhichRpt]![EndDate]
```

Now open the form, and enter some dates. Be sure to tab away from the second control after entering a date; the entry is not complete until the focus leaves the control. Test the query. If your object reference is incorrect, the query might complain, ask for an unexpected parameter, or simply contain no records. Save the query with the name `qryClientsByCompany`.

With the form still open, create a new report based on this query, using a simple tabular layout. The new report lacks meaning unless its header explains that it is a listing of clients who joined in a specific period. To do this, you again need to refer to the dates in the form.

With the report in Design view, add two text boxes to the Report Header. Give them names such as `StartDate` and `EndDate`, and alter their labels to `from` and `to`. Set the `ControlSource` for the first to

```
= Forms![frmWhichRpt]![StartDate]
```

and for the second to

```
= Forms![frmWhichRpt]![EndDate]
```

Now change the main label in the Report Header from qryClientsByCompany to `New Clients For The Period`, and arrange the textboxes side by side below this label, as shown in Figure 13.3. Save the report with a name such as `rptClientsNew`. The report is now ready for use (provided the form is open, of course). In following chapters, you will explore a method of opening the report using a button on the form and without needing to hard code the limiting dates into the query.

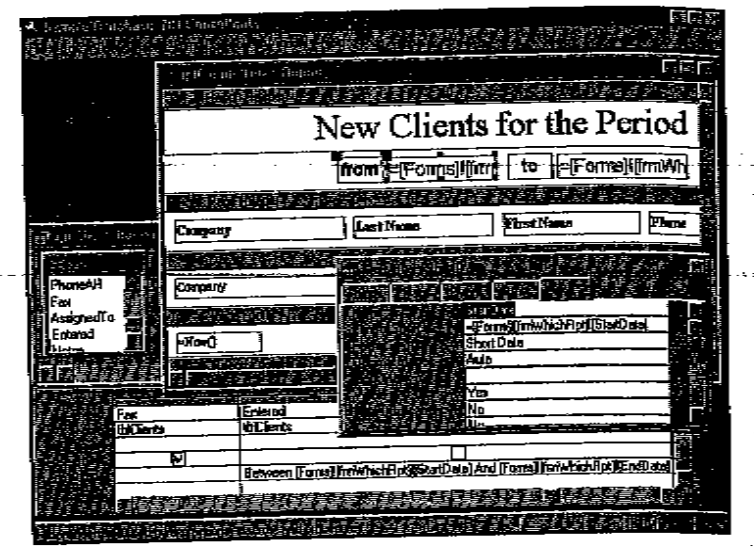

Figure 13.3.
Both the report and the query refer to controls on a form.

In the "Essential Example" of Chapter 8, "Report Basics," the subreport named rptReceiptSub contained a TotalReceiptAmt control. You needed to refer to the value in this control in the main report, to subtract the total received from the total due. Open the rptClientAccounts report in Design view, and add a text box to the bottom of the JobID footer. Name it `BalanceOwing`, and set its `ControlSource` to subtract the total in the subform from the total in the JobID footer:

```
= [TotalCharge] - [rptReceiptSub].Report![TotalReceiptAmt]
```

Essential Summary

What The forms and reports in Access are treated as objects. The controls that they contain are also objects. Each object has properties and events.

Why Knowing how to refer to an object opens the way for you to manipulate the object or access the data it contains. All the remaining material in this book assumes that you have learned how to refer to these objects.

How To give the full name of an object in Access, begin with the collection the object belongs to, and enclose names in square brackets. Use the bang operator (`!`) before user-defined names, and the dot operator (`.`) before system names.

The collection of open forms is called Forms. The collection of open reports is called Reports. To refer to a subform or subreport, you must include the `.Form` or `.Report` as part of the name.

Use the `DLookup()` function to refer to data in a table. `Dlookup()` requires three arguments: the field to look up, the table to look in, and a criteria statement identifying the record to find.

CHAPTER 14

Macros

Now that you know how to refer to objects, you can begin to manipulate them to your advantage. The macro provides a pleasantly simple interface that nonprogrammers can follow.

What Is a Macro?

What A macro is a sequence of actions programmed to occur on command. Unlike the other major products in the Microsoft Office suite, Access does not enable the user to create macros by recording keystrokes. Instead, you select each action from a drop-down list and supply details of the objects involved with the action.

The macros listed in the Database window can actually be several macros combined as a macro group. Grouping similar macros makes them easier to locate and maintain.

Why Use Macros?

Why Macros make repetitive tasks easier and faster. By assigning a macro to a button, a keystroke, or a menu item, you can improve the interface of