Chapter 6

Forms and Reports

Chapter Outline

What You Will Learn in This Chapter

- How do users interact with the database?
- What is the difference between a good form and a bad form?
- What common structures are used in forms?
- What are the main steps used to create forms?
- Can form usability be improved?
- What are the basic roles of reports?

A Developer's View

Ariel: Why the concerned look?

Miranda: Well, I finally figured out how to answer those hard questions. But I'm a little worried. Lots of times I got answers, but they were wrong. I have to be really careful with SQL.

Ariel: Oh, I'm sure you'll do fine. You're always careful about testing your work.

Miranda: I suppose it'll get easier.

Ariel: That's the spirit. Now, are you finally ready to start building the application?

Miranda: I sure am. I looked at some of the information about forms and reports. This is going to be easy.

Ariel: Really?

Miranda: Sure. And you know the best part? All the forms and reports are based on SQL. To get the initial forms and reports, all I have to do is build queries to get the data I want. There are even wizards that will help create the basic forms and reports.

Ariel: I always knew that someday people would call on spirits again.

Getting Started

Users should never see the underlying tables. Instead, data is entered through forms that match the business processes and managers explore data through reports and interactive forms. It is a challenge to build forms and reports that match users' needs, are easy to use, and are nice to look at. This is the stage where you get to start building the application.

Introduction

How do users interact with the database? The true power of databases lies in the ability to create business applications. Applications provide a way for users to do their jobs more efficiently—storing and retrieving data through forms and reports. You never want to give users direct access to tables. Imagine the chaos that would result if you ask a user to enter an OrderID, ItemID, price and quantity directly into a SaleItem table. Stop for a second and think about the applications you already use—particularly Web-based that run on browsers, cell phones, and tablets. Almost all of those applications rely on a database to store data. But users are never directly aware of that relationship. They simply focus on the application. Business data is often more complex than simple consumer applications, so the underlying database is more complex, but the principle of emphasizing the user interaction and hiding the actual database is still important.

Forms and reports are an important part of the database application. Designers use them to create an integrated application, making it easier for users to perform their tasks. Decision makers and clerical workers use forms and reports on a daily basis. Years ago forms were used primarily as input devices, and reports were used to display results. Today, forms and reports can be distributed electronically, display data interactively, and can display a variety of outputs. The Internet, and specifically the World Wide Web, is becoming an increasingly popular means of

Collect data
Display query results
Display analyses and computations
Startup for other forms and reports
Direct manipulation of objects
    Graphics
    Drag-and-drop

Figure 6.1

Basic uses of database forms. It is important to understand the use of a form, since forms designed for data collection will be different from those designed to analyze data.

distributing data as electronic forms and reports. The same design principles used for database forms also apply to the Web.

As summarized in Figure 6.1, forms are used to collect data, display results of queries, display analysis, and perform computations. They are also used as menus, or connectors, to other forms and reports. With the proper devices, forms can used for drag-and-drop or touch-based interactions. With this type of form, users interact visually with a model of the firm.

Reports are often printed on paper, but they are increasingly being created as Web pages for display on the screen. Reports are used to format the data and present results from complex analysis. Reports can be detailed and cover several pages, such as a detailed inventory report. Alternatively, reports can present summary data, incorporating graphs and totals. A common business example would be a weekly sales report comparing sales by division for the past few weeks. The report would generally be presented graphically and would occupy one page.

At this stage in the project, you need to create all of the forms and reports needed by the users. Fortunately, most systems have tools or wizards to help you create rough drafts of forms and reports. You will still spend considerable time improving the layout, formatting data, and establishing a consistent design scheme. But, it is relatively easy to come back later and modify the forms and reports. In many cases, you will want to take an interactive or prototyping approach where you obtain feedback from the user several times and improve the layout and design at each iteration.

This chapter looks at some basic issues in designing forms and reports in terms of human factor elements. It then discusses the most common types or layouts of forms used in business. Common report layouts are covered. The most common form layouts are: (1) Single-row showing one row on a page with detailed layout control, (1) Tabular showing multiple rows at a time in simple columns, (3) Main/subform that combines the first two types for parent-child relationships, and (4) menu forms with buttons and controls to support navigation and connect forms.

The chapter also looks at the basic elements of reports. Most report writers contain an innermost detail section to display rows of data, and then use headers and footers to display column headings and subtotals.

The details of creating forms and reports for a specific DBMS are covered in the associated workbooks, because these tools can be quite different across the various systems. Dealing with local desktop applications, mobile apps, and browser-based Web apps, quickly leads to even greater differences across the tools. The

chapter in this book focuses on the overall concepts which provide a powerful structure for understanding the overall goals. The workbooks show how to build actual forms and reports using specific tools.

Two-Minute Chapter

By itself, a database is a useful way to store data, but business users really need applications. Users should never deal directly with tables and queries. Instead, you create forms that are used to enter data in a layout that makes it easy for users to understand. Reports are used to display standard information, including subtotals and charts.

Think about how the process started with design questions. User forms and reports were collected to identify the data needed. These were subdivided into separate tables that can efficiently store and retrieve data. But split data is difficult for users to work with, so now you need to create electronic versions of the forms and reports that put everything back together. Users will enter data into the forms and it will be stored in the appropriate underlying tables.

Four main layouts are commonly used for forms: tabular, single row, main/subforms, and menu forms. Report writers typically define sections including page headers/footers, and grouping sections: header, detail, footer. Controls, such as text boxes, checkboxes, and drop-down lists, are placed on forms and reports to display or collect data for tables. Forms and controls use properties to set visual attributes. They also support events that can activate custom programming code. For instance, when data is entered into a text box, a program might check the data for proper format or compare it to other values. The overriding goal of forms and reports is to make it easier for users to perform their tasks.

Effective Design of Reports and Forms

What is the difference between a good form and a bad form? Designing forms and reports has much in common with creating art. You start with a blank page and choose layout, colors, and design elements. In fact, you might want to take a graphics design class or consult a graphics designer to help with the artistic side. However, you have the additional burden of making the forms and reports easy to use. You also have to ensure that they convey correct and useful information. The most important concept to remember when designing forms and reports is to understand that they are the primary contact with the users. Each form and report must be tailored to specific situations and business uses. For example, some forms will be used for **heads-down data entry**—where touch typists concentrate on entering data without looking at the screen. Other forms present exploratory analyses, and the decision maker will want to examine various scenarios. The features, layout, and capabilities of these two types of forms are radically different. If you choose the wrong design for the user, the form (or report) will be virtually useless. Keep in mind that these forms and reports will be used every day. Good design and attractive forms are critical.

The key to effective design is to determine the needs of the user. The catch is that users often do not know what they need (or want). In particular, they may not be aware of the capabilities and limitations of a modern DBMS. As a designer, you talk with the users to learn what they want to accomplish. Then you use your experience to provide features that make the form more useful. Just be careful to find the fine line between helping the user and trying to sell users an application they do not need.

Human Factors	Examples
User Control	Match user tasks Respond to user control and events User customization
Consistency	Layout, design, and colors Actions
Clarity	Organization Purpose Terminology
Aesthetics	Art to enhance Graphics Sound
Feedback	Methods Visual Text Audio Uses Acceptance of input Changes to data Completion of tasks Events / Activation
Forgiveness	Anticipation and correction of errors Confirmation on delete and updates Backup and recovery

Figure 6.2

Basic human factors design elements. All designs should be evaluated in terms of these basic features.

Researchers in human factors have developed several guidelines to help you design forms. To begin, all forms and reports within an application (or even within an organization) should be as consistent as possible. Keystrokes, commands, and icons should be used for the same purposes throughout the application. Color, layout, and structure should be coordinated so users can understand the data and context on any form or report. Basing applications on a set of common tasks reduces the time it takes for users to learn new applications. The increasing importance of Web-based applications simplifies design to some extent, because you now have a limited set of tools that are understood by almost all users. Research into **human factors design** has also led to several hints and guidelines that designers should follow when building forms and reports.

Human Factors Design

Figure 6.2 summarizes some human factors design elements that system designers should incorporate in their applications. With current operating systems, the primary factor is that the users—not the programmer and not the application—should always have control. For example, do not expect (or force) users to enter data in a particular sequence. Instead, set up the base forms and let users choose

the data entry order that is easiest for them. In this approach, the user's choices trigger various events. Your application responds to these events or triggers by performing calculations, retrieving or storing data, and offering new choices.

Also, whenever possible, provide options for user customization. Many users want to change display features such as color, typeface, or size. Similarly, users have their own preferences in terms of sorting results and the data to be included. Today, most development tools can automatically pick up the color scheme defined on the user's computer. The key is to use those color choices and not override them.

Both the layout (design and color) and the required actions should be consistent across an application. In terms of user actions, be careful to ensure consistency in basic features, such as whether the user must press the Enter key at the end of an input, which function key invokes the Help system, how the tab and arrow keys are used, and the role of each icon. These actions should be consistent across the entire application. This concept seems obvious, but it can be challenging to implement—particularly when many designers and programmers are creating the application. Two practices help ensure **consistency**: (1) At the start establish a design standard and basic templates for all designers to use, and (2) toward the end of the application development always go back and check for consistency.

Always strive for **clarity**. In many cases clarity means keeping the application simple and well organized. If the application has multiple forms and reports, organize them according to user tasks. It helps to have a clear purpose for an application and to make sure the design enhances that purpose. Use precise terminology, avoid jargon, and stick with terms that are used within the organization. If a company refers to its employees as "Associates," use that term, instead of "Employees."

Aesthetics also play an important role in the user interface. The goal is to use color and design (and sometimes sound) to enhance the forms and reports. Avoid the beginner's mistake of using different colors for every form or placing 10 different fonts on a page. Although design and art are highly subjective, bad designs are immediately obvious to others. If you have minimal experience in design aesthetics, consider taking a course or two in art or design. If nothing else, study work done by others to gain ideas, to train your artistic sensibility, and to stay abreast of current trends. Remember that graphics and art are important, and they provide an attractive and familiar environment for users.

Feedback is crucial to most human-computer interactions. People want to know that when they press a key, choose an option, or select an icon the computer recognized their action and is responding. Typical uses of feedback include accepting input, acknowledging changes of data, highlighting completion of a task, or signifying the start or completion of some event. Several options can be used to provide feedback. Visually, the cursor can be changed, text can be highlighted, a button can be "pushed in," or a box may change color. More direct forms of feedback, such as displaying messages on the screen, can be used in more complicated cases. Some systems use audio feedback, playing a musical theme or sound when the user selects a task or when the computer finishes an operation. If you decide to use audio feedback, be sure that you give users a choice—some people do not like "noisy" computers. On the other hand, do not be hasty to discard the use of audio feedback—it is particularly effective for people with low vision. Similarly, audio responses are useful when users need to focus their vision on an external task and cannot look at the computer screen.

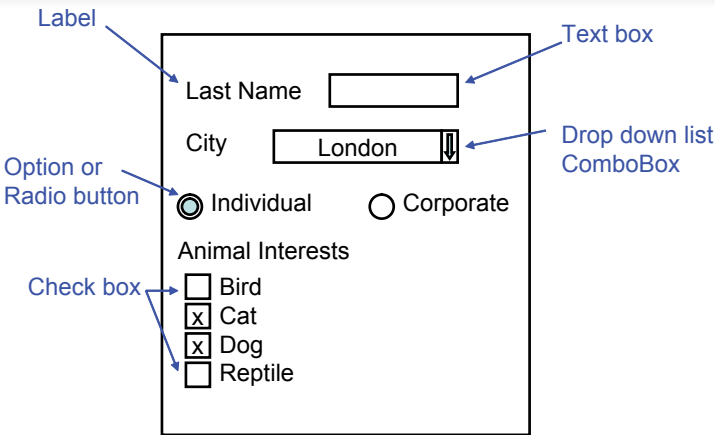


Figure 6.3

Common form controls. All DBMSs have these basic controls. Labels and text boxes are used the most. Round option buttons are used for mutually exclusive items.

Humans occasionally make mistakes or change their minds. As a designer, you need to understand these possibilities and provide for them within your application. In particular, your application should anticipate and provide for correction of errors. You should confirm deletions and major updates—giving users a chance to verify the changes, or even undo them. Finally, your overall application should include mechanisms for backup and recovery of data—both in case of natural disasters and in case of accidental deletions or loss of data.

Standard Form Controls

Most systems use an object-oriented approach to building forms. The standard form is drawn in a window or Web page that controls the size and provides scroll bars to display sections of the form that do not fit on the page. Some systems allow you to control these page features, but you should almost always leave them alone, so they continue to work the way the users expect. Today, most form-builder tools have adopted a relatively standard approach. Although the tools and options are different, the basic concepts are the same.

Figure 6.3 shows the most common form **controls** that exist. **Labels** are fixed text that is displayed on the form. Some systems attach the label to the data control, but you can generally set its font properties and move it around on the page. A **text box** displays data from a table cell. Generally, the displayed value can be edited or new data can be entered. The text box is a **data-bound** control, which means that you specify the table and column in the database. The form processor automatically retrieves the data from a row and displays the value in the text box. Any changes or new data entered is transferred back to the specified database column. You can set format properties to control the way the data is displayed. You can also specify an **input mask** to limit the type of data that can be entered by the user. For instance, you could specify that only numbers can be entered, or you can automatically format data to a fixed layout. Most systems have sample input masks to handle phone numbers and ZIP codes. However, be careful with input masks. Restricting the user to entering data a certain way might cause problems.

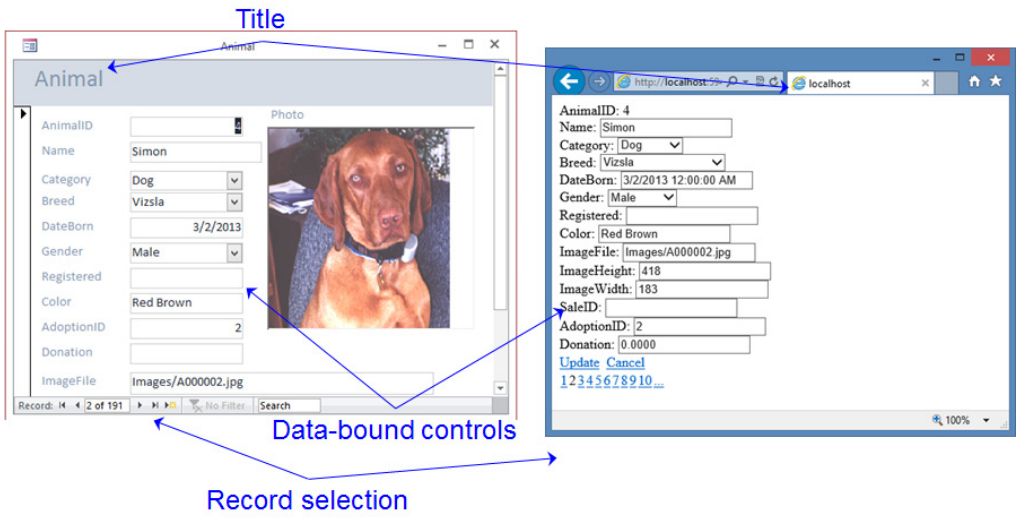


Figure 6.4

Form structure. A basic form has a title and some method of selecting new records. The data-bound controls retrieve data from the table and enable users to change and update the database.

Postal codes present a classic example. If your data is going to contain international addresses, you cannot use an input mask. Postal codes outside of America include letters and hyphens as well as digits. Similar problems arise with phone numbers. When you have to include country codes and other extensions, the standard American formatting will not work.

Figure 6.4 shows examples of a simple form to edit Animal data. A basic form has a title and some method to select the data record to be displayed. The data-bound controls then display the data for the specified record and authorized users can edit the data on the form and update the changes to the database. Most forms also have scroll bars when the form is too large to display on one screen window. There is also a mechanism to close the edit form or use a menu to switch to a different view

Commonly used controls include option buttons and check boxes. **Option buttons** are sometimes called radio buttons because on many systems they are drawn as filled circles that look like the knobs on old radios. **Check boxes** are almost always displayed as square boxes using some type of check mark to indicate selection. You need to be careful when choosing between option buttons or check boxes. Most systems are flexible and allow you to use either one for any type of problem. However, the design convention is that option buttons should be used for mutually exclusive events. Users know when they see the round buttons that they can select only one of the items. They also know that they can select multiple items if the square check boxes are used. Follow this convention to avoid confusing users. Option buttons and check boxes are usually data bound controls. Usually, they are bound to columns with yes/no or bit data types. On some systems, they are bound to integer data, where a zero represents the unchecked state.

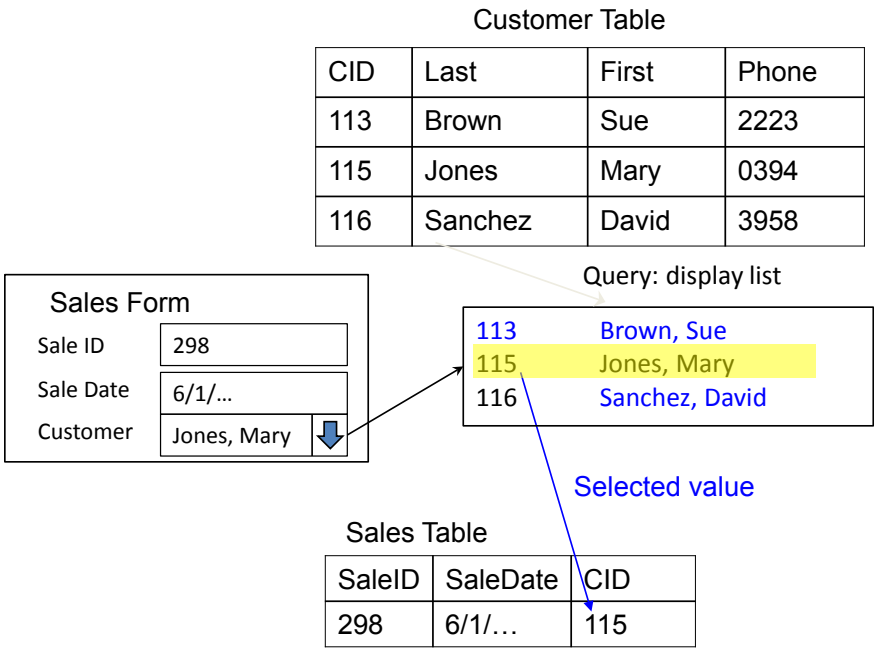


Figure 6.5

Drop down list for foreign key link. The values displayed in the list come from the Customer table. When the user selects one customer, the matching ID value is stored in the Sales table that underlies the Sales form.

The **drop down list** is sometimes called a combo box because it is a combination of a text box and a list box. It is one of the more complicated form controls, but it is extremely useful in relational databases. It retrieves lists of items from a table or query. When the user picks an item from the list, the chosen value (usually an ID) is entered into the main form field. This approach can be useful for lookup lists. For instance, users can pick a city from a preset list and store the CityID value as a link. In addition to saving user time, lookup lists ensure that data is always entered consistently—without abbreviations and typographical errors.

Drop down lists are also commonly used on referential links across tables. Figure 6.5 illustrates the process for a typical sales form. Recall that the Sales table contains the CustomerID column, and has a foreign key relationship to the Customer table. You cannot expect the sales clerks or customers to memorize ID values. So, you add a drop down list that displays a list of customers retrieved from the customer table. When the clerk chooses a customer, the matching ID value is placed into the underlying Sales table. To define the drop down list, you have to specify the table or query that will provide the list of data. You have to specify a display column and a key (ID) column. You often create a query to generate the display column, such as appending first and last names. You also have to data bind the drop down list to the CustomerID column in the target Sales table. This step ensures that the chosen value is stored in the proper column.

Because relational databases tend to have many foreign keys, you will find many opportunities to use drop down lists. However, in a Web environment with

relatively slow network links, drop down lists can cause problems. It could take a long time to transfer thousands of rows of data to display in the list. Even with a fast network, a drop down list with thousands of entries can be difficult to use. Users may find it difficult to scroll through thousands of entries in the list. You might need to implement some type of pop-up box to fill the list. Users can enter a simple search condition—such as the first letter of the name—to restrict the display list to a smaller set of entries that is faster to load and easier to scroll through.

Most systems have several other controls. For example, you can add **command buttons** and write code to open other forms or reports or perform some custom action. You can add background images, icons, or draw lines, rectangles, and circles. These items are generally decorations placed on the form and not bound to the database. Most systems have a control that stores and displays images or other binary objects in a data table. For example, you could store a photograph of each employee or product directly in the database.

User Interface—Events

Most systems treat forms using an **event** model. Each form (and control) can generate different events. You can write code to perform a custom action when some event is triggered. For example, opening and closing a form are basic events for every form. You can control how the form operates by creating actions that are taken when each event occurs. Many of the events involve the individual controls. One of the most common events to control is the click event when a button is clicked or submitted.

Although a form can have multiple controls (e.g., text boxes) on it, only one control at a time has the focus. The control that has the **focus** will receive keystrokes entered by the user. This control is often highlighted with an outline or a different color. The same concept applies when an application displays multiple forms on a screen. Only one form has the focus at a time. Within a form, users can usually use the Tab key to move to the next control in a sequence, which is known as the **tab order**. You must be sure to check the tab order on all forms so that it visually matches the layout of the form. In some cases, you might even want to handle the tab or exit event. When a user leaves a control, your code can perform additional calculations or assign default values.

By handling the events with custom code, you can make the form respond to user requests and to handle common tasks automatically. With dozens of controls and dozens of possible events per control, the challenge is to choose exactly which event is needed to perform a specific task.

User Interface—Accessibility Issues

One of the greatest strengths of the Windows and Web interfaces is its graphical orientation—which makes it easy for people to perform complex operations with a few moves of the mouse or selections on the screen. One of the drawbacks to this type of interface is that it is more difficult to make a system that is **accessible** to users facing some physical challenges. As a designer you can make your applications accessible to a wider base of users. To begin, your application should accept multiple sources of inputs. Do not rely on just a mouse or a pointer but also use the keyboard, and increasingly, the user's voice. Similarly, it is helpful if your application can provide multiple types of output. Increasingly, you should consider how to integrate sound and voice output. The user must also be able to set the color and size of the output.

- Language and characters
- Currency
- Time zones
- Time and date formats
- Number formats
- Country names and maps
- National ID numbers—privacy

Figure 6.6

Common internationalization issues. Applications that will be used by people in different nations need to adapt to national conventions as much as possible.

Microsoft guidelines provide some suggestions for making your applications accessible to more users. Detailed ideas and current developments can be found on its Web site. Human factors experience with other applications has generated some specific suggestions. For example, do not use red-green color combinations. Approximately 10 percent of the U.S. male population experiences some difficulty distinguishing between red and green. Try to pick high-contrast colors that most people can distinguish (e.g., black and white, yellow, blue, and red). When in doubt, ask people to test your color combinations. Better yet, let users select their own colors, or use the system color scheme that is configured on the user's computer.

Second, avoid requiring rapid user responses. Do not put time limits on input. Although it might be fun in a game, many users have slower data entry skills. Some designers include pop-up messages to check on user progress after a delay in data entry. These messages are usually pointless and can be annoying. With modern screen-saver security systems, users can set their own delay controls and messages.

Third, avoid controls that flash rapidly on the screen. They tend to annoy most users. Worse, certain flash rates have been known to trigger epileptic seizures in some people. An interesting situation arose in Japan at the end of 1997 when a sequence of flashing graphics on an animated television show (Pokemon) sent about 700 children to the hospital.

Fourth, as much as possible, enable users to customize their screens. Let them choose typefaces, font sizes, and screen colors. That way, users can adjust the screen to compensate for any vision problems they may have. And if you use sound, let people control the volume, even pitch, if possible. In many cases, the Windows environment provides much of this functionality, so the key point is to avoid overriding that functionality. Also, you should test your applications on various computers. Some video systems may distort your choice of colors or will be incapable of displaying your forms at the desired resolution.

User Interface—International Environment

Today, you must build your forms and reports with the understanding that people from around the world will use them. Figure 6.6 summarizes some of the common issues you have to consider when building forms and reports. Language is the most obvious, and it can be the most difficult. First, all of your text data needs to be stored using **Unicode** characters. With many DBMSs, you need to specify

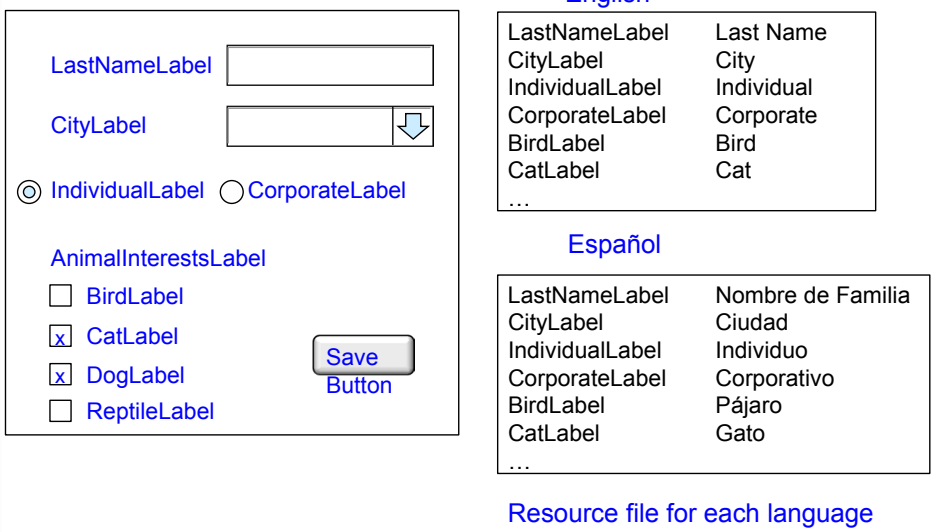


Figure 6.7

Multiple languages. All text on a form is dynamically stored as a resource variable. A resource file is created for each language and it holds the translated entry for every control.

nvarchar instead of varchar. With Unicode characters, names, addresses, and other data elements can be stored in any language.

Dealing with multiple languages on forms and reports is more challenging, but most development systems have methods to simplify the task. Your first thought might be to create every form multiple times: once for each desired language. Abandon that thought early. It would be extremely painful to alter a form if you had to change it for every different language. Instead, you specify the layout and design of the form once. Then you make all the text items (titles, labels, buttons, and so on) dynamic. The actual method depends on the tools you are using. As shown in Figure 6.7, generally, you give each item a name and declare it in a resource file. You then create a resource file for each supported language for each form or report. For example, the Sales form might have resource files for English, Spanish, French, German, and Japanese. The resource file contains simple pairs: the item name and the text translated to the desired language. Of course, you have to find someone to translate all of the items. When the form is generated for the user, the system chooses the appropriate language resource file and displays each text item in the user's chosen language. You create a form only once, but it can display the text in any language with a matching resource file. You can simply ship the resource file to a translator, and all of the work is stored in one location. You will probably want to include copies of the forms so the translators can see the context to create a better translation. You also have to test the form with each language, particularly non-Latin languages, to ensure the translated words still fit correctly on the form and do not overlap other controls. Once you have developed the dynamic form page and set up the resource files, it is relatively easy to sup-

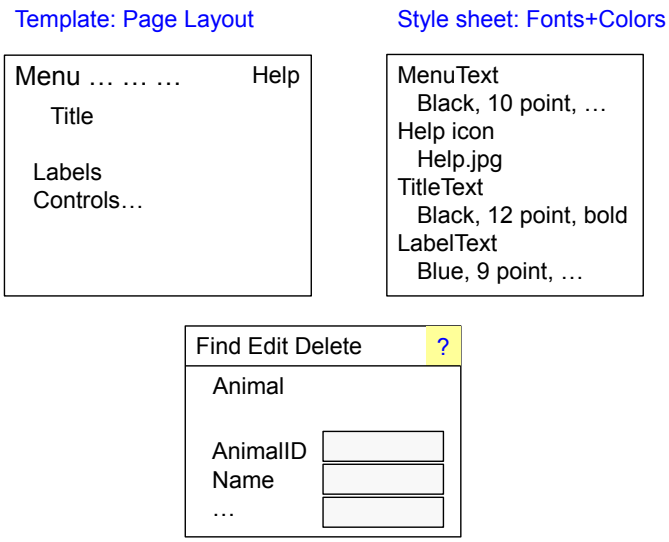


Figure 6.8

Style sheets and templates. Templates or master pages define the basic page layout and common controls. The style sheet sets colors, backgrounds, fonts, and so on for all items on the form. Ideally, style sheets are dynamic, so changes to a style are automatically picked up on every existing form.

port additional languages—simply have each resource file translated to the new language.

Data formats for currency, dates, and numbers also vary by region or nation. For instance, the U.S. uses commas to separate thousands, and a decimal point to separate fractional values in a number. Most European nations reverse these two, using dots to separate thousands. Dates can cause serious confusion if you use the shortened numeric format. In the United States, 1/5/2008 represents January 5, 2008; while the same short date in Europe indicates May 1, 2008. With Windows-based applications, you can query the regional settings on the user's computer to find the appropriate format for numbers and dates. However, it is probably safest to avoid the short-date format and use a format such as 05-Jan-2008, which is clearly understood by almost everyone. If you do retrieve local settings from the user's computer, be extremely careful about currency values. Picking up a local currency symbol does not convert the values. Serious problems could result if an American enters $100.00 into the database, but the system displays it as £100.00 to someone in England.

The issue of country names can become a problem, or even an international incident, if you are using a geographic information system or mapping program. Several areas around the globe are politically disputed or face different claims (sometimes even calling an area "disputed" might anger the participants). You need to have the correct map for each region. Country names also vary by language, but you can generally pick one language and use the names defined by the International Organization for Standardization (ISO) or the United Nations.

Finally, be careful to abide by all national laws with respect to security and access to the data. For example, many European nations have strong privacy statutes—particularly with respect to customer data and national ID numbers. In some cases, it may be illegal to transfer data collected in Europe to the United States.

Style Sheets and Templates

Consistency quickly becomes an issue for large projects—particularly when multiple developers are working on the forms and reports. It is important that all of the forms and reports use the same color schemes and as much as possible, place common items in the same location. For instance, links for help, search, or close should be the same across all forms and reports. Clearly, designers need to specify the layout details before forms and reports are built. But even then, it is difficult for developers to remember the exact specifications. Consequently, two tools are useful to help with consistency: style sheets and templates. Some systems combine the two topics, some give them different names (such as master pages), and others do not support them at all.

Figure 6.8 shows how style sheets and templates influence form development. Templates or master pages define the page layout and hold the controls that are common to every form. In the example, the template defines the location for the main menu, a Help icon, and the page title. The style sheet defines the colors, fonts, icons and so on that set the image for the page. Ideally, every item appearing on a form will be named and defined in the style sheet. Developers simply select the appropriate style when a page is created.

With some systems the style sheet is dynamically attached to each page. For example, Web browsers use the page definition and the style sheet to determine how to display a page. If the style sheet is changed, the Web browser automatically uses the latest values when displaying a form or report. Consequently, designers can alter the page style and colors at any time—even after the application has been completed and deployed.

Form Layout

What common structures are used in forms? Individual forms are your primary means of communicating with people who use your application. Forms are used to collect data, display results, and organize the overall system. From a database perspective, your application is built from several standard types of forms. As you begin working with these basic layouts, keep in mind that you can create complex forms that use features from several different form types. However, you should understand the layout and uses of each individual form type first.

As summarized in Figure 6.9, you will be working with four basic types of forms: (1) **tabular forms**, which display data in rows and columns, (2) **single-row forms**, which show data for one row at a time and in which the designer can arrange the values in any format on the screen; (3) **subform forms**, which display data from two tables that have a one-to-many relationship; and (4) **startup forms**, or menus, which direct the user to other forms and reports in the application.

You can think of a subform as a combination of a single row form and a tabular form. The Sales form is a classic example. The Sales table forms the foundation of the main form, and the SaleItems table data is handled in the subform. Because each sale can contain many items being sold, you need the subform to handle the repeating section.

Form Type	Common uses
Tabular Multiple rows of data.	Lookup lists or tables with a limited number of columns when it is useful to see several rows of data at a time.
Single row One row of data at a time.	The most common type. Provides complete control over page layout and space for many columns and links.
Subforms Combine row and details.	One-to-many relationships. Repeating section shows data related to main form. The items section of a sales form is a typical example.
Switchboard or startup Customized with buttons.	A designed form that is used as the main menu or switchboard to the other forms and reports.

Figure 6.9

Form layout. Once you understand the single row and tabular (repeating) forms, you can create more complex forms by creating subforms. The startup form is similar to a menu that links the forms and reports together.

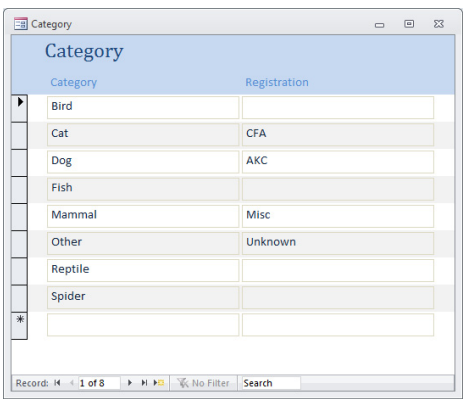
Tabular Forms

As shown in Figure 6.10, one of the simplest forms is the tabular form, which displays the columns and rows from a table or query. Some systems provide variations on the tabular form, such as the datasheet in Microsoft Access. The tabular form is used as the main form with a limited number of columns, and when users need to see multiple rows at the same time. It is particularly useful for administrative forms, such as editing values in a lookup table.

The primary feature of the form is that it displays multiple rows of data for editing. Consequently, users can see and compare data across several items. It is useful for short lists of data. It can cause problems when the table to be edited is

Figure 6.10

Sample tabular form. You define the controls for one row, and the DBMS displays data for all of the rows in the query. Scroll bars enable the user to see more rows (or columns of data).



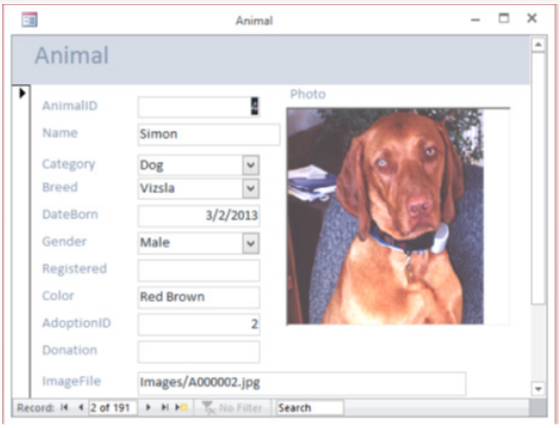


Figure 6.11

A simple single-row form. This form displays data for one row at a time. You have substantial control over layout through color, graphics, and command buttons. The navigation buttons on the bottom enable the user to display different rows.

too large—both in terms of number of rows and too many columns. If the user has to scroll repeatedly to find a specific entry, a tabular form will not work very well.

Tabular forms are commonly used as subforms, where they collect and display a limited list of data that is related to the main form. For example, an order form often contains a subform tied to the OrderItem table to display the list of items being purchased on the currently displayed order.

Single-Row or Columnar-Forms

A single-row form displays data for one row at a time. The goal is to display every column on one screen. Its greatest feature is that the designer can display the data at any location on the form. It is useful for designing a form that looks like a traditional paper form. The designer can also use color, graphics, and command buttons to make the form easier to use. As illustrated in Figure 6.11, this form design requires navigation controls that enable the user to scroll backward and forward through the rows of data. Common navigation controls also include buttons to go to the first and last rows and to go to a particular row of data.

In general, you will want to include a Find command that enables the user to locate a particular row of data—based on the values in some row. For example, a form displaying customer data should have a search option based on customer name. Similarly, the user will often want to sort the rows in different orders.

The single-row form is generally the most-used form layout. With careful design you can use it to display substantial amounts of data. By including subforms, you can highlight relationships among various pieces of data and make it easy for users to enter data. You can also include charts to help users make decisions.

Subform Forms

A subform is usually a tabular form embedded on the main form. A subform generally shows a one-to-many relationship. In the example in Figure 6.12, a sale could include many adopted animals, so you need a subform to display this re-

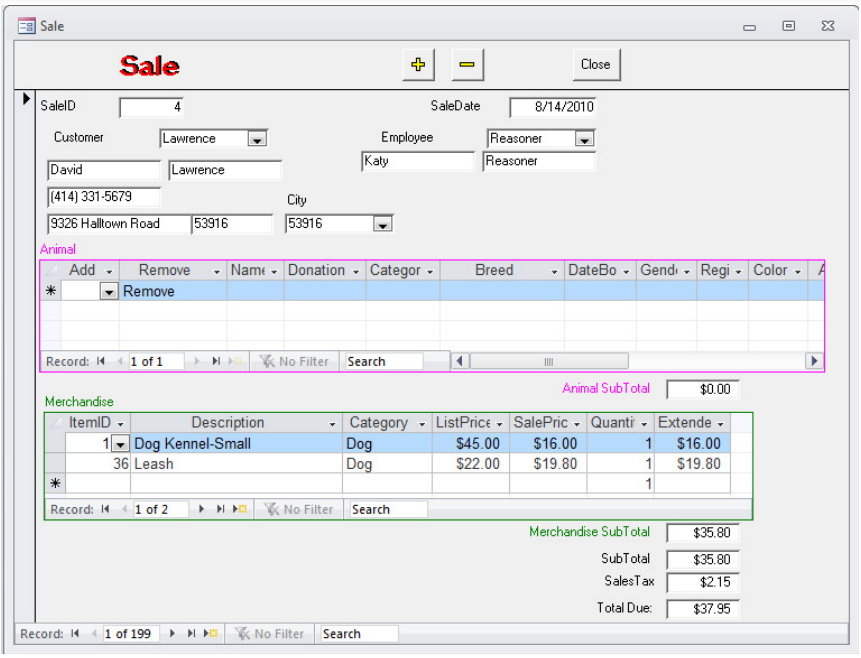


Figure 6.12

Subform example. The main form is based on the Sale table, which has a one-to-many relationship with the SaleAnimal table used on the subform. The subform contents are linked via the SaleID. The two subforms in this example are independent.

peating list. The main form must be a single-row form, and the subform should be a tabular view. The Sale example for the Pet Store is more complex than for other companies because it uses two subforms. Animals are treated as separate objects from merchandise, so each is recorded in a separate subform.

If you look at the underlying tables, you will see that SaleID links the main form (based on Sale) and merchandise subform (based on SaleItem) to each other. You will rarely display the linking column on the subform. In general, doing so would be pointless, since the linking column would always display the same value as the related column on the main form. Think about what that means for a minute. The Sale table has a SaleID that is generated by the DBMS when a new sale is created. The SaleItem table also has a SaleID column, and every animal sold must contain the same SaleID value from the main form. Yet it would be painful for the clerk to reenter the SaleID on the subform for each animal that is sold. By using the subform and specifying the SaleID as the link (Master and Child property), the DBMS automatically enters the main SaleID into the table for the subform.

Most database systems enable you to create forms that have multiple subforms. The subforms can be either independent—as separate boxes on the main form—or nested—where each subform lies inside another. In most cases, you will want the parent forms to be single-row forms. However, some systems support tabular lists for both the main and the subforms. In most applications, users would find this

Figure 6.13

Sample startup form. The buttons match the user's tasks.

approach disconcerting: They would first have to select a single row in the parent form and then deal with the matching list in the subform.

Startup Forms

Startup or menu forms provide the overall structure to an application. They are straightforward to create, although you may want the assistance of a graphics designer. The startup form often contains images, and the design reflects the style of the company. Startup forms are not required in all applications—in many cases the same functions can be provided through menus or the newer ribbon bars.

You begin with a blank form and remove any scroll bars and navigation controls. Pictures can be inserted as background images or as individual controls that can be used as buttons to open another form. As shown in Figure 6.13, command buttons or links are the most important feature of the startup form. When the user selects a button, a corresponding form or report is opened. The main startup form will be used quite often, so you should pay careful attention to its design. In Web-based applications, the buttons are usually replaced with hyperlinks to the desired form or report.

The key to a successful application begins with the startup form—not just its design but also its content. The startup form and links on other forms create the application flow which should match the user's tasks. One approach is to first identify the user and then provide a selection of buttons that matches his or her tasks. Consider a simple example. A manager needs to print a daily sales report of best-selling items. Every week the DBMS must print out a list of total sales by employee. The firm also sends letters to the best customers every month offering them additional discounts. A secretary will be in charge of printing these reports, so you create a simple menu that lists each report. The secretary chooses the desired report from the list. Some reports might ask questions, such as which week to use. The secretary enters the answers, and the report is printed.

The first step in creating an application is to think about the people who will use it. How do they perform their jobs? How do the database inputs and reports fit into their job? In object-oriented design terminology, each situation is called a **use case**. The goal is to devise a menu system that reflects the way they work.

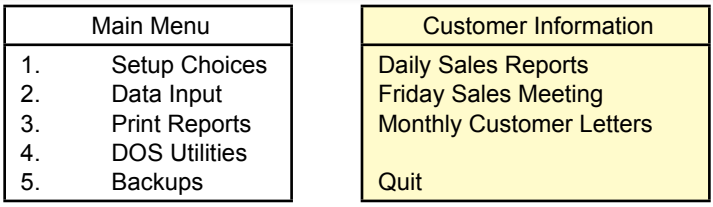


Figure 6.14

Designing menus for users. Which menu is easier for a secretary to understand? When designing applications, you should organize the application to match the processes users perform.

Two examples of a first menu are shown in Figure 6.14. Which menu is easier for a clerk to understand? Answer: The one that best relates to the job. Once you understand the basic tasks, write down a set of related menus. Some menu options call up other menus, some print reports, and others activate the input screens you created.

Creating Forms

What are the main steps used to create forms? The first step in creating a form is to be sure that you understand its purpose and how it will be used. Its usage dictates the specific data that needs to be displayed. Once you know the data needed, you can identify the database tables that hold that data. One important point to remember: A form should only attempt to update data to one table at a time. For example, a common sales form might display data about the Sale, the Customer, the SaleItems selected, and perhaps detailed data about the individual Products. Through the design process, this data needs to be stored in four related tables, so how can you create a form that updates only one table?

The answer to this question actually determines the characteristics of many database systems. The reason you can put only one table on a form is that multiple tables make it difficult for the form to understand exactly what the user is attempting to do. For instance, if the main Sales form contained all columns from the Sales table as well as the Customer table, what does it mean to add a new row? Should that row be added to the Sales table or the Customer table?

The process is complicated when you want to display data from multiple tables. For instance, you probably want to display a customer's name and phone number on a sale form. Depending on the DBMS you are using, you could have several options. You might be able to display the customer data on the sale form without needing to edit it. In this case, you could add a button or hyperlink to open the customer form with the corresponding data available for editing. Alternatively, you might be able to create sections within a form, where each section can hold data for a new, linked table. One of these sections could be a subform that contains repeating rows of data linked to the main form. Depending on the database system, you might be able to use an updateable query to display data from multiple tables.

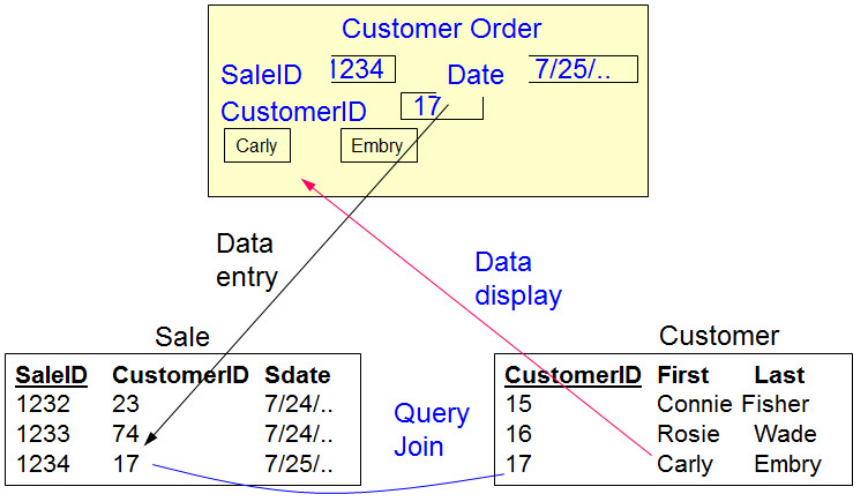


Figure 6.15

Basing the Order form on a query. The query contains all the columns from the Sale table and some columns from the Customer table. The query must never include the CustomerID column from the Customer table, which is the column used to join the two tables.

Updateable Queries

The issue of multiple tables on a form is related to the problem of updateable queries. If a system can support queries that use multiple tables as updateable, then it is possible to put carefully selected columns from multiple tables on a single form. Figure 6.15 shows a common Sales Order main form. The Sale table holds the CustomerID as a foreign key. To record a sale, a salesperson simply needs to enter the ID number of the appropriate customer. But it is not easy to remember numbers, and it would be nice to display the matching customer data on the main form to verify the name and address.

Technically, a query could be built that includes all of the columns from the Sale Order table along with some of the descriptive columns from the Customer table. To remain updateable, the query must not include the primary key (CustomerID) from the Customer table. Figure 6.16 shows the basic query. One potential

Figure 6.16

Updateable query for Sales form. New rows will be entered into the Sale table, but some systems will also support updates to the three columns from the Customer table. Note that you should never include the Customer.CustomerID column in the SELECT statement.

```
SELECT Sale.SaleID, Sale.SaleDate, Sale.CustomerID,
  Customer.LastName, Customer.FirstName, Customer.Phone
FROM Sale
INNER JOIN Customer
ON Sale.CustomerID=Customer.CustomerID;
```

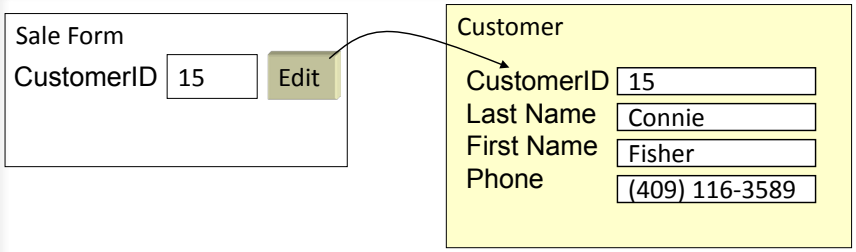


Figure 6.17

Linked forms. The Edit button opens the Customer form and loads the data that matches the current value of CustomerID on the Sale Form.

drawback to this approach is that whenever a new CustomerID is entered into a sale, the form must make a trip to the database to look up the matching customer data. Consequently, some forms systems do not support this approach. In particular, it can cause problems on Web-based forms because of the delay in sending data to the server and waiting for a reply.

Linked Forms

Another approach to the problem is to use linked forms. As shown in Figure 6.17, these forms might be displayed separately, or they might be separate sections displayed on a single form. This latter approach is used for parent/child forms, where the child form contains data from a second table that has a one-to-many link to the parent form. Conceptually, displaying a related table on one form versus in a separate window is equivalent. The main difference lies in the amount of screen space needed. If you put several sections on one form, users will need large screens to see all of the data. By using multiple windows, the user will not be able to see all of the data at one time but can switch between the windows to see and edit the data.

Linked forms work by using a query to match the data displayed in the secondary form to a key value from the original form. For instance, from the main Sale Order form, the CustomerID can be used to display the matching customer data in a linked form. Similarly, a SaleItem subform can display the rows that match the SaleID from the main form. Each related portion of the data can be displayed in a separate form or region. Ideally, the forms system will have a method to automatically perform the linking and retrieve the matching data; otherwise, you will have to write customized code to modify the underlying query and refresh the data as needed.

Properties and Controls

Most software packages are built using an object-oriented approach. With an OO design, each object has properties that describe it and methods or functions that it can perform. Objects are also closely tied to an event-driven system, where user actions and changes can trigger various events. Most database forms follow this methodology and contain dozens of objects to create and enhance forms. Your job is to assign properties and write short programs to respond to various events to make the application easier for users.

Category	Sample Properties
Data	Base table / query Filters Sort
Integrity	Edits Additions, deletions Locks
Format	Caption Scroll bars Record selectors Navigation buttons Size and centering Background/pictures Colors Tab order
Other	Pop-up menus Menu bar Help

Figure 6.18

Basic properties for forms. At a minimum, set the data source and basic format properties. Additional properties ensure consistency, protect data, and make the form easier to use.

As highlighted by the abbreviated list of properties in Figure 6.18, form and control properties can be grouped into primary categories. The first category (data) relates to the source of the data, where you set the base table or query. You can set filters to display only the data rows that meet a specific condition. You can also specify the sort order and a WHERE clause directly in the underlying query, which normally would be a more efficient approach. This step essentially binds the control element to the database.

A second set of properties refers to data integrity to help you control the type of editing and changes allowed on the form. For example, you might set the properties for individual users so that some users cannot add or delete data using a particular form. However, keep in mind that it is generally safer to set these conditions in SQL so that they apply throughout the database, and not just on one form. For example, sales clerks should probably be prevented from adding new suppliers.

A third level of properties controls the display of the form. Everything from the caption to scroll bars, form size, and background are set by display properties. Again, remember that consistency is a virtue. Before beginning a project, choose a design template and standards; then set all form and control properties to meet that standard.

Controls on Forms

The standard form controls are supported by every forms-development tool. Figure 6.19 shows some of the controls available in Visual Studio (2010). Many systems have wizards that quickly create standard labels and text box controls for

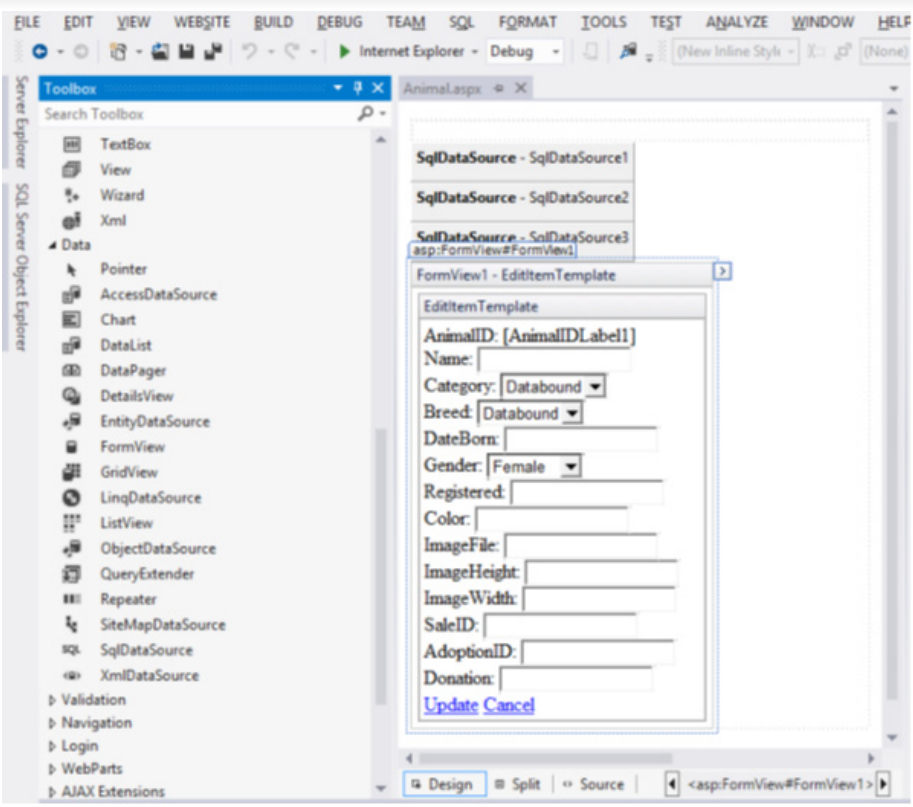


Figure 6.19

Some controls in Visual Studio. The controls are grouped into categories to make them slightly easier to find. You drag a control onto the form and assign its properties. You can buy additional controls or write your own.

desired columns. Each control needs to be given a name and bound to the appropriate data column. Be careful with the name; once you have set it, it is difficult to change. You need to pick a meaningful name when you first create the control. If you try to change it later, you will have to find every control or program that refers to that control—which can be a time-consuming and error-prone task. The name is used by other controls and program code (much like a variable name) to retrieve and store data on the form.

Some people name controls based on the type of control. For example, the name of a label control would end with the word Label, such as AddressLabel. Alternatively, some developers precede the name of the control with its type, such as lblAddress. The difference affects the sort order of the controls. If you end with the "Label" notation, control names sorted alphabetically will group by the data name (AddressLabel, AddressTextBox). If you start with the "lbl" notation, all of the labels will be listed together (lblAddress, lblPhone). You need to decide at the start how you want to locate and remember the control names.

Once the base controls are set, you will have to set formatting, and rearrange the controls on the page. You will also have to pay attention to the layout of controls on the page. Generally, you can drag the controls and labels to new positions, but be sure to use the alignment tools to match the edges.

Figure 6.20

Image bound to a data column. Employee photo is scanned and stored in the database column.

Graphics Features

Occasionally you will want to add graphics features to your form. There are controls to add pictures, as well as simple lines and boxes. Lines and boxes are often used to create a three-dimensional effect for other controls by adding shading or highlighting.

To display an image from a table as shown in Figure 6.20, you must first define an Object or image column within that table. Then the bound object frame is used like a text box to position and display the image on the form.

To use an image or texture as a background, first use a graphics package to make sure the image is light enough to not interfere with the readability of the other boxes. Then set the Picture property of the form to the name of the image file. In most cases, you want to embed the image on the form so the picture is included directly with the database.

For Web-based applications, it is easier to store images as files on the server—instead of trying to store them within the database itself. Web pages ultimately use HTML controls to display pages to the user. Web servers and browsers know how to retrieve and display images from files. Hence, it is straightforward to store the image filename in the database and embed that name in the HTML so it is retrieved automatically. If you store the actual image within an object control in the table, you need a special program to convert the page HTML request into a database query to retrieve the image and deliver it to the browser. Besides the extra work, storing images in a database table quickly eats up storage space. And if you want to use the free versions of the commercial DBMSs, you need to hold down the size of the database.

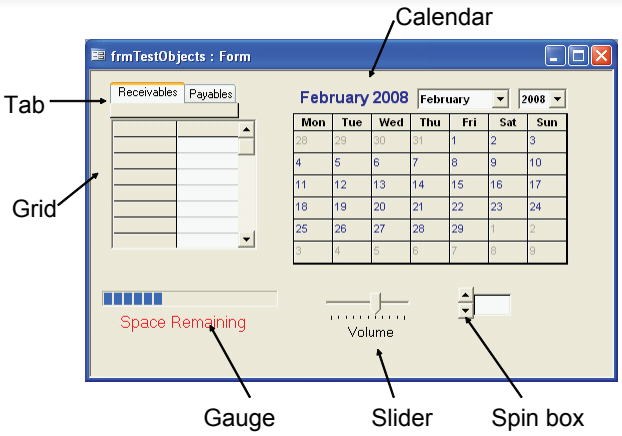


Figure 6.21

Additional controls. Thousands of controls are available to improve the user interface or perform specialized tasks. Common controls include tabs, grid, calendar, gauges, sliders, and the spin box. You can also create custom controls.

Complex Controls

Additional control objects can be created using a variety of computer languages or purchased from commercial vendors.

A few additional controls are shown in Figure 6.21. The Tab and Calendar controls are particularly useful in business applications. There is also a Grid control that enables you to display data in a spreadsheet layout. These types of controls are not as easy to use as the standard bound controls. The developer has to write short programs to load data into the control and to respond to the control's events. Note that the Calendar control was removed from Microsoft Office 2010. The date picker is available for text boxes that are formatted as dates. If you want calendars for other purposes, you will have to find or build a new one—or switch to Visual Studio.

Charts

Database applications used for making decisions often contain charts or graphs. Charts are another type of control that can be placed on a form (or report). The first step in creating a useful chart is to discuss with the user exactly what type of data and what type of chart will be needed. Then you usually build a new SQL query that will collect the data to be displayed on the chart. The chart control places the chart on the form and specifies the individual attributes (like type of chart, axis scale, and colors).

Two basic types of charts are used on database forms and reports: (1) graphs that show detail from the currently displayed row and (2) graphs that display summary data across all (or several) of the rows. The difference between the two approaches lies in the level of data displayed. Detail graphs change with each row of data displayed. Summary graphs are usually generated from totals or averages.

Figure 6.22 illustrates the two types of graphs as they might be used in the Pet Store database. Each chart shows the amount of money spent on animals versus the amount spent on merchandise. However, the top set of charts shows the split

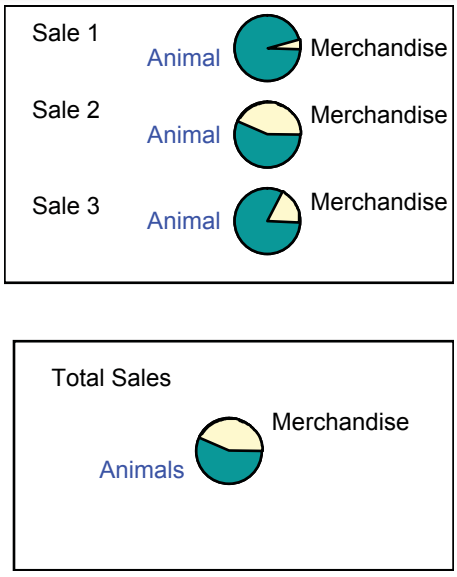


Figure 6.22

Charts on forms or reports. The top charts show the split in sales for each individual sale and will change with each row of data. The bottom graph shows the sales split for all sales and is not bound to an individual row.

for each individual sale, so the graphs vary with each row in the Sale query. The bottom chart shows the overall total for the store—even if it is placed on a Sales form that shows each row of data, it will not change (except over time). To create the two types of charts, the main difference is in the query. The query for the detail charts contains a column (SaleID) that is linked to the row of data being displayed on the form (based on its SaledID). The summary graph computes the totals across all of the sales and is not linked to any particular sale.

Multiple Forms

As you can guess, an application will quickly spawn many different forms. Of course, the forms should be linked to each other so users can quickly move between the forms by clicking a button, data value, or image. Startup menu forms play an important role in tying forms together. However, you can also connect forms directly. The most common example is the use of subforms placed on a main form. In this situation the forms are linked by setting the Master and Child properties of the subform. Then the database system keeps the data synchronized so that when the user selects a new row in the main form, the matching rows in the subform are located and displayed automatically.

When the forms contain related data, another approach is to build a link that opens the second form based on the ID value in the first form. For example, if the Order form contains customer data, when the user clicks an Edit button (or double-clicks on the customer name), the application should open the Customer form. The Customer form should display the data that corresponds to the customer on the Order form. The technique for linking forms varies by DBMS. The accompanying workbooks provide the syntax and examples needed for each DBMS. In the

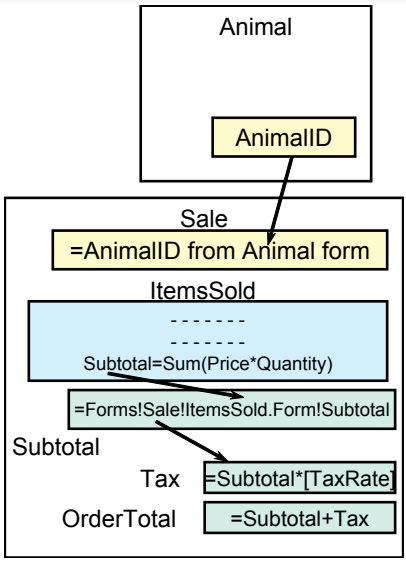


Figure 6.23

Copying data from a different form. The default AnimalID is copied into the Sale form from the Animal table. Likewise, the subtotal is first computed on the subform and then copied to the main form.

Sale form example, the link criteria specifies that the CustomerID in the Customer form must match the CustomerID from the Sales form.

In most cases, the user would close the Customer form and return to the main Sale form. But what if users commonly keep both forms open at the same time? Then they would expect the data between the two forms to be synchronized so that when a new row is displayed on the Sale form, the matching data would be displayed on the Customer form. You might have to write a couple of lines of code to enable this synchronization to work. Essentially, whenever the Sale is changed, your code grabs the new CustomerID, passes it to the Customer form, requeries the database, and redisplays the form with the matching data.

A third, related situation is shown in Figure 6.23. Perhaps while looking at animal data, the customer decides to adopt that animal. An adoption button on the Animal form could quickly bring up the Sale form. It would be convenient if the button then copied the AnimalID into the appropriate space on the Sale form.

Again, you need to write a couple of lines of code to insert the AnimalID into the appropriate control on the Sale form. In some situations, you might want the Sales form to refer back to the ID value on the Animal table.

Business applications commonly need to compute subtotals from subforms. As shown in Figure 6.23, some systems treat subforms as entirely separate forms, so you must first do the subtotal calculation on the subform, and then copy its value back to the main form.

Direct Manipulation of Graphical Objects

Can form usability be improved? In the last few years, the user interface to applications has been changing. The heavy use of graphics has led to an emphasis on **direct manipulation of objects**. Instead of typing in commands, the user can

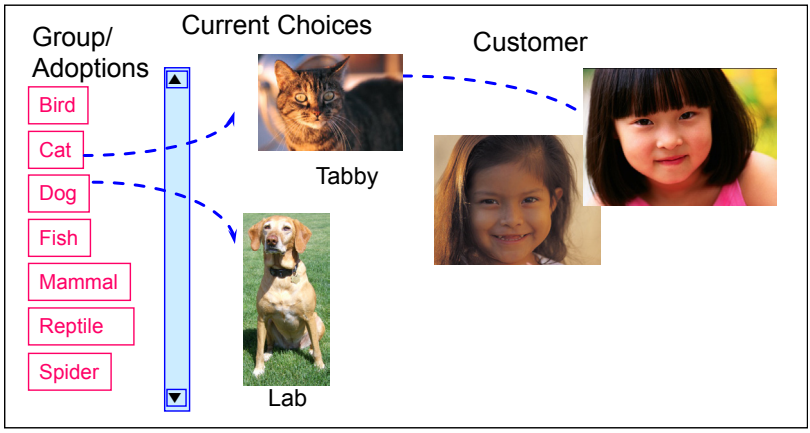


Figure 6.24

Direct manipulation of graphical objects at the Pet Store. Instead of entering an AnimalID into a box, you drag the picture of the animal to the customer to indicate a sale. Double-clicking on an item brings up more detail or related graphics screens.

drag an item from one location on the screen to another to indicate a change. Most people have seen this approach used with basic operating system commands. For example, in the days of command-line operating systems you had to type a command such as: COPY MYFILE.DOC A:MYFILE.DOC to copy a file. Today you click on the file icon and drag it to a disk drive icon.

More recently, direct manipulation has been expanded with the adoption of multi-touch screens. These devices provide options beyond the simple mouse-move commands. To date, most of the innovations have been in relatively simple actions such as the "pinch" move to contract or expand the size of an object. However, some opportunities exist to use this hardware for handling data. For instance, drilling down to see more detailed data or following referential links could be handled with a gesture. For ideas, see portions of the movie *Minority Report*. Ultimately, these gestures need to be standardized so they can be used on multiple devices and applications. Hopefully, standards groups will work on these concepts before some lawyer attempts to patent a gesture.

Sally's Pet Store Example

A graphical approach can make your applications easier to use. However, it requires changing the way you think about applications, and a good dose of creativity. Consider the Pet Store example. The basic forms designed earlier in this chapter were easy to create, and they will perform adequately. However, you could change the entire approach to the application.

Figure 6.24 shows a partial screen for the Pet Store example. Compare this form to the traditional data entry form shown in Figure 6.12. The traditional approach requires users to enter text into a box and perhaps select an item from a drop-down list. With the graphical approach the user sees photos of the individual animals or merchandise and drags them to the customer to indicate a sale. Double-clicking on an item provides more pictures or additional details. A similar approach would be used to special-order items, using **drag-and-drop** techniques

to create search conditions such as category and price to narrow down the list of products.

Note that you cannot entirely eliminate data entry. At some point, you need to collect basic data on customers (name, address, phone number, etc.). This data could be entered on a traditional form that is activated when the clerk double-clicks the customer icon or photo. That is, the form in Figure 6.24 replaces the traditional sales form but does not replace the basic customer form. Of course, once the customer data is on file, it can be dragged back to this main form whenever the customer returns.

The Internet

The emphasis on graphics and a direct manipulation of objects can be particularly valuable for forms used with the Internet. For starters, most of the users will have little experience with databases and only limited knowledge of your company. Creating a graphical model of the company and its processes achieves two important objectives: (1) It makes the site easier to use because it matches the physical purchase methods users already know, and (2) it limits the actions of the users to those that you have defined.

Many Web forms use an approach similar to the direct objects, except that drag-and-drop is rarely implemented because of browser limitations. Most of the time, users browse or search a catalog of items. Selected items are added to a shopping cart. At checkout, the items are shown in a listing similar to a traditional sales form—but users generally must return to the catalog listing to edit the list to change or select new items. Perhaps as HTML5 gains popularity and developer experience, graphical ideas will expand into Web forms and applications.

One of the goals of the graphical approach is to hide the use of the database. Yes, all basic product information, figures, and sales data are stored in the database. The database system provides search capabilities and stores user selections. It also provides reports and data analysis for managers. However, users never need to know about the database itself. Users simply see an image of a store and its products. They manipulate the objects to learn more or to place orders. One difficulty of the Internet is that you are limited by the capabilities of the user's browser. Browsers handle most simple data entry controls, but rarely have the ability to perform drag-and-drop operations. Nonetheless, searching for graphical approaches can help you find ways to make the forms more intuitive and easier to use.

Complications and Limitations of a Graphical Approach

Several potential drawbacks exist to basing a form on the direct manipulation of graphical objects. The most important is that it can be an inefficient way to enter data. For example, you would not expect workers at a receiving dock to use a drag-and-drop form to record the receipt of several hundred boxes. A bar-code scanner would be considerably more efficient. Likewise, a quality control technician would prefer a simple keystroke (or voice) system so he or she could enter data without looking away from the task.

Even the Pet Store sales form is a debatable use of the drag-and-drop approach for in-store use. Think about the operations at a typical large pet store. Consider what would happen when dozens of customers bring shopping carts full of merchandise to the checkout counter. If a clerk has to use a drag-and-drop screen, the checkout process would take forever. Again, bar-code scanners would speed

up the process. On the other hand, perhaps the operations of the store could be improved by eliminating the checkout clerk. Think about how the store would function if shoppers used the store's drag-and-drop Web site to select products, which were then delivered, or bagged and stacked for drive-through pickup. The difference in the value of the approach depends on the operations of the business and on who will be using the application.

A second difficulty with the graphics approach is that each application requires a considerable amount of custom programming. The traditional approach is reasonably straightforward. Common tools exist for entering data with forms made up of text boxes, combo boxes, and subforms. These tools can be used for virtually any database application. On the other hand, direct manipulation of objects requires that individual business objects be drawn on the screen and associated with data. Then each user action (double-click and drag-and-drop) has to be defined specifically for that application. In the future tools may be created to assist in this programming. However, today, a graphical approach requires considerably more programming effort than other approaches.

Building graphical database applications across the Internet carries similar problems. There are two primary limitations: transmission speed and limitations of software tools. However, a huge amount of money and effort is being directed toward the Web. Many firms in several industries are working on solutions to both limitations.

Database Design Revisited

What problems arise if a design always uses one-to-many relationships? It is helpful to see how database design decisions affect the layout of a form. The key element to remember is that one-to-many relationships are built on forms as either a subform or a linked form. The "many" side of the relationship is repeating, so the form needs some mechanism for collecting multiple rows of data. Remember when you design a database you often have to decide if a relationship is one-to-one or one-to-many. If you take the lazy way out and make everything one-to-one, users will yell because it will not be possible to record the important data. For example, in the standard sales form, think about what would happen if the design allowed only one item to be sold at a time, instead of many. You would no longer need the SaleItems table and its corresponding subform. Instead, the Sales form would contain a single spot to select one product being sold. In some fields (such as real estate), this approach is reasonable. But, what happens when someone wants to purchase five items at the same time? The clerk could turn away the customer (which is really bad). Or, the clerk could initiate five different Sales transactions—each with a single item. It would work, but the clerk would be unhappy, and the customer would decide you had no clue how to build an information system, or run a store. So, you created the SaleItems table and added the subform to the Sales form to handle multiple items (rows) per sale.

Now look at the other extreme. You might opt for flexibility in your design and build every relationship as one-to-many. Again, consider the sales example. Most organizations assume that any given sale is processed by one employee for one customer. To handle a wider variety of situations, you decide to model both of those as one-to-many relationships: (1) Any sale can be made to many customers, and (2) Any sale can involve multiple employees. The catch is that you now need to add two tables to the design (SaleCustomers and SaleEmployees). Figure 6.25

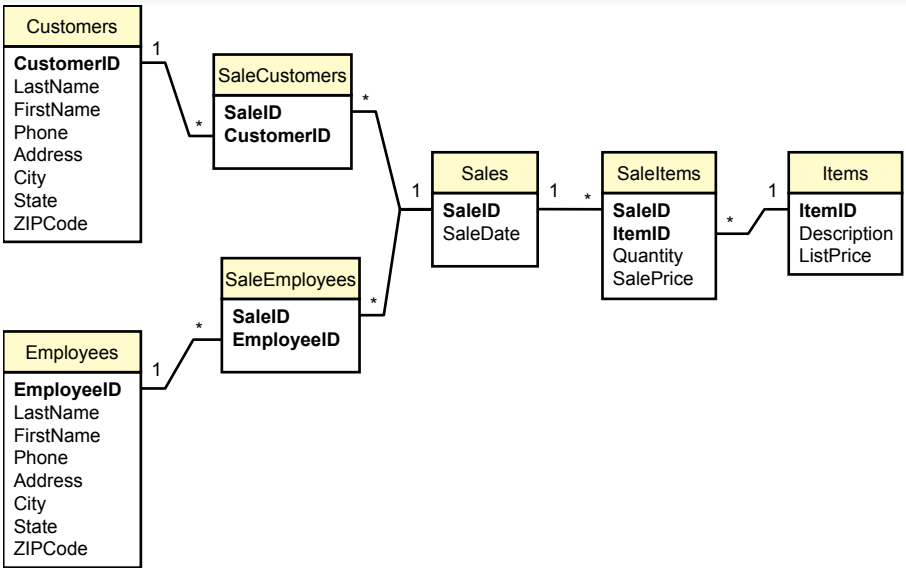


Figure 6.25

Database design variation. If there can be many customers and many employees per sale, the database design needs two new tables: SaleCustomers and SaleEmployees.

shows the two new tables. Observe that both SaleID and CustomerID (or EmployeeID) need to be part of the primary key.

More importantly, you need to add two repeating sections to the Sales form. Figure 6.26 shows a version of the new form. Notice the addition of the two scrolling regions for entering customers and employees. A traditional sales form would contain only a single entry for customer and for employee. If the organization often needs to record multiple customers and employees for each sale, there is nothing wrong with this version of the form. However, if the company almost always records only one entry for each, then this form is overkill. It takes up more screen space and is more difficult to understand. It will probably require more training to use—just to explain to clerks that most of the time, they will simply enter one customer and one employee.

The point is that you need to think about the usability of the forms when you design the database. You cannot just randomly choose one-to-one or one-to-many relationships. They must match the true needs of the organization.

Reports

What are the basic roles of reports? When you understand forms, reports are straightforward. Increasingly, the main difference between forms and reports is that forms are used to enter data and reports tend to have a set of fixed formats—with an emphasis on subtotals. Increasingly, reports are delivered electronically and sometimes have interactive elements, so the line between forms and reports is blurry. Some reports are still designed to be printed—such as receipts or invoices. However, report writers are increasingly Web based and managers use them to evaluate and summarize data using Web browsers. It is possible to create reports using traditional forms tools, but a report writer has two main strengths:

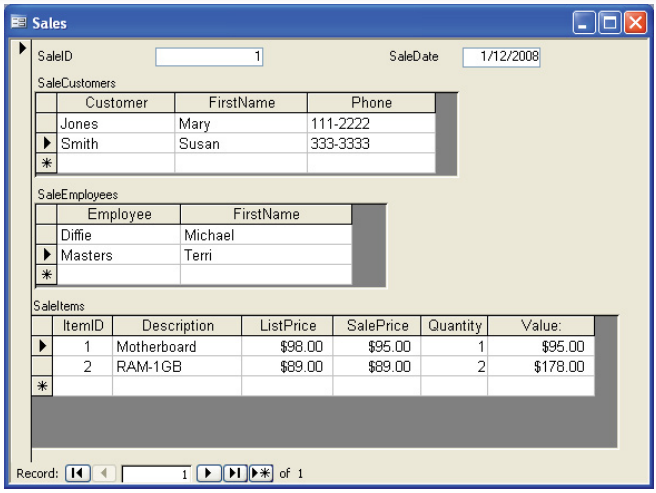


Figure 6.26

Database design variation. To handle many customers and many employees, two new subforms need to be added to the sales form. This form is more complex and more difficult to use than the standard sales form, so it should be avoided unless the users truly do need to record many customers and many employees.

(1) It can easily handle multiple pages of output (with consistent page headers and page numbering) and (2) It can combine both detailed and summary data. Chapter 5 illustrates how SQL queries can produce relatively complex results with the GROUP BY clause. However, a single SQL query can be used to display either detail rows of data or the summaries—not both. A good DBMS report writer also provides additional control over the output, such as printing negative values in red.

Report Design

As summarized in Figure 6.27, several issues are involved in designing reports. As in the development of forms, you and the users need to determine the content and layout. You must also identify the typical size of the report (number of pages and number of copies), along with noting how often it must be printed. Because of the physical steps involved, printing reports can be a time-consuming process. A report of a few dozen pages is no problem. However, when a report blooms into hundreds of pages with thousands of copies, you have to plan more carefully. First, you need a fast, heavy-duty printer. Then you need machines and people to assemble and distribute the report copies. You generally have to schedule time to use the printer for large reports.

Paper reports also present a different challenge to security. Paper reports require the use of more traditional security controls, such as written distribution lists, numbered copies, and control data. If security is an important issue in an organization, then these controls should be established when the report is designed.

Several physical and artistic aspects are involved in designing reports. The size of the page, the typeface used, and overall design of the page all must be determined. Newer DBMS report writers are relatively flexible, which is good and bad.

Report usage/user needs	Security controls
Report layout choices	Distribution list
Tabular	Unique numbering
Columns/subgroups	Concealed/nonprinted data
Charts/graphs	Secured printers
Paper sizes	Transmission limits
Printer constraints	Print queue controls
How often is it generated?	Output concerns
Events that trigger report	Typefaces
Size of the report	Readability
Number of copies	Size
Availability of color	User disabilities
	OCR needs

Figure 6.27

Fundamentals of report design. Determine content and layout with users. Estimate size and printing times. Identify security controls. Check typefaces and sizing for user readability.

The good part is that designers have greater control over the report. The bad part is that designers need to understand more about design—including the terminology.

Artistic design and a thorough treatment of design issues are beyond the scope of this book. If you are serious about design (for paper reports, forms, or Web pages), you should take a course in graphic design. In any case, it helps if you learn a few basic terms.

Terminology

Many of the basic terms come from typesetting and graphics design. The terms shown in Figure 6.28 will help you understand report writers and produce better reports. The first step is to choose the page layout, in terms of paper size; orientation (portrait versus landscape); and margins. The type of binding system will affect the margins, and you might have to leave an extra gutter margin to accommodate binding.

The next step is to choose the typeface and font size. In general, serif typefaces are easier to read, but sans serif faces have more white space, making them easier to read at larger and smaller sizes. Avoid ornamental typefaces except for covers and some headings. Columns of numbers are generally printed at a fixed width to keep columns aligned. Special fixed-width typefaces (e.g., Courier), in which all of the characters use exactly the same width, are especially appropriate if you need to align columns of nonnumeric data without the use of tab stops.

Font size is generally specified in terms of points. Most common printed material ranges from 10- to 12-point fonts. A useful rule of thumb is that a capitalized letter in a 72-point font is approximately 1 inch tall. Some report systems measure sizes and distance in picas. A pica is 1/6 of an inch, or the same height as a 12-point font (72/6 = 12).

If your reports include graphs and images, the terminology becomes more complex. Be aware that the quality of bitmap images depends on the resolution of the original image and the resolution of the output device. Common laser printers have a 1200-dots-per-inch (dpi) resolution. Typesetters typically achieve about

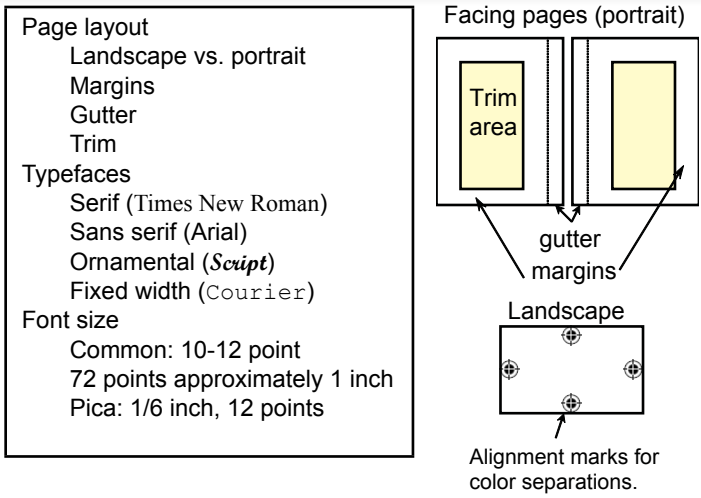


Figure 6.28

Basic publishing terminology. Understanding the basic design terms helps you design better reports and communicate with publishers and typesetters.

2,400-dpi resolution. An image that looks good on a 1200-dpi laser may be too small or too jagged on a 2,400-dpi typesetter.

If your reports are in color, you quickly encounter additional problems. In particular, colors on your screen may not be the same as on the printer. Similarly, a sample report printed on a color ink jet might look completely different when submitted to a typesetter. The Pantone® color standard is designed to minimize these problems by providing numbers for many standard colors. Advanced software also supports gamma correction that you can apply to your monitor so that colors displayed on your monitor will match those from the printer. The related issue you will encounter in color printing is the need to create color separations for all of your reports. For full-color submissions to print shops, each report page will need four separate color sheets. Denoted CMYK for each of the three primary colors—cyan (blue), magenta (red), and yellow—and the key color (black). In this case, each page will need high-resolution alignment marks so the colors can be reassembled properly.

One of the first elements of design that you must learn is to keep your reports simple and elegant. For instance, stick to one typeface and one or two font sizes on a page. Use plenty of white space to highlight columns and features. Look at the entire layout to observe where the eyes will be attracted and how they will move. Most important, since design style continually changes, examine newspaper and magazine layouts regularly for new ideas and patterns.

Basic Report Types

From the perspective of data layout, there are essentially three types of report designs: tabular, groups or subtotals, and labels. The choice you make depends on the type of data and use of the report. All report writers support these three basic formats. Many provide options to combine various elements.

Customer

CID	Phone	First Name	Last Name	Address	ZIP
1		Walkin	Walkin		
2	(808) 801-9830	Brent	Cummings	9197 Hatchet	96815
3	(817) 843-8488	Dwight	Logan	1760 Clearview	02109
4	(502) 007-0907	Shatika	Gilbert	4407 Green	40342
5	(701) 384-5623	Charlotte	Anderson	4333 Highland	58102
6	(606) 740-3304	Seeroba	Hopkins	3183 Highland	40330
7	(408) 104-9807	Anita	Robinson	8177 Horse Park	95035
8	(606) 688-8141	Cora	Reid	8351 Locust	41073
9	(702) 533-3419	Elwood	Henson	4042 West	89125
10	(302) 701-7398	Kaye	Maynard	5095 Sugar	19901

Figure 6.29

Tabular report layout. Tabular reports have few options but are good for detailed data listings. They are used for itemized listings of data.

Tabular and Label Reports

The tabular layout shown in Figure 6.29 is the simplest report design. It basically prints columns of data, much like the output of a query. The advantage over a simple query is that the tabular report can print page headings and page numbers on every page. You also have a little more control over font size and column width. Tabular reports are generally used for detail item listings, such as inventory reports. Note that the sort order becomes crucial, since these reports will be used to search for specific items. On the other hand, these reports do not contain much information for making decisions. In general, printed versions of these reports should no longer be needed. It is generally easier to use a form or report to lookup the specific items needed instead of carrying them around on paper.

As shown in Figure 6.30, labels are also straightforward. The essence of a label report is that all output for one row of data is printed in one "column" on the page. Then the next row is printed in the following column. The name label report comes from the use of preprinted or precut pages used for labels. These reports are sometimes named based on the number of physical columns. The example in Figure 6.30 has three labels across a page, so it is a three-up report. Before report writers, printing a label report was quite challenging, since the printer could only work from the top of the page. Hence, you had to write a program that printed the top line for three different rows of data, then return and print the second line, and so on. Today's printers are more flexible, and report writers make the job easy.

Keep in mind that label reports can be useful for other tasks—whenever you want to group data for one row into separate locations on a page. For instance, by inserting blank rows and changing the label size, you might create a tic-tac-toe pattern of data. It could be an interesting effect for a cover page or advertising sheet, but avoid using such patterns for hundreds of pages of data.

Groups or Subtotals

The most common type of report is based on groups and computes subtotals. It also provides the most flexibility over the layout of items on the report. Common examples include printing a receipt or a bill. Many times the report will print several rows of data, like the order form shown in Figure 6.31. Each order for the

Dwight Parrish 9904 Plum Springs Road Worcester, MA 01613	Dwight Logan 1760 Clearview Street Boston, MA 0210	David Sims 6623 Glenview Drive Boston, MA 02216
Hershel Keen 8124 Industrial Drive Nashua, NH 03080	Reva Kidel 5594 Halltown Road Bangor, ME 04401	Dan Kennedy 3108 Troon Court Burlington, VT 05401
Sharon Sexton 2551 Elementary Drive Barre, VT 05641	Kelly Moore 6116 Clearview Street Middlebury, VT 05753	Cassy Tuck 7977 Fairways Drive Clifton, NJ 07015

Figure 6.30

Tabular report layout. Tabular reports have few options but are good for detailed data listings. They are used for itemized listings of data.

month is printed in one report, but the items are grouped together to show the individual order subtotals. Many people refer to these reports as control break reports.

Some Web-based reporting tools support a level of interactivity with group reports. For example, Microsoft SQL Server Reporting Services supports icons for each subtotal level that enable the user to roll-up the details and show just the subtotals or drill-down to see the details in one section only. The process is a simplified version of data cube browsers presented in Chapter 9.

The key to the subtotal report is to note that it includes both detail item listings (item ordered, quantity, cost, etc.), and group or total data (order date, customer, and order total). To create this report, you first build a query that contains the data that will be displayed. The example would probably include the Order, OrderItem, Merchandise, Customer, Employee, and Supplier tables. Be careful: If you want to see the detail, do not include a GROUP BY statement in the query. If you examine the raw data from this huge query, you will see a large number of rows and columns—many with repeating data. That is fine at this point, but not exactly

Figure 6.31

Group or subtotal report. Note that several orders are being printed. Each order is a group and has a detailed repeating section of items being ordered. The report can compute subtotals for each order and a total for the entire report.

Merchandise Order

PONumber	ReceiveDate						
1	3/8/2013			Shipping Cost	$33.54		
Employee	4 Hopkins	Alan		Supplier	10 Rhodes	Brad	
ItemID	Quantity	Cost	Description		ListPrice	Category	Extended
27	8	$24.65	Aquarium Filter & Pump		$35.00	Fish	$197.20
30	208	$4.42	Flea Collar-Dog-Medium		$7.00	Dog	$919.36
				Sum	$1,116.56		

Report Header
Page Header
Group Header1
    Group Header2
        …
        Detail
        …
    Group Footer2
Group Footer1
Page Footer
Report Footer

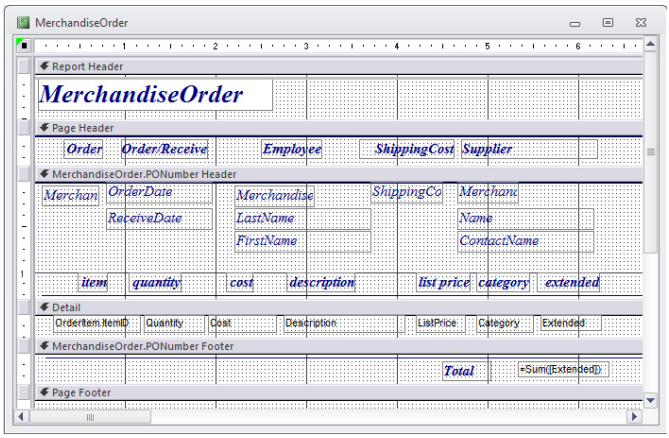


Figure 6.32

Report layout for groups. Note the basic report elements (report header, etc.). Also notice that the page layout is set by the position and properties of the data controls. In the sample report for Merchandise Orders, only one group is defined (on the Order number).

what the user wants to see. The objective of the report is to clean up the display of the data.

To create a grouped report, examine the report design shown in Figure 6.32. This layout page shows the **group breaks** in the data and specifies the layout of each element on the page. Again, layout is set by the individual controls. The controls have properties that can be changed to alter the appearance of the data displayed by that control. For example, you can set basic typeface and font attributes.

The basic elements of a report are headers, footers, group breaks and detail areas. The **report header** contains data that is displayed only when the report is first printed, such as a cover page. Similarly, the **report footer** is used to display data at the end of the report, for example, summary statistics or graphs. The **page header** and **page footer** are displayed on every page that is printed, except for the report header and footer pages. Page headers and footers can be used to display column headings, page numbers, corporate logos, or security identifiers.

The report features that define this type of report are the groups. The example shown in Figure 6.32 has one group defined: MerchandiseOrder.PONumber. The report will break, or create a new group of data, for each PONumber in the query. Notice that each Order can contain many items ordered. The report design specifies that these rows will be sorted by the ItemID number (within each order). Each grouping can have a group header and a group footer. The group header displays data that applies to the entire order (e.g., Date, Customer, Employee, and Supplier). It also holds the column labels for the detail (repeating) section. The group footer displays the subtotal for each group.

The common uses of each report element are summarized in Figure 6.33. Note that all of the elements (except detail) can work in pairs—headers and footers. You are not required to use both. For instance, you might choose to display page numbers in a page header and delete the page footer to provide additional space on the page.

Report Section	Usage
Report header	Title pages that are printed one time for entire report.
Page header	Title lines or page notes that are printed at the top of every page.
Group header	Data for a group (e.g. Order) and headings for the detail section.
Detail	Innermost data.
Group footer	Subtotals for the group.
Page footer	Printed at the bottom of every page—page totals or page numbers and notes.
Report footer	Printed one time at the end of the report. Summary notes, overall totals, and graphs for entire data set.

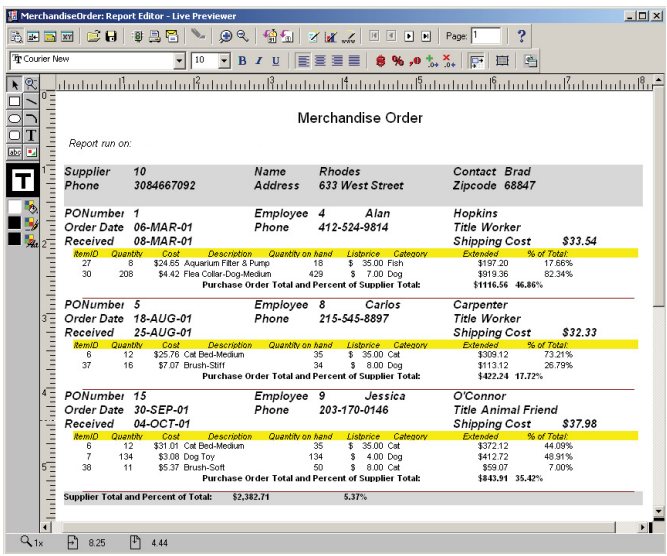
Figure 6.33

Common uses for report layout elements. Most elements are available in pairs, but you are free to delete any components you do not need.

For comparison, Figure 6.34 shows a similar report created using Oracle's 10g report writer. Notice that the overall structure is the same. Shading, boldface, and a highlight color have been used to improve the appearance of the report. However, it is still a little cramped, and could benefit from more white space. Ultimately, you would ask the users to focus the report on a specific item, and then improve the layout to highlight that item. In this version, the detail listing is highlighted,

Figure 6.34

Group report created and previewed with the report writer in Oracle. Shading is used to show the group headers and a highlight color draws attention to the detail heading.



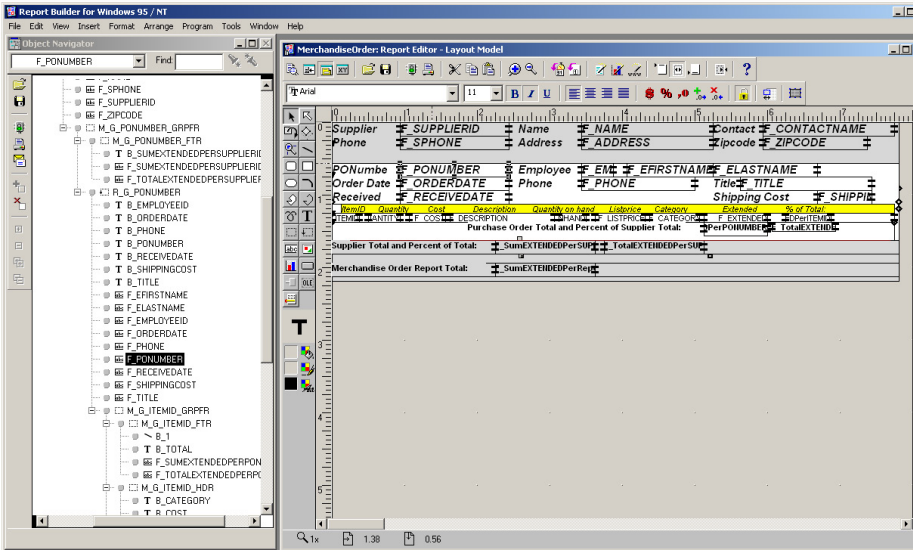


Figure 6.35

Group report design in Oracle. Shading is used to show the group headers and a highlight color draws attention to the detail heading.

but if the total value of the order is more important to the users, you would need to reorganize this report.

Figure 6.35 shows the design view for the Oracle report. First, notice the overall structure is controlled by the sections, which are shaded in the design view. Second, observe that the individual data elements are displayed using text box controls. These controls are similar to the text boxes on a form, but only display data. You can set properties to change the format, font, or layout of the data. Finally, check out the hierarchical tree listing on the left side. It shows the complete structure of the report and makes it easy to find and select individual sections and controls.

Groups represent one-to-many relationships. For example, each order can have many items in the detail section. If there are several one-to-many (or many-to-many) relationships in the data, you might want to use multiple levels of groups. As illustrated in Figure 6.36, each group is nested inside another group, with the detail at the innermost level.

To create this report, you must build a query that contains every item that will be displayed. Begin by focusing on the detail level and then join additional tables until you have all the columns you need. You can use computed columns for minor computations such as Price * Quantity. Be careful to avoid aggregate functions (e.g., Sum) and avoid the use of GROUP BY statements. The only time you might include these two features is if your "detail" row is actually a subtotal (or average) itself.

Group reports are generally used for computations—particularly subtotals. In general, computations on one row of data should be performed with the query. On the other hand, aggregations (Sum, Average, etc.) are handled by the report writer. Report writers have different methods of defining the scope of the operation—that is, what data should be included in the total.

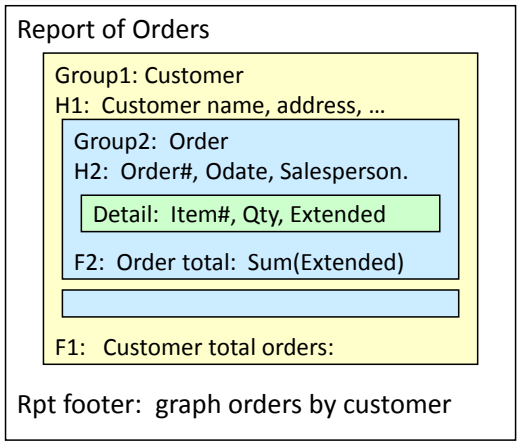


Figure 6.36

Nested groups. For example, each customer can place many orders, and each order has many detail lines. Two groups are used: (1) to show the total orders for each customer and (2) to show the total value of each order.

Charts

Reports are increasingly used for analysis and to identify patterns and trends in the data. As a result, users want charts to help them visualize the data. Charts on reports are similar to graphs on forms. The first step is to decide with the user what type of graph will best illustrate the data. The second step is to determine where the chart should be positioned within the report elements. If you are graphing detail items, then the graph belongs in the detail section, where it will be redrawn for every row of data. If it is a summary graph, it belongs in a group footer, or perhaps in the report footer if it summarizes data across the entire report.

Once you have determined the type and location of the graph, you build a query to collect the data. This query can be different from the query used to produce the overall report. In particular, when the graph is in a group footer, you might need to use aggregation functions in the query for the graph. Be sure to include a column that links the graph to the data in the report—even if that column will not be displayed on the graph. Figure 6.37 shows one sale on the Sales report for the Pet Store. The totals for the graph are computed by a separate query.

Summary

Forms must be designed to match the user's tasks and make your application easy to use. To meet this goal, you need to pay attention to design principles, operating system guidelines, and human limitations. Where possible, you should build the form to use direct manipulation of objects, such as dragging items from one location to another to signify shipment.

Forms are based on tables or queries. Each form has a single purpose and can store data in one table. More complex forms can be created by placing subforms onto the main form. Controls on the form are used to enter data into the tables, perform lookup functions, and manipulate data. Several standard controls are available for a Windows environment (e.g., text boxes, combo boxes, and option

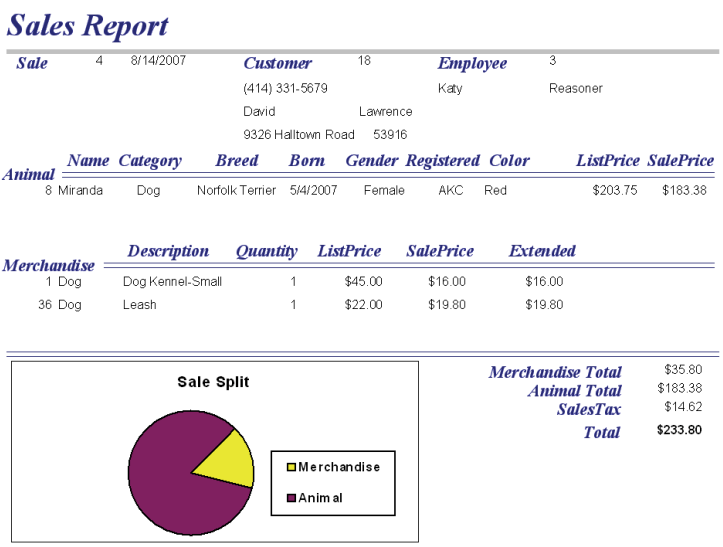


Figure 6.37

Sample graph on the sales report. It illustrates the portion of the sale spent on animals versus merchandise. Note that the graph appears on the same level as the Sale table—not on the detail level and not on the report footer.

buttons). Additional controls can be purchased to handle more complex tasks, such as calendars for scheduling and three-dimensional imaging.

Reports are generally printed and differ from forms because reports are designed only to present data, not to collect it. There are several types of forms, but many business forms rely on subtotals or groupings to display different levels of data. For example, a sales report might be grouped by sales division or salesperson or both. You use a query to combine all data items needed for a report. There are two benefits to using a report writer: (1) It is a straightforward way to set data formats and alignment, and (2) The report can include detail listings as well as subtotals and totals.

A Developer's View

As Miranda noted, the database wizards can create basic forms for you. However, before you crank up the form wizard and generate hundreds of small forms, think about the tasks of the users and the overall design. Try to put the most important information on one central form with a few secondary forms to help. Strive for a clean, well-organized screen and use colors and graphics sparingly to enhance the appearance. You should also develop a design standard and layout for the application to ensure consistency. Just be sure to leave room for creativity. For your class project, you should begin creating the basic forms and reports.

Key Terms

accessibility
aesthetics
check box
clarity
command button
consistency
controls
data-bound controls
direct manipulation of objects
drag-and-drop
event
feedback
focus
group break
heads-down data entry
human factors design
input mask
label
option button
page footer
page header
report footer
report header
resource file
single-row form
startup form
subform form
tab order
tabular form
text box
Unicode
use case

Review Questions

1. Which human factors are important to consider when designing forms?
2. How can you make your applications accessible to a wider group of workers?
3. How are international issues handled on forms?
4. What are the primary form types?
5. What are the main controls you can use on forms?
6. Why are updateable queries an issue with forms?
7. What is the purpose of subforms?
8. What is the purpose of linked forms?
9. What are the main report types?
10. What are the primary sections of reports?

Exercises

1. Research and report on the steps needed to create a multi-language version of a form using Microsoft .NET (Web version). Or use Java if your instructor prefers.
2. Review the documentation for the DBMS you are using and identify the features that it provides to support accessibility for all users.
3. Review the documentation for the DBMS you are using and identify the main forms controls that it provides.
4. Review the documentation for the DBMS you are using and describe how reports can be accessed via the Web.

For the following questions, create the databases and build the forms for the exercises from chapters 2 and 3. Note: Make initial forms. They do not have to be perfect matches to the drawings.

5. Medical test results, Chapter 2, Exercise 1.
6. Household appliance sales, Chapter 2, Exercise 2.
7. Repair service, Chapter 2, Exercise 3.
8. Day spa, Chapter 2, Exercise 4.
9. Glove manufacturer, Chapter 2, Exercise 5.
10. Custom ear bud manufacturer, Chapter 2, Exercise 6.
11. Automobile maintenance, Chapter 2, Exercise 7.
12. Farmer's market sales, Chapter 3, Exercise 1.
13. Exercise records, Chapter 3, Exercise 2.
14. Basketball league, Chapter 3, Exercise 3.
15. Network tracking, Chapter 3, Exercise 4.
16. Custom cell phone cases, Chapter 3, Exercise 5.
17. Vintage clothing sales, Chapter 3, Exercise 6.
18. Voter contacts, Chapter 3, Exercise 7.

Sally's Pet Store

19. Create basic forms to handle administrative tasks such as editing Breed, Category, AdoptionGroup, and Supplier data.

20. Create a form to edit Merchandise information. Eventually, this table will contain thousands of entries so provide features to enable users to sort and filter the data by category and description.
21. Create a form that makes it easy for adoption groups to enter new animal information. The group should only be able to edit data for their own animals that have not yet been adopted. (Assume they will eventually log in, but initially just use a drop-down list to have them set the Adoption Group.)
22. Create a form with a drop-down list to select a year/month and then display the total merchandise sales for that month by category. Include a chart.
23. Create a form that enables managers to select a starting and ending date and then display a chart showing total merchandise purchases over that time period for each supplier.
24. Create a report showing merchandise sales by category over time (monthly). Include a line chart.
25. Create a report showing total sales of merchandise by month for each supplier compared to monthly purchases from those manufacturers. Include a chart.
26. Create a report with a chart showing a count of the number of sales by employee by month.
27. Create a report showing total merchandise purchases by category for customers who have adopted at least one cat and one dog. (Hint: First build a query to get the list of customers who have adopted a cat and a dog.)

Rolling Thunder Bicycles

28. Create a form with a drop-down list for year that then displays a list of the top 10 customers for the year in terms of total sales. Hint: Use parameters in the query to refer to the control values.

29. Create a form that enables a manager to choose an employee, starting date, and ending date and display a chart of sales by model type for those conditions. Hint: Use parameters in the query to refer to the control values.
30. Create a form that enables employees to select a component and see the current quantity in stock along with a chart of sales (installations) by year/month.
31. Create a form that enables a manager to enter a start date and ending date and display a chart of sales by model type by month for that date range.
32. Create a sales report that shows total sales by employee by month.
33. Create a report that enables a manager to specify a start and end date and show total sales organized by model type between those dates. Hint: Use parameters in the query.

34. Create a sales report that shows total sales by model type for each month, organized by year.
35. Create a report to show total purchases and total payments to manufacturers by year.

Corner Med

36. Create a form that lets a physician select a diagnosis code and see all patients (sorted from the most recent) with that diagnosis..

37. Create a report that prints a bill for a patient visit.
38. Create a form that enables a physician to enter keywords and then search for ICD10 diagnosis codes that match those keywords. Note: You probably have to stick with relatively simple search methods.
39. Create a report that displays a chart for a specified patient. It should include all visits in chronological order and lists symptoms and treatments at each visit.
40. Create a report that displays the Bills (visits) that have been overpaid. Hint: First create a query to compute the total amount owed by visit.
41. Create a report that lists the count of the number of patients by month under the primary ICD diagnosis categories. The major groupings are based on the following table.

ICD10 Code	Category
A00-B99	1. Infectious and Parasitic Diseases
C00-D48	2. Neoplasms
D50-D89	3. Diseases of the blood and blood-forming organs…
E00-E90	4. Endocrine, nutritional and metabolic diseases
F00-F99	5. Mental and behavioral disorders
G00-G99	6. Diseases of the nervous system
H00-H59	7. Diseases of the eye and adnexa
H60-H95	8. Diseases of the ear and mastoid process
I00-I99	9. Diseases of the circulatory system
J00-J99	10. Diseases of the respiratory system
K00-K93	11. Diseases of the digestive system
L00-L99	12. Diseases of the skin and subcutaneous tissue
M00-M99	13. Diseases of the musculoskeletal system and connective tissue
N00-N99	14. Diseases of the genitourinary system
O00-O99	15. Pregnancy, childbirth and puerperium
P00-P96	16. Certain conditions originating in the perinatal period
Q00-Q99	17. Congenital malformations, … and chromosomal abnormalities
R00-R99	18. Symptoms, signs and abnormal clinical and lab findings, not elsewhere
S00-T98	19. Injury, poisoning and certain other consequences of external causes
V01-Y98	20. External causes of morbidity and mortality
Z00-Z99	21. Factors influencing health status and contact with health services
U00-U99	22. Codes for special purposes

Web Site References

http://www.microsoft.com/enable	Accessibility guidelines.
http://www.unicode.org	Primary site for Unicode information.
http://www.sigchi.org/	Association for Computing Machinery—Special Interest Group: Computer and Human Interaction.
http://www.sigaccess.org/	Association for Computing Machinery—Special Interest Group: Computers and the Physically Handicapped.

Additional Reading

Cooper, A. *About Face: The Essentials of User Interface Design.* Foster City, CA: IDG Books, 1997. [A good discussion of various design issues.]

Ivory, M. and M. Hearst, The State of the Art in Automating Usability, *Communications of the ACM*, 33(4), December 2001, 470-516. [General discussion on evaluating system usability.]

Koletzke, P. *Oracle Developer Advanced Forms and Reports*, Berkeley: Osborne/McGraw-Hill, 2000.

O'Reilly, Inc, Ed, *The Oracle PL/SQL CD Bookshelf: 7 Best Selling Books on CD ROM*, Cambridge, MA: O'Reilly & Associates, 2000. [A collection of several useful Oracle reference books on CD-ROM.]

Raskin, J. *Humane Interface*, The: New Directions for Designing Interactive Systems, Reading, MA: Addison-Wesley, 2000. [The need for a new interface as explained by the creator of the Apple Macintosh project.]

Tsichritzis, D. Form Management, *Communications of the ACM*, 25(7), July 1982. [Basic concepts of database forms.]