

ACCESS DESIGNING

VDC MEDIA – 2001

Tables of content

What is a Database?	2
Designing Your Database Application.....	6
Database Design Concepts	10
Building Your Database in Microsoft Access	22
Importing, Linking, and Exporting Data In Microsoft Access.....	35
Adding Power with Select Queries	53
Designing a Relational Database	66
Advanced Query Design—And SQL select commands language.....	70
Forms and Control tools	88
Access Tools in Designing Custom Multitable Forms	94
Creating Bound, Multiline, and Calculated Text Boxes	99
Designing Access Report	105
The Finishing Touches	108

What is a Database?

In the simplest sense, a *database* is a collection of records and files that are organized for a particular purpose. On your computer system, you might keep the names and addresses of all your friends or customers. Perhaps you collect all the letters you write and organize them by recipient. You might have another set of files in which you keep all your financial data—accounts payable and accounts receivable or your checkbook entries and balances. The word processor documents that you organize by topic are, in the broadest sense, one type of database. The spreadsheet files that you organize according to their uses are another type of database.

If you're very organized, you can probably manage several hundred spreadsheets by using folders and subfolders. When you do this, *you're* the database manager. But what do you do when the problems you're trying to solve get too big? How can you easily collect information about all customers and their orders when the data might be stored in several document and spreadsheet files? How can you maintain links between the files when you enter new information? How do you ensure that data is being entered correctly? What if you need to share your information with many people but don't want two people to try updating the same data at the same time? Faced with these challenges, you need a *database management system (DBMS)*.

Relational Databases

Nearly all modern database management systems store and handle information using the *relational* database management model. The term *relational* stems from the fact that each record in the database contains information related to a single subject and only that subject. Also, data about two classes of information (such as customers and orders) can be manipulated as a single entity based on related data values. For example, it would be redundant to store customer name and address information with every order that the customer places. In a relational database system, the information about orders contains a field that stores data, such as a customer number, which can be used to connect each order with the appropriate customer information. In a relational database management system, sometimes called an *RDBMS*, the system manages all data in tables. Tables store information about a subject (such as customers or students) and have columns that contain the different kinds of information about the subject (for example, customers' or students' addresses) and rows that describe all the attributes of a single instance of the subject (for example, data on a specific customer or student). Even when you *query* the database (fetch information from one or more tables), the result is always something that looks like another table.

Some Relational Database Terminology

- *Relation*—Information about a single subject such as customers, orders, students, or colleges. A relation is usually stored as a table in a relational database management system.
- *Attribute*—A specific piece of information about a subject, such as the address for a customer or the dollar amount of a contract. An attribute

is normally stored as a data column, or field, in a table.

- *Relationship*—The way information in one relation is related to information in another relation. For example, customers have a *one-to-many relationship* with orders because one customer can place many orders, but any order belongs to only one customer. Students might have a *many-to-many relationship* with colleges because each student is interested in applying to multiple colleges, and each college receives applications from many students.
- *Join*—The process of linking tables or queries on tables via their related data values. For example, customers might be joined to orders by matching customer ID in a customers table and an orders table.

You can also *join* information on related values from multiple tables or queries. For example, you can join student information with college application information to find out which students applied to which colleges. You can join employee information with contract information to find out which salesperson should receive a commission.

Database Capabilities

An RDBMS gives you complete control over how you define your data, work with it, and share it with others. The system also provides sophisticated features that make it easy to catalog and manage large amounts of data in many tables. An RDBMS has three main types of capabilities: data definition, data manipulation, and data control. All this functionality is contained in the powerful features of Microsoft Access. Let's take a look at how Access implements these capabilities and compare them to what you can do with spreadsheet or word processing programs.

Main Functions of a Database

- *Data definition*—You can define what data will be stored in your database, the type of data (for example, numbers or characters), and how the data is related. In some cases, you can also define how the data should be formatted and how it should be validated.
- *Data manipulation*—You can work with the data in many ways. You can select which data fields you want, filter the data, and sort it. You can join data with related information and summarize (total) the data. You can select a set of information and ask the RDBMS to update it, delete it, copy it to another table, or create a new table containing the data.
- *Data control*—You can define who is allowed to read, update, or insert data. In many cases, you can also define how data can be shared and updated by multiple users.

The Architecture of Microsoft Access

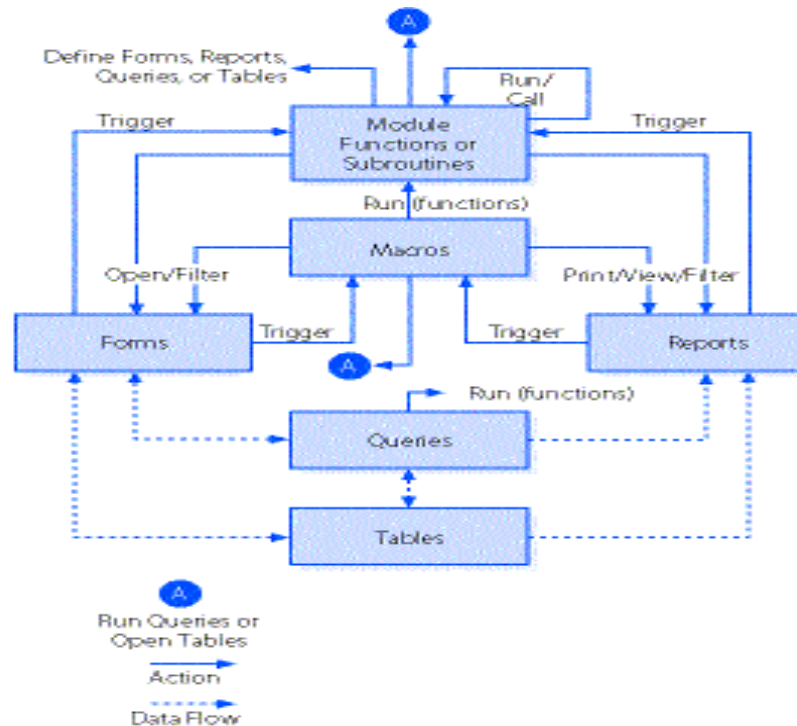
Microsoft Access calls anything that can have a name an *object*. Within an Access database, the main objects are tables, queries, forms, reports, macros, and modules.

If you have worked with other database systems on desktop computers, you might have seen the term *database* used to refer to only those files in which you store data. In Access, however, a database also includes all the major objects related to the stored data, including objects you define to automate the use of your data. Here is a summary of the major objects in an Access database:

Table	An object you define and use to store data. Each table contains information about a particular subject, such as customers or orders. Tables contain fields (or columns) that store different kinds of data, such as a name or an address, and records (or rows) that collect all the information about a particular instance of the subject, such as all the information about an entertainment group named The Belltones. You can define a primary key (one or more fields that have a unique value for each record) and one or more indexes on each table to help retrieve your data more quickly.
Query	An object that provides a custom view of data from one or more tables. In Access, you can use the graphical query by example (QBE) facility or you can write SQL statements to create your queries. You can define queries to select, update, insert, or delete data. You can also define queries that create new tables from data in one or more existing tables.
Form	An object designed primarily for data input or display or for control of application execution. You use forms to customize the presentation of data that your application extracts from queries or tables. You can also print forms. You can design a form to run a macro or a Visual Basic for Applications (VBA) procedure (see the sections on macros and modules below) in response to any of a number of events for example, to run a procedure when the value of data changes.
Report	An object designed for formatting, calculating, printing, and summarizing selected data. You can view a report on your screen before you print it.
Macro	An object that is a structured definition of one or more actions that you want Access to perform in response to a defined event. For example, you might design a macro that opens a second form in response to the selection of an item on a main form. You might have another macro that validates the content of a field whenever the value in the field changes. You can include simple conditions in macros to specify when one or more actions in the macro should be performed or skipped. You can use macros to open and execute queries, to open tables, or to print or view reports. You can also run other macros or VBA procedures from within a macro.
Module	An object containing custom procedures that you code using VBA. Modules provide a more discrete flow of actions and allow you to trap errors something you can't do with macros. Modules can be stand-alone objects containing functions that can be called from anywhere in your application, or they can be directly associated with a form or a report to respond to events on the associated form or report.

on the following page shows a conceptual overview of how objects in Access are related. Tables store the data that you can extract with queries and display in reports

or that you can display and update in forms. Notice that forms and reports can use data either directly from tables or from a filtered "view" of the data created by using queries. Queries can use VBA functions to provide customized calculations on data in your database. Access also has many built-in functions that allow you to summarize and format your data in queries.



Events on forms and reports can "trigger" either macros or VBA procedures. What is an event? An event is any change in state of an Access object. For example, you can write macros or VBA procedures to respond to opening a form, closing a form, entering a new row on a form, or changing data either in the current record or in an individual control (an object on a form or report that contains data). You can even design a macro or a VBA procedure that responds to the user pressing individual keys on the keyboard when entering data!

Activity for Week Opening

Create simple database base on information's the class member's details.

- Name contact.mdb
- Full details Name, address, suburb, state, Postcode, Phone number, Email address, Gender, age (optional), Note...etc..
- This will be a solution to do some exercise late on.
- Save in floppy disk with **your full name**.
- Hand in next week

Designing Your Database Application

You could begin building a database in Microsoft Access much as you might begin creating a simple single-sheet solution in a spreadsheet application such as Microsoft Excel by simply organizing your data into rows and columns and then throwing in formulas where you need calculations. If you've ever worked extensively with a database or a spreadsheet application, you already know that this unplanned approach works in only the most trivial situations. Solving real problems takes some planning; otherwise, you end up building your application over and over again. One of the beauties of a relational database system such as Access is that it's much easier to make midcourse corrections. However, it's well worth spending time up front designing the tasks you want to perform, the data structures you need to support those tasks, and the flow of tasks within your database application.

You don't have to go deeply into application and database design theory to build a solid foundation for your database project. You'll read about application design fundamentals in the next section, and then you'll apply those fundamentals in the succeeding sections. "Data Analysis." The section "Database Design Concepts" teaches you a basic method for designing the tables you'll need for your application and for defining relationships between those tables.

Application Design Fundamentals

Methodologies for good computer application design were first devised in the 1960s by recognized industry consultants such as James Martin, Edward Yourdon, and Larry Constantine. At the dawn of modern computing, building an application or fixing a broken one was so expensive that the experts often advised spending 60 percent or more of the total project time getting the design right before penning a single line of code.

Today's application development technologies make building an application much cheaper and faster. An experienced user can sit down with Microsoft Access on a PC and build in an afternoon what used to take months to create on an early mainframe system (if it was even possible). It's also easier than ever to go back and fix mistakes or to "redesign on the fly."

Today's technologies also give you the power to build very complex applications. And the pace of computing is several orders of magnitude faster than it was just a decade ago. But even with powerful tools, creating a database application (particularly a moderately complex one) without first spending some time determining what the application should do and how it should operate invites a lot of expensive time reworking the application. If your application design is not well thought out, it will also be very expensive and time-consuming later to track down any problems or to add new functionality.

The following is a brief overview of the typical steps involved in building a database application.

Step 1: Identifying Tasks

Before you start building an application, you'll probably have some idea of what you want it to do. It is well worth your time to make a list of all the major tasks you want to accomplish with the application including those that you might not need right away but might want to implement in the future. By "major tasks" I mean application functions that will ultimately be represented in a form or a report in your Access database. For example, "Enter customer orders" is a major task that you would accomplish by using a form created for that purpose, while "Calculate extended price" is most likely a subtask of "Enter customer orders" that you would accomplish by using the same form.

Step 2: Charting Task Flow

To be sure your application operates smoothly and logically, you should lay out the major tasks in topic groups and then order those tasks within groups on the basis of the sequence in which the tasks must be performed. For example, you probably want to separate employee-related tasks from sales-related ones. Within sales, an order must be entered into the system before you can print the order or examine commission totals.

You might discover that some tasks are related to more than one group or that completing a task in one group is a prerequisite to performing a task in another group. Grouping and charting the flow of tasks helps you discover a "natural" flow that you can ultimately reflect in the way your forms and reports are linked in your finished application. Later in this chapter, you'll see how I laid out the tasks performed in one of the sample applications included with this book.

Step 3: Identifying Data Elements

After you develop your task list, perhaps the most important design step is to list the data required by each task and the changes that will be made to that data. A given task will require some input data (for example, a price to calculate an extended amount owed on an order); the task might also update the data. The task might delete some data elements (remove invoices paid, for example) or add new ones (insert new order details). Or the task might calculate some data and display it, but it won't save the data anywhere in the database.

Step 4: Organizing the Data

After you determine all the data elements you need for your application, you must organize the data by subject and then map the subjects into tables and queries in your database. With a relational database system such as Access, you use a process called *normalization* to help you design the most efficient and most flexible way to store the data.

Step 5: Designing a Prototype and a User Interface

After you build the table structures needed to support your application, you can easily mock up the application flow in forms and tie the forms together using simple macros or Visual Basic for Applications (VBA) event procedures. You can build the actual forms and reports for your application "on screen," switching to Form view or Print Preview periodically to check your progress. If you're building the application to be used by someone else, you can easily demonstrate and get approval for the "look and feel" of your application before you write any complex code that's needed to actually accomplish the tasks.

Step 6: Constructing the Application

For very simple applications, you might find that the prototype *is* the application. Most applications, however, will require that you write code to fully automate all the tasks you identified in your design. You'll probably also need to create certain linking forms that facilitate moving from one task to another. For example, you might need to construct switchboard forms that provide the navigational road map to your application. You might also need to build dialog forms to gather user input to allow users to easily filter the data they want to use in a particular task. You might also want to build custom menus for most, if not all, of the forms in the application.

Step 7: Testing, Reviewing, and Refining

As you complete various components of your application, you should test each option that you provide. As you'll learn in this course later on, you can test macros by stepping through the commands you've written, one line at a time. If you automate your application using VBA, you'll have many debugging tools at your disposal to verify correct application execution and to identify and fix errors.

If at all possible, you should provide completed portions of your application to users so that they can test your code and provide feedback about the flow of the application. Despite your best efforts to identify tasks and lay out a smooth task flow, users will invariably think of new and better ways to approach a particular task after they've seen your application in action. Also, users often discover that some features they asked you to include are not so useful after all. Discovering a required change early in the implementation stage can save you a lot of time reworking things later.

The refinement and revision process continues even after the application is put into use. Most software developers recognize that after they've finished one "release," they often must make design changes and build enhancements. For major revisions, you should start over at step 1 to assess the overall impact of the desired changes so that you can smoothly integrate them into your earlier work.

Typical Application Development Steps

- Identifying tasks
- Charting task flow

- Identifying data elements
- Organizing the data
- Designing a prototype and a user interface
- Constructing the application
- Testing, reviewing, and refining

An Application Design Strategy

The two major schools of thought on designing databases are *process-driven design* (also known as *top-down design*), which focuses on the functions or tasks you need to perform, and *data-driven design* (also known as *bottom-up design*), which concentrates on identifying and organizing all the bits of data you need. The method used here incorporates ideas from both philosophies.

This method begins with you identifying and grouping tasks to decide whether you need only one database or more than one database. (This is a top-down approach.) As explained previously, databases should be organized around a group of related tasks or functions. For each task, you choose the individual pieces of data you need. Next you gather all the data fields for all related tasks and begin organizing them into subjects. (This is a bottom-up approach.) Each subject forms the foundation for the individual tables in your database. Finally, you apply the rules you will learn in the "Database Design Concepts" section of other lesson to create your tables.

Analyzing the Tasks

Let's assume that you've been hired by the Information Technology Group (ITG) at Microsoft to design a book catalog and order entry database for Microsoft Press. The database application must allow authorized users to enter and update book and author data. Potential customers who receive this catalog must be able to search for books of interest, select ones they want to order, search for nearby stores that carry Microsoft Press books, and print out an order that they can take to the store.

Database Design Concepts

When using a relational database system such as Microsoft Access, you should begin by designing each database around a specific set of tasks or functions. For example, you might design one database for customers and orders that contains data about each customer, the products available for sale, the orders for each customer, and the product sales history. You might have another database that handles human resources for your company. It would contain all relevant data about the employees and their dependents, such as names, job titles, employment histories, departmental assignments, insurance information, and the like.

At this point, you face your biggest design challenge: How do you organize data within each task-oriented database so that you take advantage of the relational capabilities of Access and avoid inefficiency and waste? If you followed the steps outlined earlier in this chapter for analysing application tasks and identifying database subjects, you're well on your way to creating a logical, flexible, and usable database design. But what if you just "dive in" and start laying out your tables without first analyzing tasks and subjects? The rest of this chapter shows you how to apply some rules to transform a makeshift database design into one that is robust and efficient.

Waste Is the Problem

A table stores the data you need for the tasks you want to perform. A table is made up of columns, or fields, each of which contains a specific kind of data (such as a customer name or a credit rating), and rows, or records, that collect all the data about a particular person, place, or thing. You can see this organization in the Customers table in the Microsoft Press Books database, as shown in [Figure 3-1](#).

Customer ID	Last Name	First Name	M.I.	Address	City	State/Province	Postal
1	Davolio	Nancy	Q	507 - 20th Ave. E.	Seattle	WA	98122
2	Fuller	Andrew	R	908 W. Capital Way	Tacoma	WA	98401
3	Leverling	Janet	V	722 Moss Bay Blvd.	Kirkland	WA	98033
4	Peacock	Margaret	W	4110 Old Redmond F	Redmond	WA	98052
5	Buchanan	Steven	V	13920 S.E. 40th Stre	Bellevue	WA	98006
6	Suyama	Michael	J	112 Pike	Seattle	WA	98123
7	King	Robert	B	Edgeham Hollow	London		RG1 9S
8	Callahan	Laura	Q	4726 148th N.E.	Bellevue	WA	98008
9	Dodsworth	Anne	M	7 Houndstooth Rd.	London		WG2 7L
10	Viescas	John	L	15127 NE 24th, Suit	Redmond	WA	98052

FIGURE 3-1

The Customers table in Datasheet view.

For the purposes of this design exercise, let's say you want to build a new database (named Books) for creating book orders without the benefit of first analyzing the tasks and subjects you'll need. You might be tempted to put all the data about the task you want to do—keeping track of customers and the books they order—in a single Customer-Orders table, whose fields are represented in [Figure 3-2](#) on the following page.

There are many problems with this technique. For example:

- Every time a customer adds another order, you have to duplicate the Customer Name and Customer Address fields in another record for the new order. Repeatedly storing the same name and address in your database wastes a lot of space and you can easily make mistakes if you have to enter basic information about a customer more than once.
- You have no way of predicting how many titles will be ordered in any given order. If you keep track of each order in a single record, you have to guess the largest number of titles and leave space for Book 1, Book 2, Book 3, and so on, all the way to the maximum number. Again you're wasting valuable space in your database. If you guess wrong, you'll have to change your design just to accommodate an order that has more than the maximum number of titles. And later, if you want to find out which books were sold to what customers, you'll have to search each Book Name field in every record.
- You have to waste space in the database storing data that can easily be calculated when it's time to print a report. For example, you'll certainly want to calculate the total order amount for each order, but you do not need to keep the result in a field.
- Designing one complex field to contain all the parts of simple data items (for example, lumping together Street Address, City, State, and Postal Code) makes it difficult to search or sort on part of the data. In this example, it would be impossible to sort on customer zip code because that piece of information might appear anywhere within the more complex single address field.

Customer Orders

Order Date	Order Total	Customer Name	Customer Address, City, State, Postal	Customer Country	Customer Phone	Store Name	Store Address, City, State, Postal	Store Phone
Book 1 Title	Book 1 Author 1 Name	Book 1 Author 1 Bio	Book 1 Author 2 Name	Book 1 Author 2 Bio	Book 1 Sugg. Price	Book 1 Quantity	Book 1 Discount	Book 1 Extended Price
Book 2 Title	Book 2 Author 1 Name	Book 2 Author 1 Bio	Book 2 Author 2 Name	Book 2 Author 2 Bio	Book 2 Sugg. Price	Book 2 Quantity	Book 2 Discount	Book 2 Extended Price
...
Book n Title	Book n Author 1 Name	Book n Author 1 Bio	Book n Author 2 Name	Book n Author 2 Bio	Book n Sugg. Price	Book n Quantity	Book n Discount	Book n Extended Price

FIGURE 3-2. The design for the Books database using a single Customer-Orders table.

Normalization Is the Solution

You can minimize the kinds of problems noted above (although it might not always be desirable to eliminate all duplicate values), by using a process called *normalization* to organize data fields into a group of tables. The mathematical theory behind normalization is rigorous and complex, but the tests you can apply to determine whether you have a design that makes sense and that is easy to use are quite simple and can be stated as rules.

Rule 1: Field Uniqueness

Since wasted space is one of the biggest problems with an unnormalized table design, it makes sense to remove redundant fields from a table. So the first rule is about field uniqueness.

Rule 1: Each field in a table should represent a unique type of information.

This means that you should break up complex compound fields and get rid of the repeating groups of information. In this example, the complex address fields should be separated into simple fields and new tables designed to eliminate the repeating book information. When you create separate tables for the repeating data, you include some "key" information from the main table to create a link between the new tables and the original one. One possible result is shown in Figure 3-3.

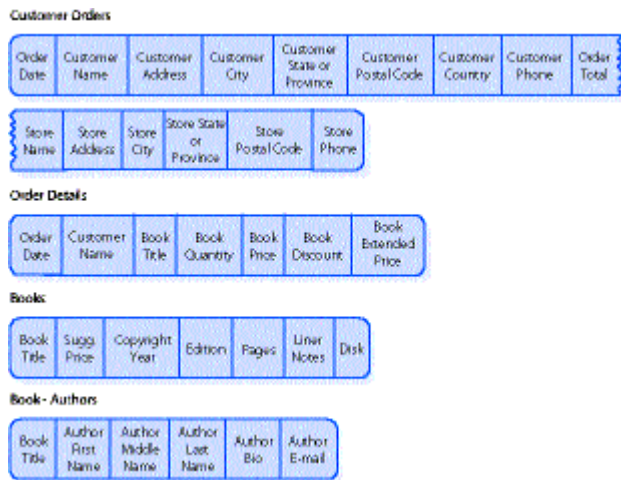


FIGURE 3-3.

A design for the Books database that eliminates redundant fields.

These tables are much simpler because you can process one record per book ordered. Also, you don't have to reserve room in your records to hold a large number of books per order. And, if you want to find out what book has the highest price, you can now search the separate Books table, where key information about each book is recorded only once.

The duplicate data problem is now somewhat worse, however, because you are repeating the Order Date and Customer Name fields in each Order Details record. The potentially long Book Title field is also redundant in the Books, Book-Authors, and Order Details tables. This "duplicate" data is necessary, however, to maintain the links between the tables. You can solve this problem by following the second rule.

Rule 2: Primary Keys

In a good relational database design, each record in any table must be uniquely identified. That is, some field (or combination of fields) in the table must yield a unique value for each record in the table. This unique identifier is called the *primary key*.

Rule 2: Each table must have a unique identifier, or primary key, that is made up of one or more fields in the table.

Whenever possible, you should use the simplest data that "naturally" provides unique values. Nearly all books published in the world have a relatively short (12 character) International Standard Book Number or ISBN that uniquely identifies each book. This makes the ISBN field a good "natural" primary key for the Books table. Although it appears that you've created duplicate data with the book ISBN field in three of the tables, you've actually significantly reduced the total amount of data stored. The lengthy book title data is stored only once for each book in the Books table and not for each detail line in an order. You've duplicated only a small piece of data, the ISBN field, which allows you to *relate* the order detail and author data to the appropriate book data. Relational databases are equipped to support this design technique by giving you powerful tools to bring related information back together easily.

Whenever you build a table, Access always recommends that you define a primary key for that table. For many tables, you might need to create an artificial unique value to act as the primary key. The Books application will probably generate a unique order number or Order ID for each new order entered. (You'll see in the next chapter that Access provides a special data type, called AutoNumber, that generates a unique number for every new row in a table.) In the case of Order Details, the combination of the Access-generated Order ID and the book ISBN is most likely unique for each row in the table (you're not likely to create more than one order detail line for a particular title in a single order). The result of adding primary keys is shown in **Figure 3-4**.

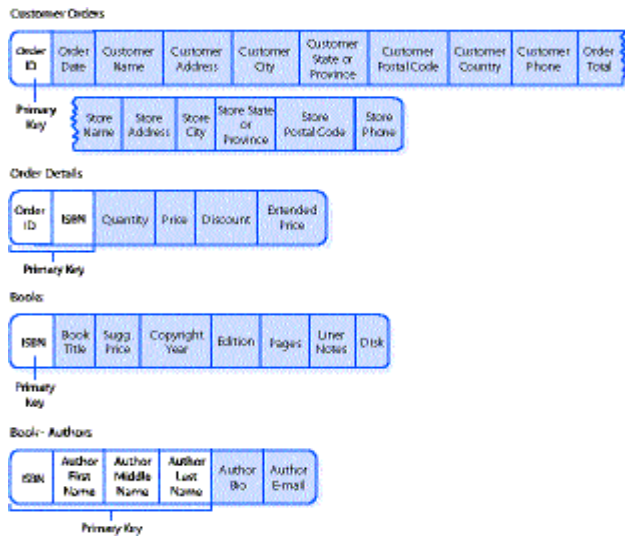


Figure 3-4.
The Books database tables with primary keys defined.

Rule 3: Functional Dependence

After you define a primary key for each table, you can check to see whether you included all the data relevant to the subject of the table. In other words, you should check to see whether each field is *functionally dependent* on the primary key that defines the subject of the table.

Rule 3: For each unique primary key value, the values in the data columns must be relevant to, and must completely describe, the subject of the table.

This rule works in two ways. First, you shouldn't have any data in a table that is not relevant to the subject (as defined by the primary key) of the table. For example, although customer information is required for each order, customers are in fact a separate subject and should have their own table. Likewise, an author may write more than one book, so creating a separate table for authors makes sense. Second, the data in the table should completely describe the subject. For example, books in an order may be shipped to a person and location different from the customer who is buying the books (perhaps as a gift). Adding shipping information to the Orders table makes that information more complete. The result is shown in **Figure 3-5**.



FIGURE 3-5.

Creating additional subject tables in the Books database to ensure all fields in a table are functionally dependent on the primary key of the table.

Rule 4: Field Independence

The last rule checks to see whether you'll have any problems when you make changes to the data in your tables.

Rule 4: You must be able to make a change to the data in any field (other than a field in the primary key) without affecting the data in any other field.

Take a look again at the Orders table in **Figure 3-5**. As we applied the second and third rules, we left Store information with the Orders information because it seems reasonable that you need Store information to complete an order. Note that if you need to correct the spelling of a store name, you can do so without affecting any other fields in that record. If you misspelled the same store name for many orders, however, you might have to change many records. Also, if you entered the wrong store (the order is actually for Powell's Technical Bookstore, not University Bookstore), you can't change the store name without also changing that record's address and phone data. The Store Name, Store Address, and Store Phone fields are not independent of one another. In fact, Store Address, Store City, Store State, and Store Phone are functionally dependent on Store Name. (See Rule 3.) Although it wasn't obvious at

first, Store Name describes another subject that is different from the subject of orders. You can see how carefully applying this fourth rule helps you identify changes that you perhaps should have made when applying earlier rules. This situation calls for another table in your design: a separate Stores table, as shown in [Figure 3-6](#) on the following page.

Now, if you've misspelled a store name, you can simply change the store name in the Stores table. Also, instead of using the Store Name field (which might be 40 or 50 characters long) as the primary key for the Stores table, you can create a shorter Store ID field (perhaps a five-digit number) to minimize the size of the relational data you need in the Order table.

Note also that the Order Total field has been removed from the Orders table and the Price and Extended Price fields have been removed from the Order Details table. Because the price of a book rarely changes, it makes little sense to carry the price in both the Books table and the Order Details table. As you'll see later when you learn about building queries, it's a simple matter to link the Order Details table with the Books table in a query to retrieve the price and calculate the extended price for each book. Likewise, Order Total is removed from the Orders table because any change to a price, order quantity, or discount will cause a change in the total. It's better to calculate the total order value when the order is complete perhaps as part of the report that prints the order.

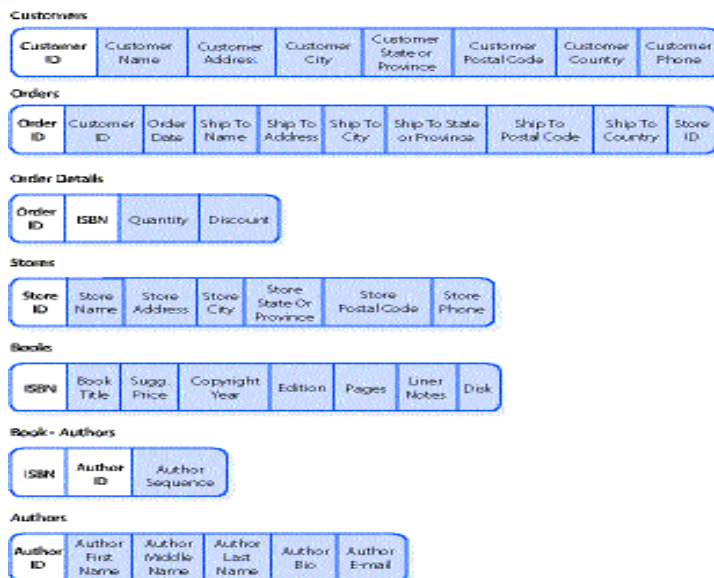


FIGURE 3-6. A design for the Books database that follows all the design rules.

An alternative (but less rigorous) way to check for field independence is to see whether you have the same data repeated in your records. In the previous design, whenever you created an order for a particular store, you had to enter the store's name, address, state, zip, and phone number in the order record. With a separate Stores table, if you need to correct a spelling or change an address, you have to make the change only in one field of one record in the Stores table. If you entered the wrong store in an order, you have to change only the Store ID in the Orders table to fix the problem.

The actual Microsoft Press Books sample database includes ten tables, which are all shown in the Relationships window in **Figure 3-7**. Notice that additional fields were created in each table to fully describe the subject of each table and that other tables were added to support some of the other tasks identified earlier in this section. For example, fields were added to the Orders table to capture payment method and credit card number. New tables were added to provide a means to designate book categories and category classes, such as Intermediate (a category) and Book Audience (a class).

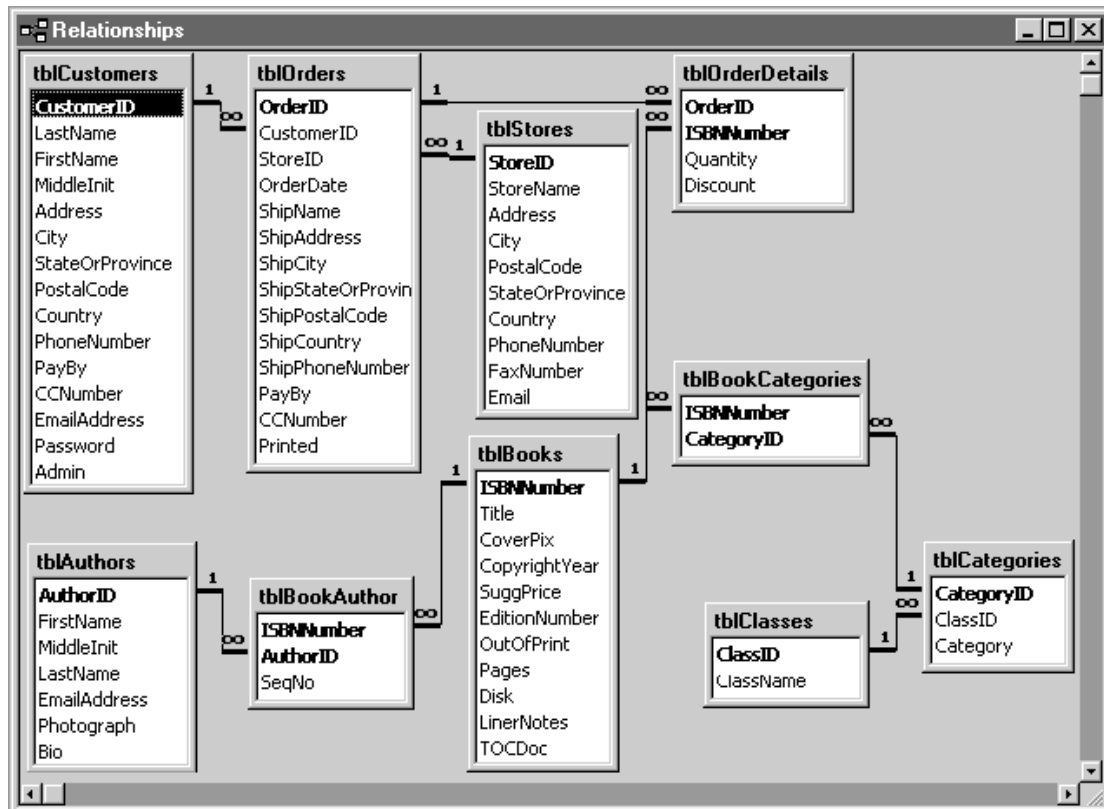


FIGURE 3-7 Tables in the Microsoft Press Books sample database shown in the Relationships window.

The Four Rules of Good Table Design

Rule 1: Each field in a table should represent a unique type of information.

Rule 2: Each table must have a unique identifier, or primary key, that is made up of one or more fields in the table.

Rule 3: For each unique primary key value, the values in the data columns must be relevant to, and must completely describe, the subject of the table.

Rule 4: You must be able to make a change to the data in any field (other than a field in the primary key) without affecting the data in any other field.

Efficient Relationships Are the Result

When you apply good design techniques, you end up with a database that efficiently links your data. You probably noticed that when you normalize your data as recommended, you tend to get many separate tables. Before relational databases were invented, you had to either compromise your design or manually keep track of the relationships between files or tables. For example, you had to put customer data in your Orders table or write your program to first open and read a record from the Orders table and then search for the matching record in the Customers table. Relational databases solve these problems. With a good design you don't have to worry about how to bring the data together when you need it.

Foreign Keys

You might have noticed as you followed in the Microsoft Press Books example that each time you created a new table, you left behind a field that could link the old and new tables, such as the Customer ID and the Store ID fields in the Orders table. These "linking" fields are called *foreign keys*.

In a well-designed database, foreign keys result in efficiency. You keep track of related foreign keys as you lay out your database design. When you define your tables in Access, you link primary keys to foreign keys to tell Access how to join the data when you need to retrieve information from more than one table. As you'll learn in *Next Lesson*, you can add indexes to your foreign key fields to improve performance. You can also ask Access to maintain the integrity of your table relationships for example, Access will ensure that you don't create an order for a product that doesn't exist. When you ask Access to maintain this *referential integrity*, Access automatically creates indexes for you.

One-to-Many and One-to-One Relationships

In most cases, the relationship between any two tables is one-to-many. That is, for any one record in the first table, there are many related records in the second table, but for any record in the second table, there is exactly one matching record in the first table. You can see several instances of this type of relationship in the design of the Microsoft Press Books database. For example, each customer might have several orders, but a single order record applies to only one customer.

Occasionally, you might want to break down a table further because you use some of the data in the table infrequently or because some of the data in the table is highly sensitive and should not be available to everyone. For example, you might want to keep track of certain customer data for marketing purposes, but you don't need access to that data all the time. Or you might have data about credit ratings that should be accessible only to authorized people. In either case, you can create a separate table that also has a primary key of CustomerID. The relationship between the original Customers table and the Customer Info or Customer Credit table is one-to-one. That is, for each record in the first table, there is exactly one record in the second table.

Creating Table Links

The last step in designing your database is to create the links between your tables. For each subject, identify those for which you wrote *Many* under "Relationship" on the worksheet. Be sure that the corresponding relationship for the other table is *One*. If you see *Many* in both places, you must create a separate *intersection table* to handle the relationship. (Access won't let you define a many-to-many relationship directly between two tables.) In the example of the Order Books task, a customer may have an order for "many" books, and a book can appear in many orders. The OrderDetails table in the Microsoft Press Books database is an intersection table that clears up this many-to-many relationship between orders and books. BookAuthor is another table that works as an intersection table because it has a one-to-many relationship with both Authors and Books. (A book can have more than one author, and an author might write more than one book.)

After you straighten out the many-to-many relationships, you need to create the links between tables. To complete the links, place a copy of the primary key from the "one" tables in a field in the "many" tables. *The same Worksheet tasks that you did last week you should writing down the relations for table, so that you can surmise that the primary key for the Orders table, OrderID, also needs to be a field in the OrderDetails table.*

When to Break the Rules

As a starting point, for every application that you build, you should always analyze the tasks you need, decide on the data required to support those tasks, and create a well-designed (also known as *normalized*) database table structure. After you have a design that follows all the rules, you might discover changes that you need to make either to follow specific business rules or to make your application more responsive to the needs of your users. In every case for which you decide to "break the rules," you should know the specific reason for doing so, document your actions, and be prepared to add procedures to your application to manage the impact of those changes. The following sections discuss some of the reasons why you might need to break the rules.

Improving Performance of Critical Tasks

The majority of cases for breaking the rules involve manipulating the design to achieve better performance for certain critical tasks. For example, although modern relational database systems (like Microsoft Access) do a good job of linking many related tables back together to perform complex tasks, you might encounter situations in which the performance of a multiple-table link is not fast enough. Sometimes if you "demoralise" selected portions of the design, you can achieve the required performance. For example, instead of building a separate table of book classification codes that requires a link, you might place the classification descriptions directly in the book categories table. If you choose to do this, you will need to add procedures to the forms you provide to edit these categories to make sure that similar descriptions aren't duplicate entries.

Another case for breaking the rules is the selective inclusion of calculated values in your database. For example, if a critical management report needs the calculated totals for all orders, but the data is retrieved too slowly when calculating the detailed values

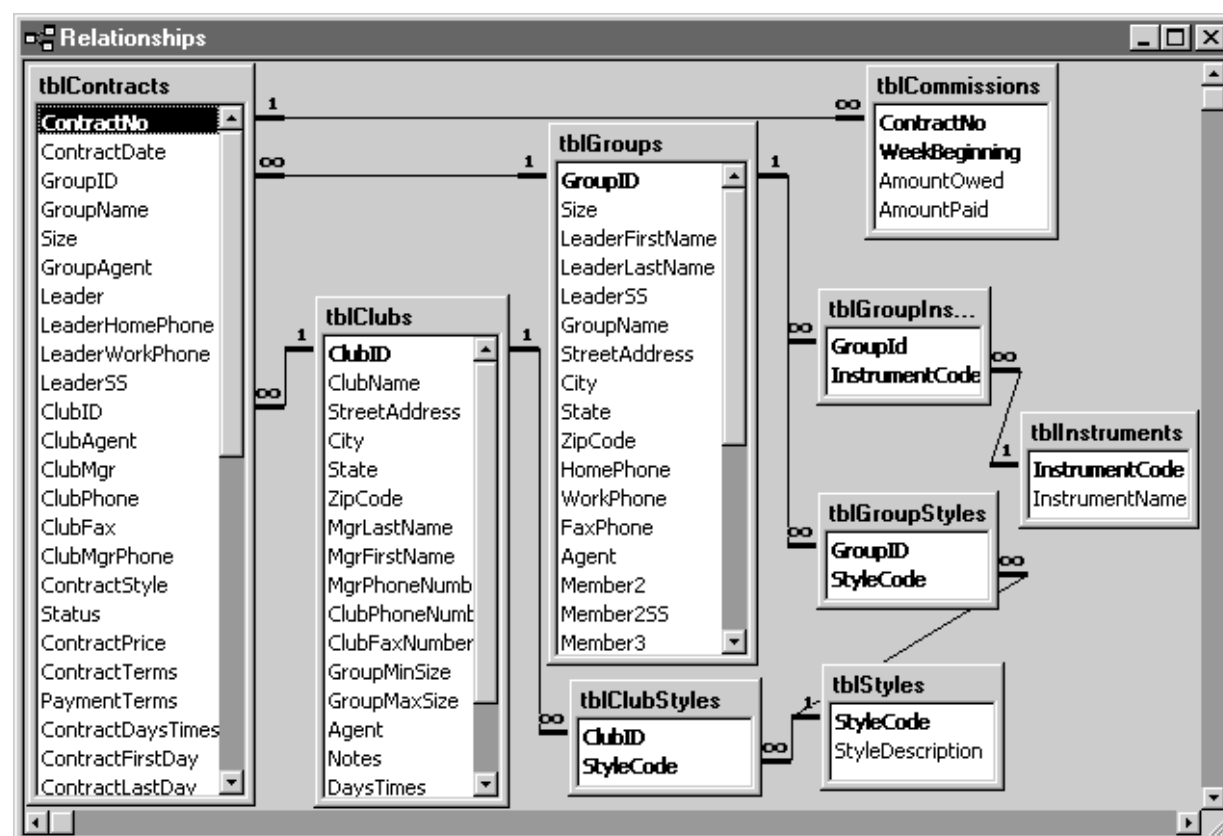
for thousands of order detail records, you might want to add a field for order total in the Orders table. Of course, this also means adding procedures to your order-entry forms to ensure that any change in an order detail record is reflected in the calculated order total. Your application will spend a few extra fractions of a second processing each order so that month-end totals can be obtained quickly.

Capturing Point-In-Time Data

Sometimes you need to break the rules to follow known business rules. In the previous design exercise, we removed the Price field from the Order Details table because it duplicated the price information in the Books table. However, if your business rules say that the price of a book (or any product) can change over time, you may need to include the price in your order details to record the price at the *point in time* that the order was placed. If your business rules dictate this sort of change, you should add procedures to your application to automatically copy the "current" price to any new order detail row.

You can see a similar case in the Microsoft Press Books database. Some of the shipping name and address information in the Orders table may duplicate information in the Customers table. If you examine the way the Books database works, you'll find some code that automatically copies the customer information to the shipping information when you create a new order. The user is free, however, to change the shipping information as required by the order.

There's a similar example in the Entertainment Schedule database. In this database, the user creates new contracts that specify the commitment of an entertainment group to perform at a specific club on a specific date. The information about clubs remains fairly constant, but groups change their name and membership all the time. If you look at the database design for the Entertainment Schedule database, shown in *figure below* on the following page, you'll see what looks like lots of duplicate information in the Contracts table. For example, the club manager who arranged the contract is copied to the contract (the manager might be different next week!). In addition, all the information about the group (Group Name, Leader Name, and Member Names) is copied to the contract to capture the information at the point in time that the contract was signed. If you look in the Entertainment Schedule sample database, you'll find lots of code in the contract edit form (frmContracts) to make this happen automatically.



The design for the Entertainment Schedule database includes "duplicate" point-in-time group information in the tblContracts table.

Report Snapshot Data

One additional case for breaking the rules involves accumulating data for reporting. As you'll learn in "*Advanced Report Design*," later on about the queries required to collect data for a complex report can be quite involved. If you have a lot of data required for your report, running the query could take an unacceptably long time, particularly if you need to run several large reports from the same complex collection of data. In this case, it's acceptable to create temporary but "rule-breaking" tables that you load once with the results of a complex query in order to run your reports. I call these tables "snapshots" because they capture the results of a complex reporting query for a single moment in time. You can look in Solution learning, "Modifying Data with Action Queries," for some ideas about how to build action queries that save a complex data result to a temporary table. If you use the resulting "snapshot" data from these tables, you can run several complex reports without having to run long and complex queries more than once.

Now that you understand the fundamentals of good database design, you're ready to do something a little more fun with Access—building a database, try to add more feature in your *Contact database*. The next Lesson "*Building Your Database in Microsoft Access*," shows you how to create a new database and tables with all the principle and Rules which you are learned in this week; Please reads in Solution learning about, "*How Modifying Your Database Design*,"

shows you how to make changes later if you discover that you need to modify your design

Building Your Database in Microsoft Access

After you design the tables for your database, defining them using Microsoft Access is incredibly easy. This chapter shows you how it's done. You'll learn how to:

- Create a new database application using the Database Wizard
- Create a new empty database for your own custom application
- Create a simple table by entering data directly in the table
- Get a jump-start on defining custom tables by using the Table Wizard
- Define your own tables from scratch
- Select the best data type for your fields
- Set validation rules for your fields and tables
- Tell Access what relationships to maintain between your tables
- Optimize data retrieval by adding indexes
- Print a table definition

Creating a New Database

When you first start Microsoft Access, you see the opening choices dialog box shown in [Figure 4-1](#). In this dialog box you specify whether you want to create a brand-new empty database, to use the Database Wizard to create a database application using any of the more than 20 database application templates that come with Access, or to open an existing database (mdb) file. If you've previously opened other databases, such as the Northwind Traders sample database that is included with Access, you'll also see a "most recently used" list of up to four database selections in the Open An Existing Database section of the dialog box. If you have Microsoft Office installed, when you open Access for the first time you'll also see the Microsoft Office Assistant in the lower right corner of your screen (as shown in [Figure 4-11](#) later in this chapter). If this is the first time you've used Access, choose Start Using Microsoft Access in the Office Assistant. You can learn about using the Office Assistant and Access online help in more detail later in this chapter.

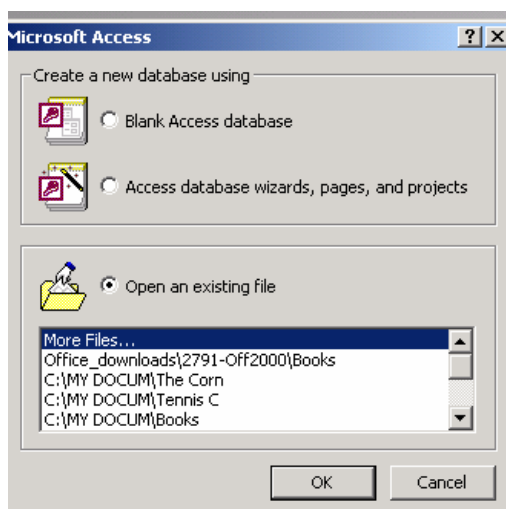


FIGURE 4-1.
The Access opening choices dialog box.

Using the Database Wizard

Just for fun, let's explore the Database Wizard first. If you're a beginner, you can use the Database Wizard to work with any of the more than 20 database application templates included with Access without needing to know anything about designing database software. You might find that the application the wizard builds meets most of your needs right off the bat. As you learn more about Access, you can build on and customize the basic application design and add new features.

Even if you're an experienced developer, you might find that the application templates save you lots of time in setting up the basic tables, queries, forms, and reports for your application. If the application you need to build is covered by one of the templates, the wizard can take care of many of the simpler design tasks.

When you start Access, you can select the Database Wizard option in the opening choices dialog box and then click OK to open the dialog box shown in [Figure 4-2](#). Or, if you have already started Access, you can choose New Database from the File menu. You work with all of the templates in the Database Wizard in the same way. This example will show you the steps that are needed to build a Household Inventory database.

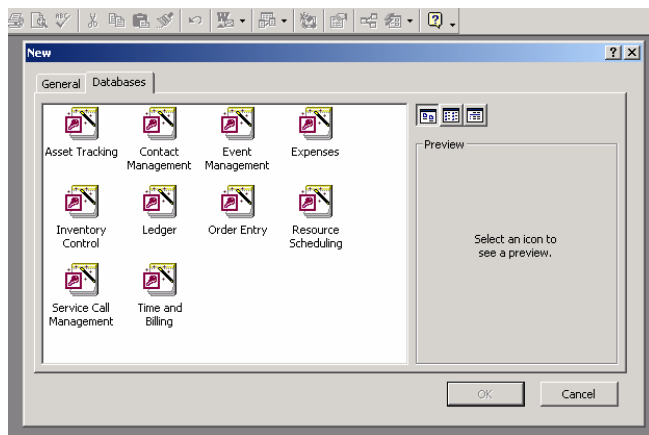


FIGURE 4-2.
Some of the Database Wizard templates.

Scan the list of available templates under the Databases tab of the New dialog box. When you click on a template icon, Access shows a preview graphic to give you another hint about the purpose of the template. You start the Database Wizard by selecting a template and then clicking OK. You can also double-click on a template icon. Access opens the File New Database dialog box and suggests a name for your new database file. You can modify the name and then click Create to launch the wizard.

The wizard takes a few moments to initialise and to create a blank file for your new database application. The wizard first displays a screen with a few more details about the capabilities of the application you are about to build. If this isn't what you want, click Cancel to close the wizard and delete the database file. You can click Finish to have the wizard quickly build the application with all the default options. Click Next

to proceed to a window that provides options for customizing the tables in your application, as shown in [Figure 4-3](#).

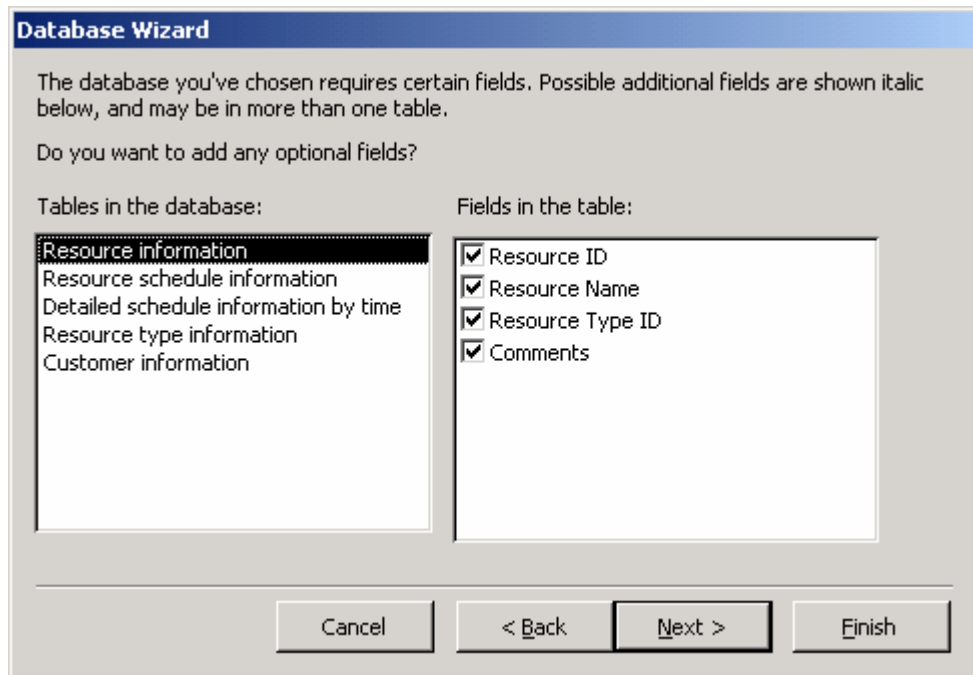


FIGURE 4-3. Selecting optional fields in the Database Wizard.

In this window, you can see the names of the tables the wizard plans to build. As you select each table name in the list on the left, the wizard shows you the fields it will include in that table in the list on the right. For many of the tables, you can have the wizard include or exclude certain optional fields (which appear in italics). In the Household Inventory application, for example, you might not be interested in keeping track of the name of the manufacturer for each inventory item.

If this is your first experience using the Database Wizard, it's a good idea to select the Yes, Include Sample Data check box in this window. If you do so, the wizard will build the database with a small amount of sample data so that you can see how the application works without having to enter any of your own data. Click Next when you finish selecting optional fields for your application.

In the next window, shown in [Figure 5-4](#), you select one of several styles for the forms in your database. The forms are objects in your database that are used to display data on your screen. Some of the styles, such as Clouds or Dusk, are quite whimsical. The Standard style has a very businesslike gray-on-gray look.

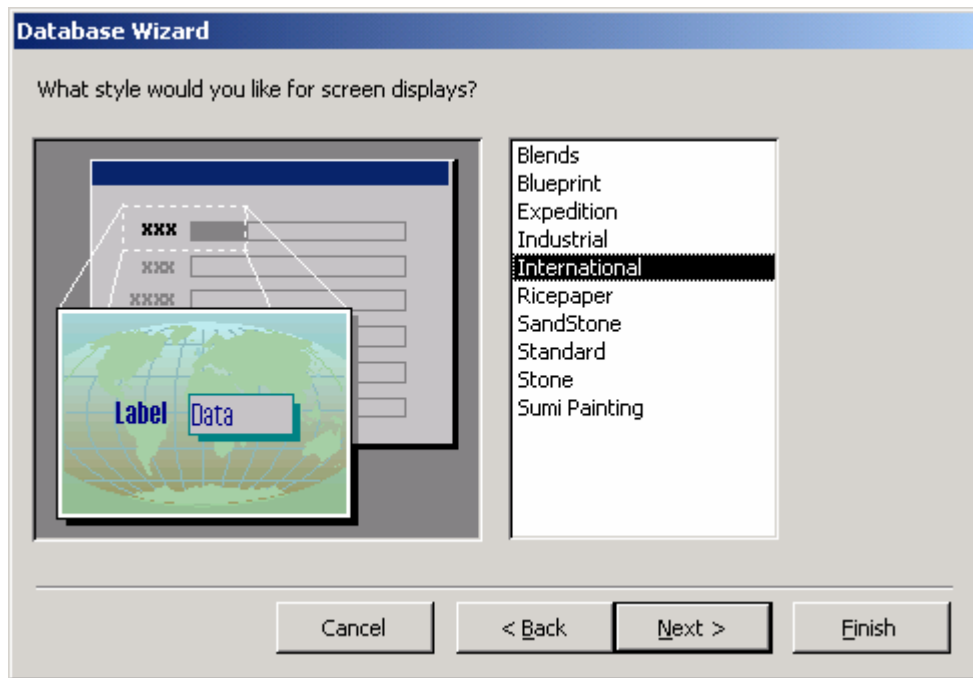


FIGURE 4-4. Selecting a style for forms in the Database Wizard.

After you select a form style, click Next to proceed to the window shown in Figure 4-5. You use this window to select a report style. You might want to select Bold, Casual, or Compact for personal applications. Corporate, Formal, and Soft Gray are good choices for business applications. Select an appropriate report style, and then click Next.

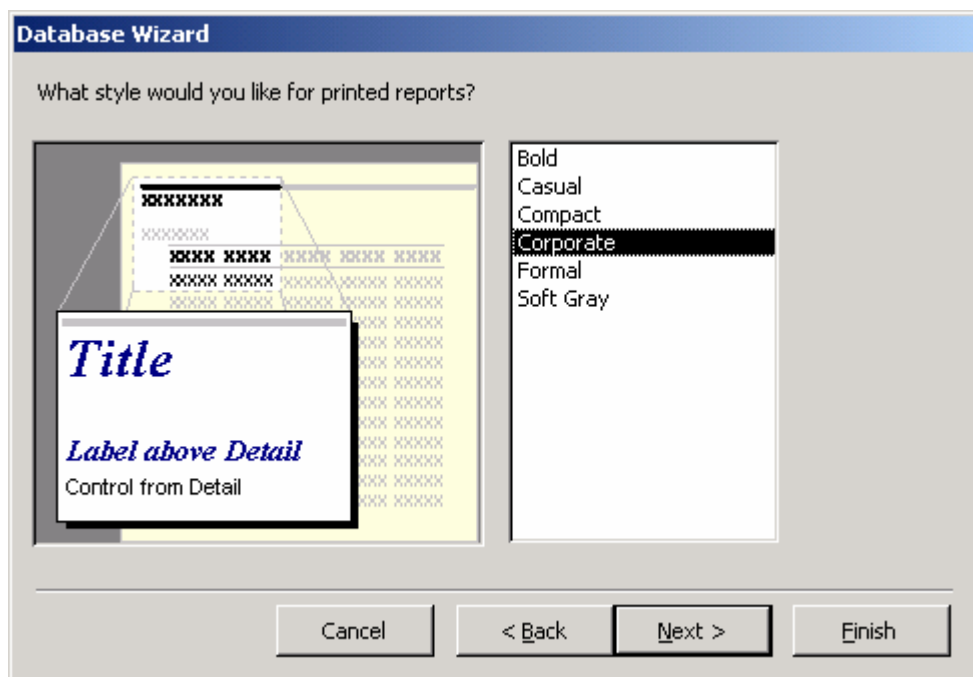


FIGURE 4-5. Selecting a report style in the Database Wizard.

In the window shown in Figure 4-6 on the next page you specify a title that will appear on the Access title bar when you run the application. You can also ask to

include a picture file in your reports. This picture file can be a bitmap, a Windows metafile, or an icon file. Click Next after you supply a title for your application.

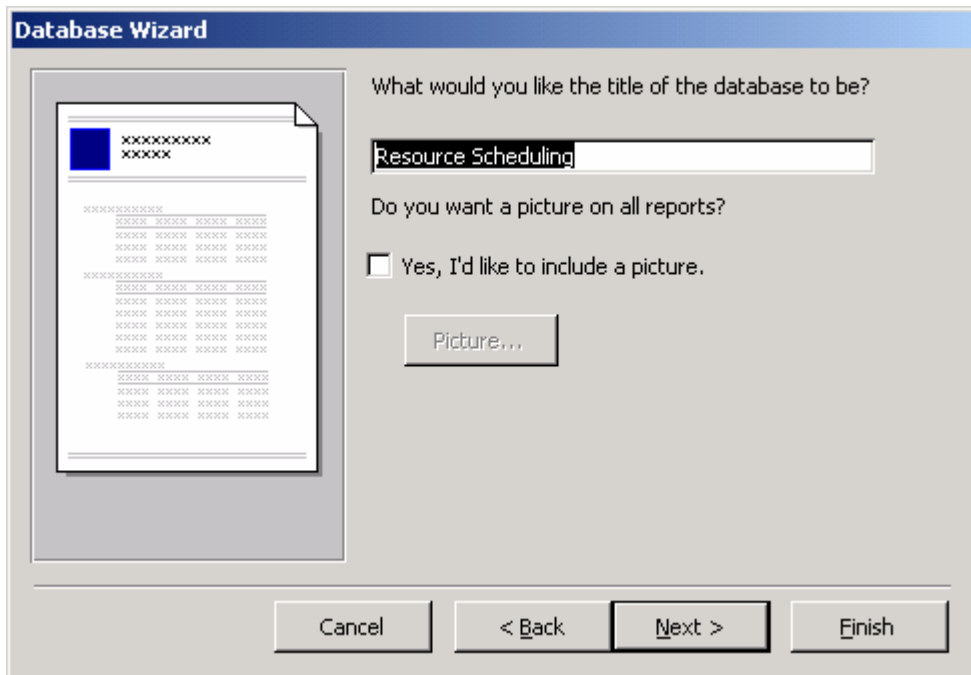
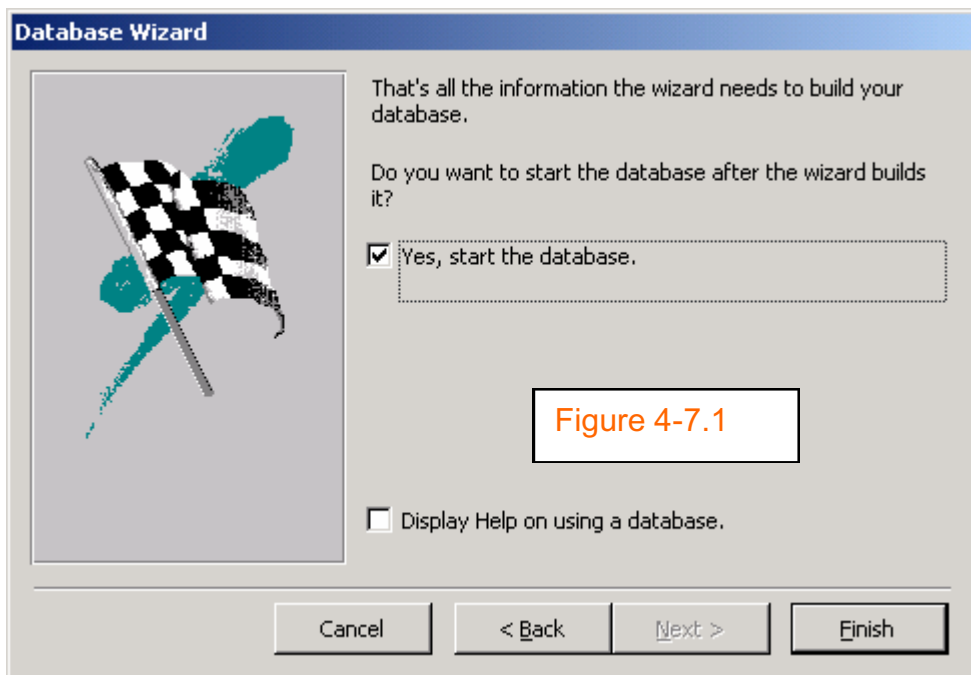


FIGURE 4-6. Naming your database in the Database Wizard.

In the final window, you can choose to start the application immediately after the wizard finishes building it. You can also choose to open a special set of help topics to guide you through using a database application. Select the Yes, Start The Database option and click Finish to create and then start your application. Figure 4-7.1 & 4-7.2 shows the opening "switchboard" form for the Resources scheduling database application.



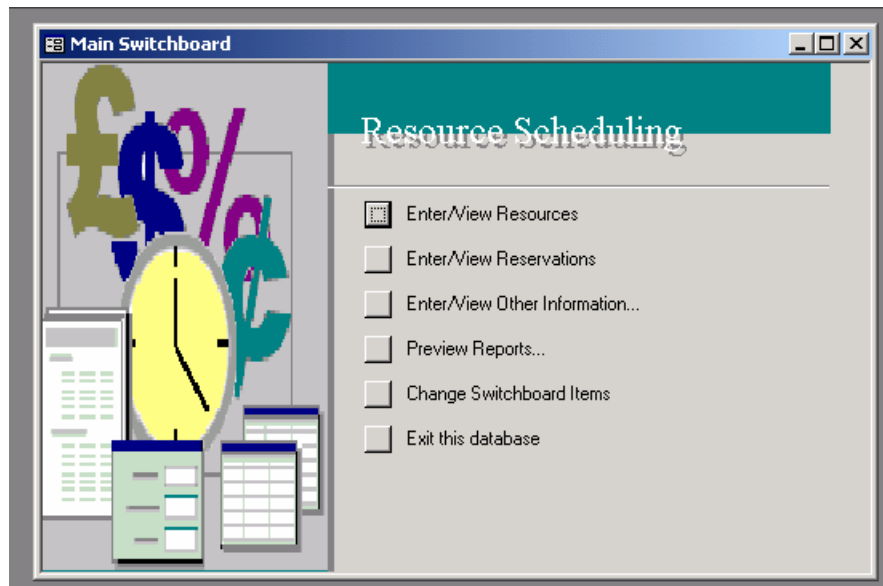


FIGURE 4-7.2.

The switchboard form for the Resources scheduling database application.

Creating a New Empty Database

To begin creating a new empty database when you start Access, select Blank Database in the opening choices dialog box shown in [Figure 4-1](#). (If you have already started Access, you can choose the New Database command from the File menu and then double-click on the Blank Database icon under the General tab in the New dialog box.) This opens the File New Database dialog box, shown in [Figure 4-8](#). Select the drive and folder you want from the Save In drop-down list. In this example, the My Documents folder of the current drive is selected. Finally, go to the File Name text box and type the name of your new database. Access appends an mdb extension to the filename for you. (Access uses a file with an mdb extension to store all your database objects, including tables, queries, forms, reports, macros, and modules.) For this example, create a new sample database named Kathy's Wedding List to experiment with one way to create a database and tables. Click the Create button to create your database.

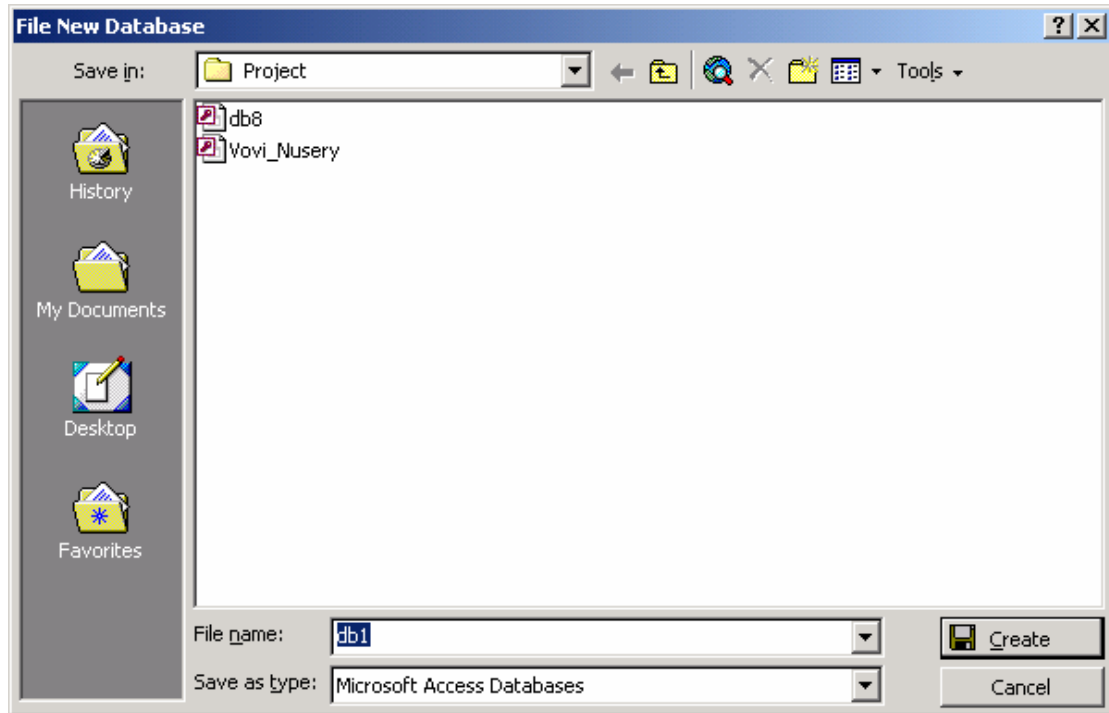


FIGURE 4-8.
The File New Database dialog box.

You can create a new database either by choosing the New Database command from the File menu or by clicking the New Database button on the toolbar. The New Database button is the first button at the left end of the toolbar.

Access takes a few moments to create the system files in which to store all the information about the tables, queries, forms, reports, macros, and modules that you might create. When Access completes this process, it displays the Database window for your new database, shown in [Figure 4-9](#) on the next page.

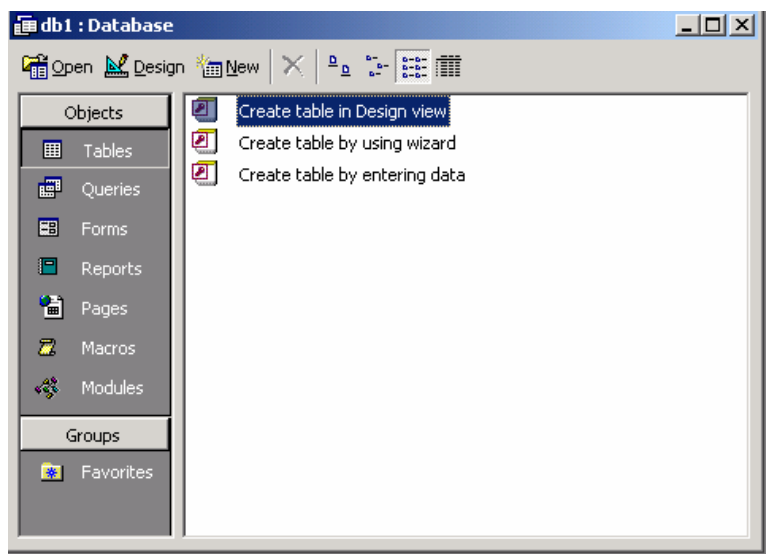


FIGURE 4-9.
The Database window for a new database.

When you open a database, Access selects the tab you last chose in the Database window for that database and shows you the available objects on that tab (unless the database includes special startup settings). Because this is a new database and no tables or startup settings exist yet, you see an empty Database window.

Tip

You can see a short description of any toolbar button by placing your mouse pointer over the button (without clicking the button) and waiting a second. Below the button, Access displays a small label, called a *ScreenTip*, that contains the name of the button. If you can't see ScreenTips, choose the Toolbars command from the View menu while you have a database open, and then choose the Customize command from the submenu. Under the Options tab of the Customize dialog box, make sure the Show ScreenTips On Toolbars check box is selected.

Using Microsoft Access Help

Access provides several ways to obtain help. You can open the Help menu, shown in Figure 4-10, and choose Microsoft Access Help to start the Office Assistant (which we'll explore in a moment) or choose Contents And Index to open a standard Windows help file. Choose What's This? to turn your mouse pointer into a question mark—you can then click the mouse on an item of interest to see a pop-up definition of the item. If you have an Internet browser on your computer, choose Microsoft On The Web to see a list of links to Web sites that provide product information and support.



FIGURE 5-10.

Choosing Microsoft Access Help from the Help menu.

Just about anywhere within Access, you can find an Office Assistant button at the far right of the toolbar. Clicking this button is the same as choosing Microsoft Access Help from the Help menu or pressing the F1 key anywhere in the product. The Office Assistant is a new feature in all the Microsoft Office 97 products. It provides "intelligent" context-sensitive access to help topics and tutorials and is linked to your choice of entertaining and informative "characters" that guide you on your search for information. (For those of you who thought Microsoft Bob was an unsuccessful product, the technology lives on in this more efficient and useful tool within the Microsoft Office products!) You probably noticed the Assistant window open the first time you started Microsoft Access, as shown in Figure 4-11.

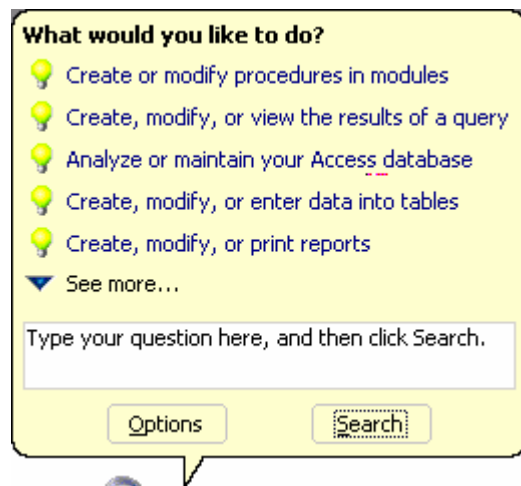


FIGURE 5-11.

You are introduced to the Office Assistant when you start Access for the first time.

Anytime the Office Assistant is open, you can click on the character with your right mouse button to open a pop-up menu that lets you set options for how the Office Assistant works. You can see the options settings for the Office Assistant in [Figure 4-12](#). [Table 4-1](#) provides a summary of what the various options can do for you.

FIGURE 4-12.

Customizing the way the Office Assistant works.

TABLE 5-1. Option settings for the Office Assistant

Option	Usage
Respond To F1 key	Select this option if you want the Office Assistant to appear instead of the normal help window when you press the F1 (help) key. Some of the property sheets in Microsoft Access do not activate the Office Assistant even if you choose this option.
Help With Wizards	Choose this option to get additional step-by-step help from the Office Assistant when you're running a wizard.
Display Alerts	When you select this option, any messages from Access or your application are displayed in the Office Assistant window if you have the window open.
Move When In The Way	Select this option to have the Office Assistant try to move out of the way when you're typing on the screen or when a message appears. With this option checked, the Office Assistant automatically disappears if more than 5 minutes elapses between uses.
Guess Help Topics	Choose this option to have the Office Assistant automatically guess what topics you might need based on what you're doing in Access. The Office Assistant also tracks

	topics you use frequently and offers those first.
Make Sounds	If you have a sound card and speakers installed, you can select this option to hear humorous (or sometimes annoying) sounds from the Office Assistant as it does its work.
Search For Both Product And Programming Help When Programming	With this option checked, the Office Assistant includes related product help topics when you invoke Help from within a programming area.
Show Tips About	Choose any of these options to include tips about using the mouse or keyboard or about working with the product more effectively when the Office Assistant suggests help topics.
Only Show High Priority Tips	Select this option to include only time-saving tips in suggested help topics.
Show The Tip Of The Day At Startup	Choose this option to see a randomly selected tip each time you start Microsoft Access.

If you don't like the standard Clippit character, you can click on the Gallery tab to choose from nearly a dozen options, as shown in [Figure 4-13](#) on the next page. Each character has a slightly different "personality." Clippit, Dot, and Power Pup can be a bit hyperactive. Will and The Genius are a bit more sober and straightforward. Mother Nature is calm and soothing. I chose a character named Scribblean origami-like paper cat. Oh, by the way, each character comes with a distinctive set of sounds and animations. Perhaps I chose Scribble because she makes my real cats crazy when she purrs and meows through my computer speakers!

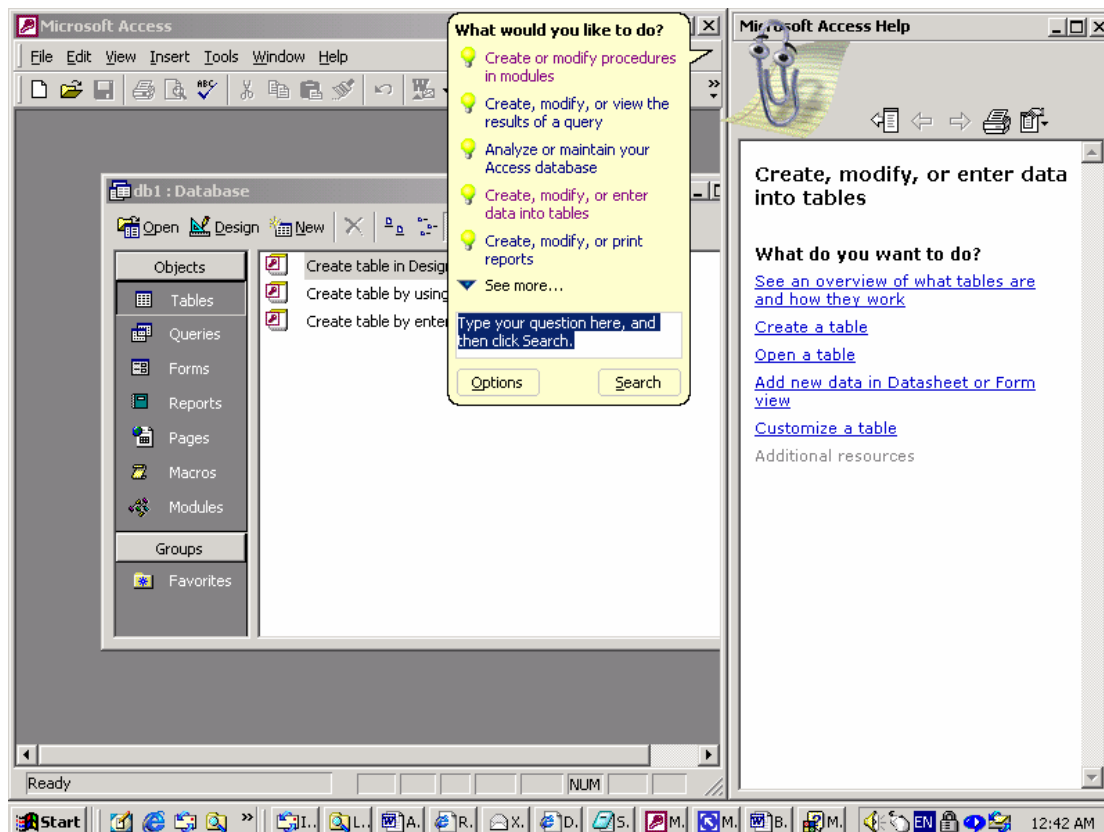


FIGURE 4-13.

Choosing your Office Assistant.

Although the Office Assistant might seem a bit whimsical at first blush, it's really a very useful (as well as entertaining) tool. In addition to providing context-sensitive help at the click of a button, it can also respond to questions you might have as you perform various tasks. Let's say you're about to create your first new table. Click on the Office Assistant button on the toolbar to open the Office Assistant together with a pop-up message box, as shown in Figure 5-14. (The Guess Help Topics option has been turned off for this figure.) If the Office Assistant is already open, left-click on the Office Assistant to open the message. You're interested in learning how to create a table in your new database, so type *Create a table* or *How do I build a new table* or some similar request in the box labeled "What Would You Like To Do?" (Notice that in this case, Scribble turns to the side and a pencil "scribbles" on the paper as you type in your request.)

You don't have to be too specific with your request, but the more "key" words that you give the assistant to work with (verbs such as "build," "create," or "define," or nouns such as "table," or "form," or "report"), the more likely that the assistant will find the most relevant help topics or tutorials. Once you have phrased a request that you think will do the trick, click on the Search button. In this case, the phrase "Create a New Table" returned at least five topics that may be related to what we want to do, as shown in Figure 4-15.



FIGURE 5-14.

Asking the Office Assistant a question.

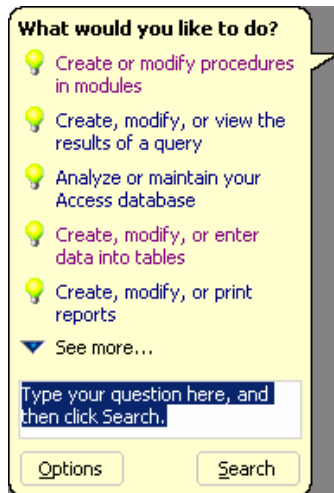


FIGURE 5-15.

The results of the Office Assistant's search.

It looks like the topic "Create a table" should do the trick. Click on that selection to see the beginning panel from the tutorial about creating tables, as shown in [Figure 5-16](#) on the next page. If you don't see a topic listed that you think is relevant to what you want to do, the Office Assistant will often offer a See More button to look at other topics. You can also rephrase your request and click Search again.

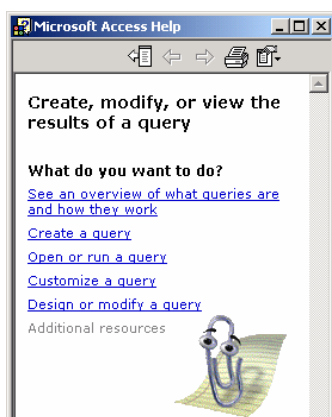


FIGURE 5-16.

The beginning panel from the Help tutorial about creating database tables.

In the next section, you'll see how to create your first new table by entering data in a datasheet.



Importing, Linking, and Exporting Data In Microsoft Access

Although you can use Microsoft Access as a self-contained database and application system, one of the primary strengths of the product is that it allows you to work with many kinds of data in other databases, in spreadsheets, or in text files. In addition to using data in your local Access database, you can *import* (copy in) or *link* (connect to) data that's in text files, spreadsheets, other Access databases, dBASE, Paradox, Microsoft FoxPro, and any other SQL database that supports the Open Database Connectivity (ODBC) software standard. You can also *export* (copy out) data from Access tables to the databases, spreadsheets, Web pages, or text files of other applications

About Open Database Connectivity

If you look under the hood of Access, you'll find that it uses a database language called *SQL (Structured Query Language)* to read, insert, update, and delete data. SQL grew out of a relational database research project conducted by IBM in the 1970s. It has been adopted as the official standard for relational databases by organizations such as the American National Standards Institute (ANSI) and the International Standards Organization (ISO). When you're viewing a Query window in Design view, you can see the SQL statements that Access uses by choosing the SQL command from the View menu or by selecting SQL View from the Query View toolbar button's drop-down list.

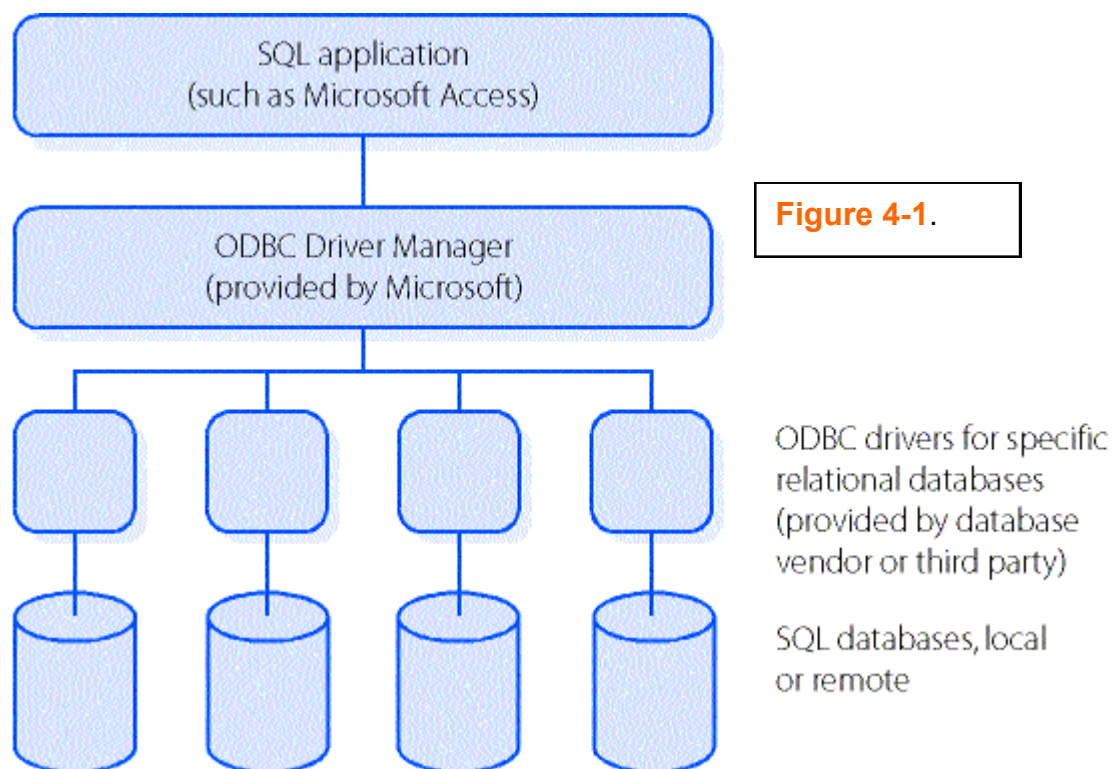
Note: In an ideal world, any product that "speaks" SQL should be able to "talk" to any other product that understands SQL. You should be able to build an application that can work with the data in several relational database management systems using the same database language. Although standards exist for SQL, most software companies have implemented variations on or extensions to the language to handle specific features of their products. Also, several products evolved before standards were well established, so the companies producing those products invented their own SQL syntax, which differs from the official standard. An SQL statement intended to be executed by *Microsoft SQL Server* might require modification before it can be executed by other databases that support SQL, such as DB2 or Oracle.

To solve this problem, several years ago a group of influential hardware and software companies—more than 30 of them, including Microsoft Corporation—formed the SQL Access Group. The group's goal was to define a common base SQL implementation that its members' products could all use to "talk" to one another. The companies jointly developed the *Common Language Interface (CLI)* for all the major variants of SQL, and they committed themselves to building CLI support into their products. About a dozen of these companies jointly demonstrated this capability in early 1992. (We will discuss about Query in next lesson)

In the meantime, Microsoft formalized the CLI for workstations and announced that Microsoft products—especially those designed for the Microsoft Windows operating

system—would use this interface to access SQL databases. Microsoft calls this formalized interface the *Open Database Connectivity (ODBC) standard*. In the spring of 1992, Microsoft announced that more than a dozen database and application software vendors had committed to providing ODBC support in their products by the end of 1992. With Access, Microsoft provides the basic ODBC driver manager and the driver to translate ODBC SQL to Microsoft SQL Server SQL. Microsoft has also worked with several database vendors to develop drivers for other databases. The ODBC architecture is represented in **Figure 4-1**.

Access was one of Microsoft's first ODBC-compliant products. You have an option to install ODBC when you install Access on your computer. Once you've added the drivers for the other SQL databases that you want to work with, you can use Access to build an application using data from any of these databases



The Microsoft ODBC architecture.

Comparison Importing with Linking Database Files

You have the choice of importing or linking data from other databases, but how do you decide which type of access is best? Here are some guidelines.

You should consider *importing* another database file when any one of the following is true:

- The file you need is relatively small and is not changed frequently by users of the other database application.

- You don't need to share the data you create with users of the other database application.
- You're replacing the old database application, and you no longer need the data in the old format.
- You need the best performance while working with the data from the other database (because Access performs best with a local copy of the data in Access's native format).

On the other hand, you should consider *linking* another database file when any one of the following is true:

- The file is larger than the maximum capacity of a local Access database (1 gigabyte).
- The file is changed frequently by users of the other database application.
- You must share the file on a network with users of the other database application.
- You'll be distributing your application to several individual users, and you might offer updates to the application interface you develop. Separating the "application" (queries, forms, reports, macros, and modules) from the "data" (tables) can make it easier to update the application without having to disturb the users' accumulated data.

Importing Data and Databases

You can copy data from a number of different file formats to create a Microsoft Access table. In addition to copying data from a number of popular database file formats, Access can also create a table from data in a spreadsheet or a text file. When you copy data from another database, Access uses information stored by the source database system to convert or name objects in the target Access table. You can import data not only from other Access databases but also from dBASE, Paradox, FoxPro, and—using ODBC—any SQL database that supports the ODBC standard.

Note: If the source Access database is secured, you must have at least read permission for the database, read data permission for the tables, and read definition permission for all other objects in order to import objects. Once you import the objects into your database, you will own the copies of those objects in the target database

Importing Spreadsheet Data

Access also allows you to import data from spreadsheet files created by Lotus 1-2-3, Lotus 1-2-3 for Windows, and Microsoft Excel versions 2 and later. You can specify a portion of a spreadsheet or the entire spreadsheet file to import into a new table or to append to an existing table. If the first row of cells contains names suitable for field names in the resulting Access table, as shown in the *Plants.xls* spreadsheet in **Figure 5-2**, you can tell Access to use these names for your fields. (This example in [Nurseryshop project](#) downloads from Weeks 04 proj1.zip)

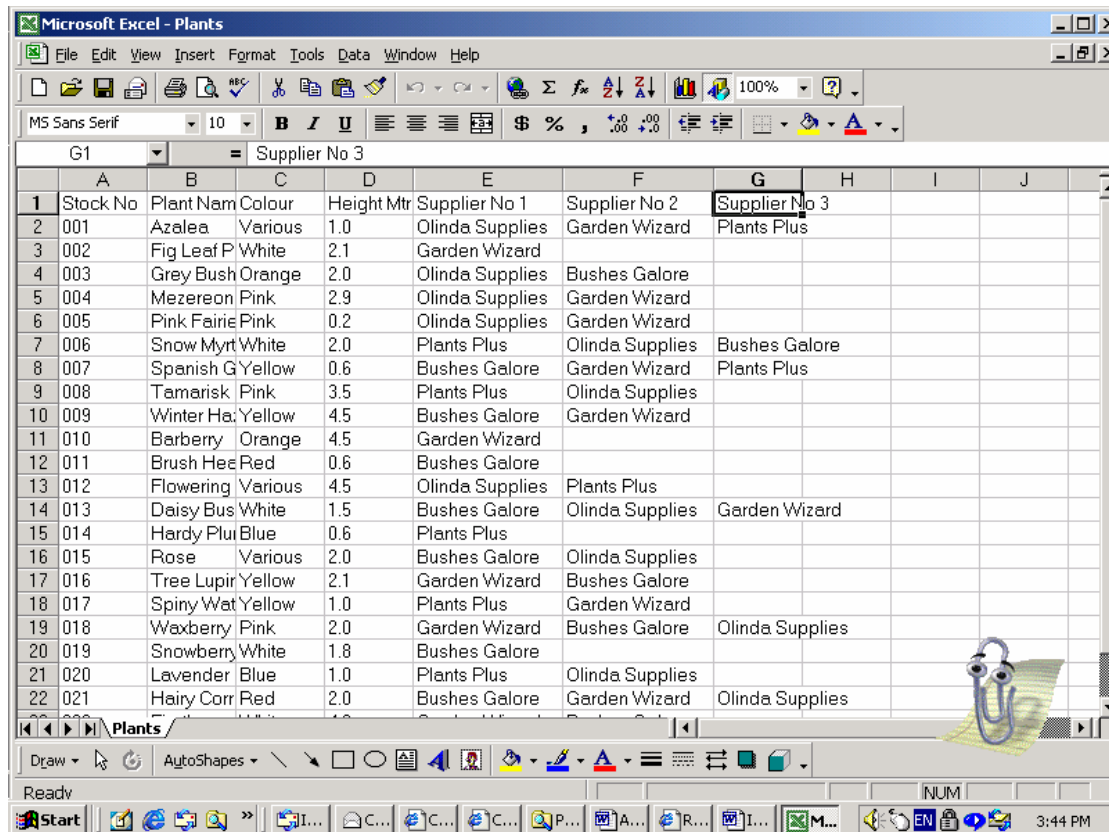


Figure 5-2 The data in the first row of this Excel spreadsheet can be used as field names when you import the spreadsheet into a new Access table (as show in demo).

Preparing a Spreadsheet

Access determines the data type for the fields in a new table based on the values it finds in the first few rows of data being imported. When you import a spreadsheet into a new table, *Access stores alphanumeric data as the Text* data type with an entry length of 255 characters, numeric data as the Number type with the Field Size property set to Double, numeric data with currency formatting as the Currency type, and any date or time data as the Date/Time type. If Access finds a mixture of data in any column in the first few rows, it imports that column as the Text data type.

Note: If you want to append all or part of a spreadsheet to a target table, you should import or link the entire spreadsheet as a new table and then use an append query to edit the data and move it to the table you want to update.

Importing a Spreadsheet

To import a spreadsheet into an Access database, do the following:

1. **Open the Access database** that will receive the spreadsheet. If that database is already open, switch to the Database window.
2. Choose the **Get External Data** command from the File menu, and then choose Import from the submenu. Access opens the Import dialog box, as shown

Figure 5-3.

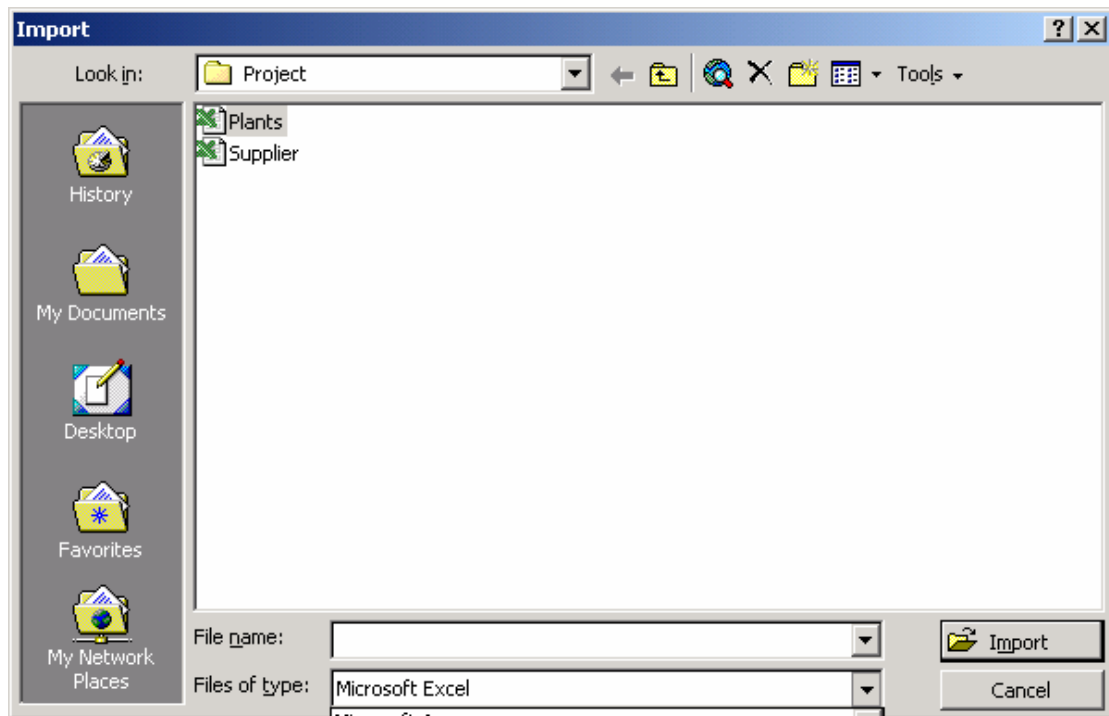


Figure 5-3.

3. Select the type of spreadsheet you want to import (Excel or Lotus 1-2-3) in the **Files Of Type** drop-down list. Select the folder and the name of the spreadsheet file that you want to import. If you want to follow along with this example, select the Colleges.xls file on the sample disc.
4. Click the **Import button**. If your spreadsheet is from Excel version 5.0 or later, it can contain multiple worksheets. If the spreadsheet contains multiple worksheets or any named ranges, Access shows you the first window of the Import Spreadsheet Wizard, as shown in the following illustration. (If you want to import a range that isn't yet defined, exit the wizard, open your spreadsheet to define a name for the range you want, save the spreadsheet, and then restart the import process in Access.) Select the worksheet or the named range that you want to import, and click Next to continue.
5. After you select a worksheet or a named range, or if your spreadsheet file contains only a single worksheet, the wizard displays the window shown on the following **Figure 5-4**.

Select the First Row Contains Column Headings check box if you've placed names at the tops of the columns in your spreadsheet. Click Next to go to the next step. In the window that appears, you can specify whether you want to import the data to a new table or append it to an existing one. Click Next to go to the next step.

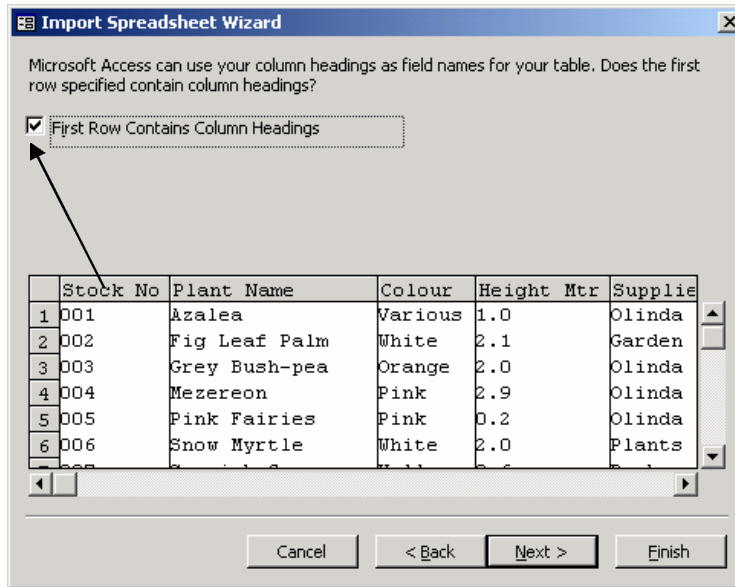


Figure 5-4

6. If you choose to create a new table, you can scroll left and right to the various fields and tell the wizard which fields should be indexed in the new table. Your indexing choices are identical to the ones you'll find for the Indexed property of a table in Design view. In this case, for the CEEB field, select Yes (No Duplicates) from the Indexed drop-down list box, as shown here, and for the Zip field select Yes (Duplicates OK). **Figure 5-5**

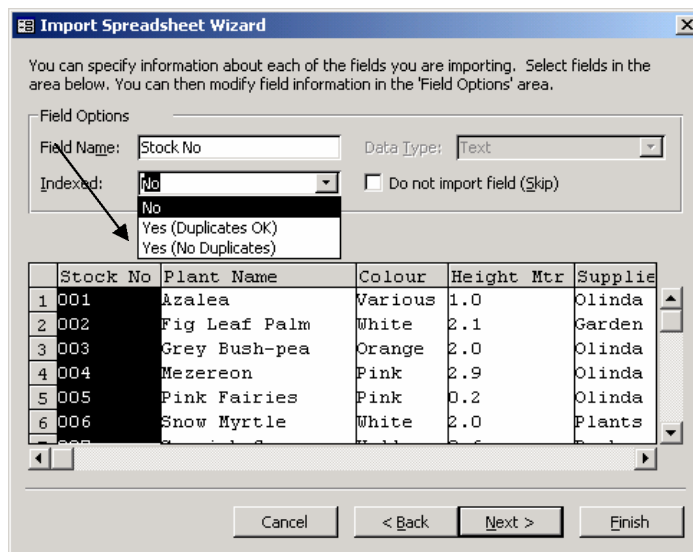
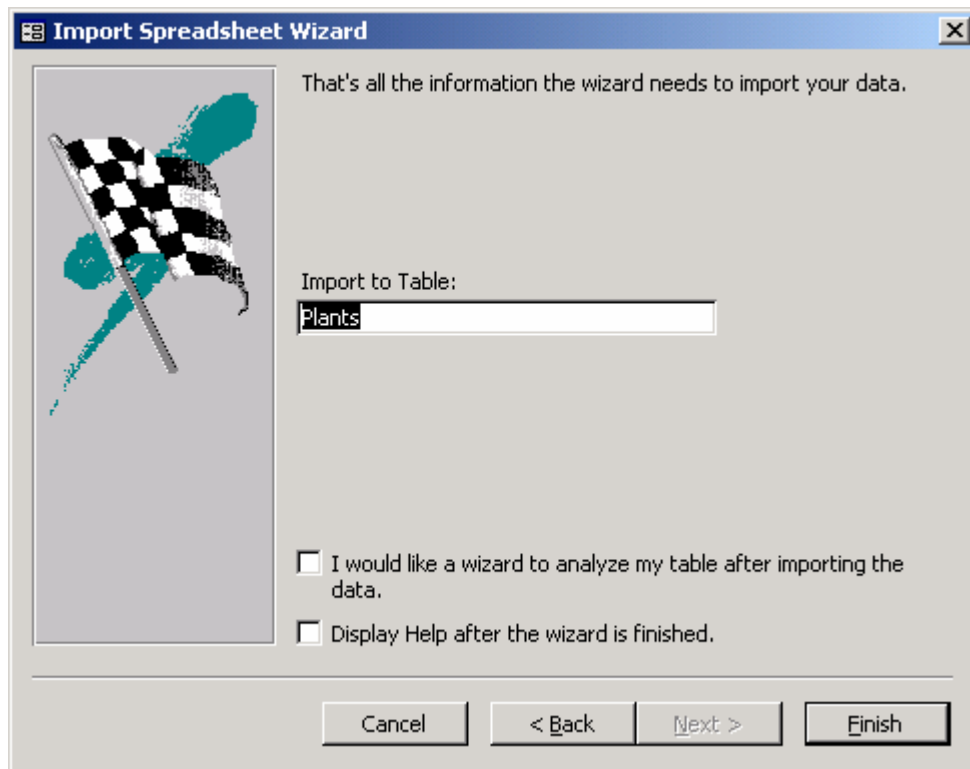


Figure 5-5

7. Click Next to go to the final window. In the final window, you can type in the name of your new table and select an option to start the Table Analyse Wizard to analyse your new table (If you enter the name of an existing table, Access asks if you want to replace the old table).
8. Click **Finish** in the last window to import your data. Access opens a message box that indicates the result of the import procedure. If the procedure is successful, the new table will have the name of the spreadsheet you selected. If you asked to append the data to an existing table and Access found errors, you can choose to complete the import with errors or go back to the Wizard to attempt to fix the problem (such as incorrectly defined columns). You may

need to exit the Wizard and correct data in the original spreadsheet file as noted in the following section. See **Figure 5-5**



Figure

5-5

Earlier in this lesson, in the section titled "[Preparing a Spreadsheet.](#)" you learned that Access determines data types for the fields in a new table based on the values it finds in the first several rows being imported from a spreadsheet. Figures 10-3 and 10-4, shown earlier, show a spreadsheet whose first few rows would generate a wrong data type for the Zip column in a new Access table. The Number data type that Access would generate for that field, based on the first several entries, would not work for all the remaining entries, some of which have hyphens in them. In addition, one of the rows doesn't have a value in the CEEB column. If you attempt to use this column as the primary key when you import the spreadsheet, you'll get an additional error.

If you were to import that spreadsheet, Access would first display an error message. This indicates that the wizard found a problem with the column that you designated as the primary key. If you have duplicate values, the wizard will also inform you. When the wizard encounters any problems with the primary key column, it imports your data but does not define a primary key. This gives you a chance to correct the data in the table and then define the primary key yourself.

In addition, if the wizard has any problems with data conversion, it displays a message box to warn you that it's ok. You can correct some of the errors in the Table window in Design view. You can then set the primary key in Design view.

Importing Text Files

You can import data from a text file into Microsoft Access even though, unlike the data in a spreadsheet, the data in a text file isn't arranged in columns and rows in an orderly way. You make the data in a text file understandable to Access either by creating a *delimited text file*, in which special characters delimit the fields in each record, or by creating a *fixed-width text file*, in which each field occupies the same location in each record.

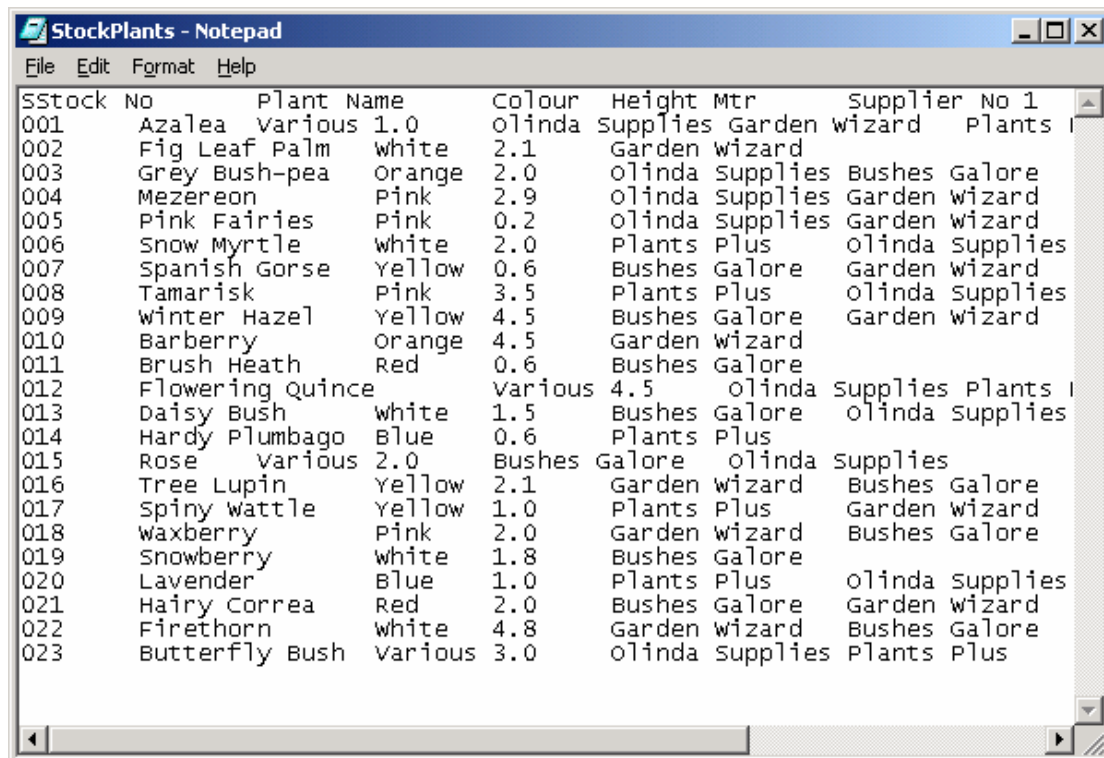
Preparing a Text File

You might be able to import some text files into Access without changing them, particularly if a program using standard field delimiters created a text file. In many cases, you'll have to modify the contents of the file, define the file for Access, or do both before you can import it.

Setting Up Delimited Data

Access needs some way to distinguish where a field starts and ends in each incoming text string. Access supports three standard separator characters: a comma, a tab, and a space. When you use a comma as the separator (a very common technique), the comma (or the carriage return at the end of the record) indicates the end of each field, and the next field begins with the first nonblank character. The commas are not part of the data. To include a comma within a text string as data, you must enclose all text strings within single or double quotation marks. If any of your text strings contain double quotation marks, you must enclose the strings within single quotation marks, and vice versa. Access accepts only single or double quotation marks (but not both) as the text delimiter, so all embedded quotes in a file that you want to import into Access must be of the same type. In other words, you can't include a single quotation mark in one field and a double quotation mark in another field within the same file. **The Demo** and practice in class will show you a sample tab-separated of plants.txt and comma-delimited in Cust2 text file.

By default, Access assumes that commas separate fields in a delimited text file and that text strings are within double quotation marks. As you saw demo in class last week, if you want to import a file that is delimited differently, you can specify different delimiters and separators in the Text Import Wizard. The important thing to remember is that your data should have a consistent data type in all the rows for each column—just as it should in spreadsheet files. If your text file is delimited, the delimiters must be consistent throughout the file. See below: **Figure 5-6**



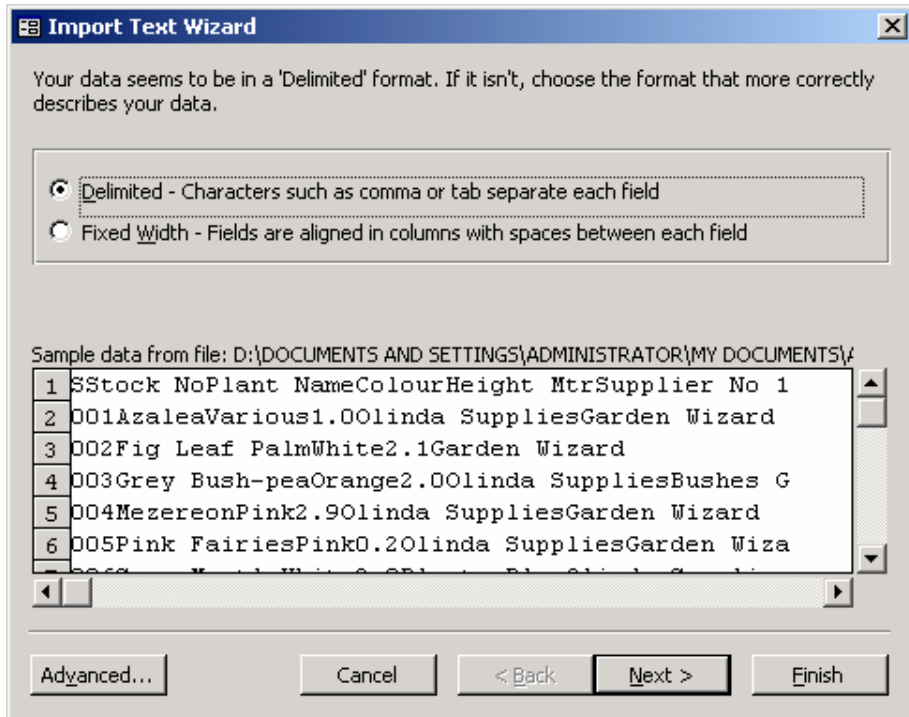
SStock No	Plant Name	Colour	Height Mtr	Supplier	No 1
001	Azalea	Various	1.0	Olinda Supplies	Garden wizard
002	Fig Leaf Palm	white	2.1	Garden wizard	Plants
003	Grey Bush-pea	Orange	2.0	Olinda Supplies	Bushes Galore
004	Mezereon	Pink	2.9	Olinda Supplies	Garden wizard
005	Pink Fairies	Pink	0.2	Olinda Supplies	Garden wizard
006	Snow Myrtle	white	2.0	Plants Plus	Olinda Supplies
007	Spanish Gorse	Yellow	0.6	Bushes Galore	Garden wizard
008	Tamarisk	Pink	3.5	Plants Plus	Olinda Supplies
009	Winter Hazel	Yellow	4.5	Bushes Galore	Garden wizard
010	Barberry	Orange	4.5	Garden wizard	
011	Brush Heath	Red	0.6	Bushes Galore	
012	Flowering Quince	Various	4.5	Olinda Supplies	Plants
013	Daisy Bush	white	1.5	Bushes Galore	Olinda Supplies
014	Hardy Plumbago	Blue	0.6	Plants Plus	
015	Rose	Various	2.0	Bushes Galore	Olinda Supplies
016	Tree Lupin	Yellow	2.1	Garden wizard	Bushes Galore
017	Spiny wattle	Yellow	1.0	Plants Plus	Garden wizard
018	waxberry	Pink	2.0	Garden wizard	Bushes Galore
019	snowberry	white	1.8	Bushes Galore	
020	Lavender	Blue	1.0	Plants Plus	Olinda Supplies
021	Hairy Correa	Red	2.0	Bushes Galore	Garden wizard
022	Firethorn	white	4.8	Garden wizard	Bushes Galore
023	Butterfly Bush	Various	3.0	Olinda Supplies	Plants Plus

Importing a Text File

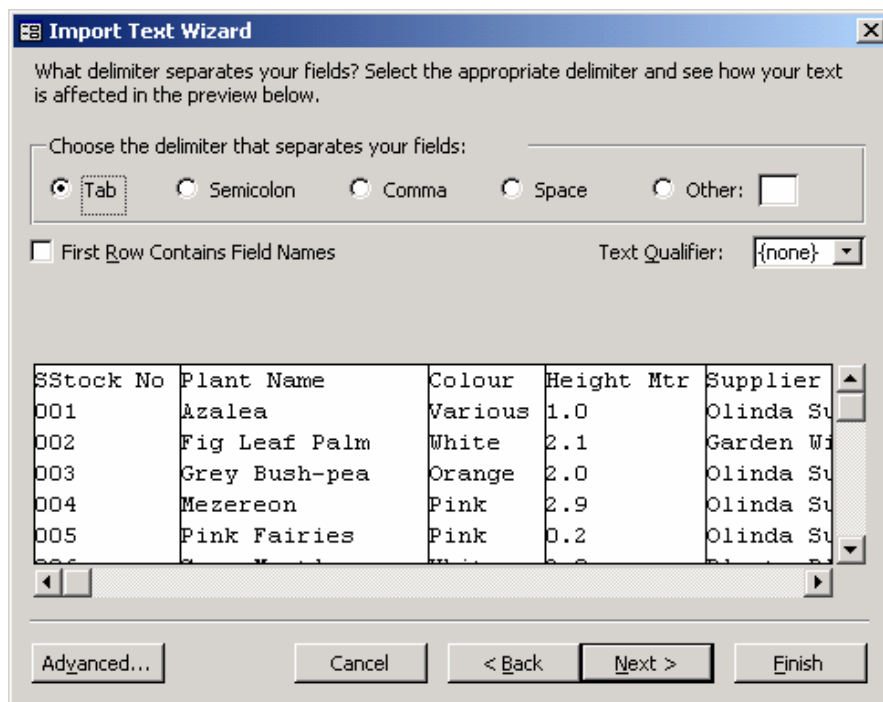
Before you can import a text file, you'll probably need to prepare the data or define the file for Access, or both, as discussed earlier in the section titled "[Preparing a Text File.](#)" After you do that, you can import the text file into an Access database by doing the following:

1. Open the Access database that will receive the text data. If that database is already open, switch to the Database window.
2. Choose the Get External Data command from the File menu, and then choose Import from the submenu. Access opens the Import dialog box, as shown earlier
3. Select Text Files in the Files Of Type drop-down list, and then select the folder and the name of the file you want to import. Access starts the Text Import Wizard and displays the first window of the wizard, as shown on the following page.

1.

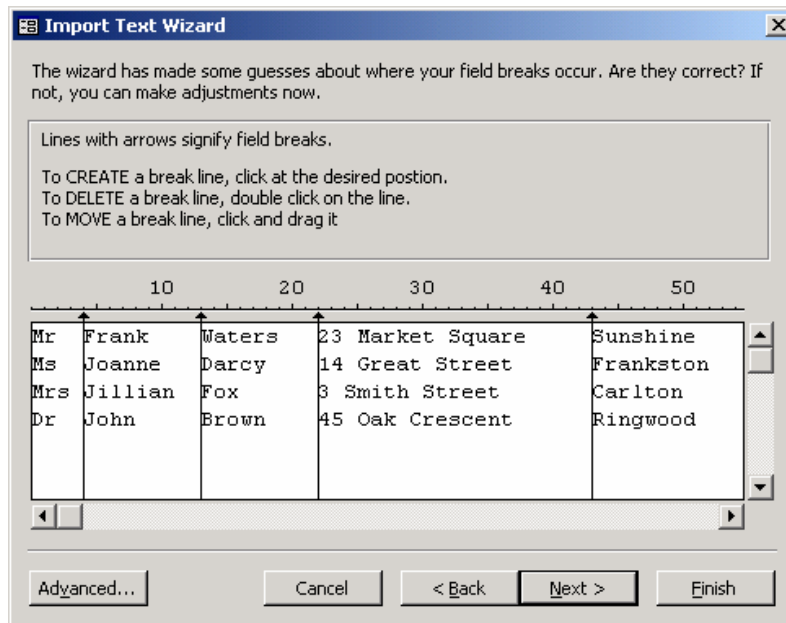


4. In this window, the wizard makes its best guess about whether the data is delimited or fixed-width. It displays the first several rows of data, which you can examine to confirm the wizard's choice. If the wizard has made the wrong choice, your data is probably formatted incorrectly. You should exit the wizard and fix the source file as suggested in the section "[Preparing a Text File.](#)" If the wizard has made the correct choice, click Next to go to the next step.
5. If your file is delimited, the Text Import Wizard displays the window shown in the following illustration.



Here you can verify the character that delimits the fields in your text file and the qualifier character that surrounds text strings. Remember that usually when you save a delimited text file from a spreadsheet program, the field delimiter is a tab character and you'll find quotation marks around only strings that contain commas. If the wizard doesn't find a text field with quotation marks in the first line, it might assume that no text is surrounded by quotes. You might need to change the Text Qualifier from *{none}* to " if this is the case.

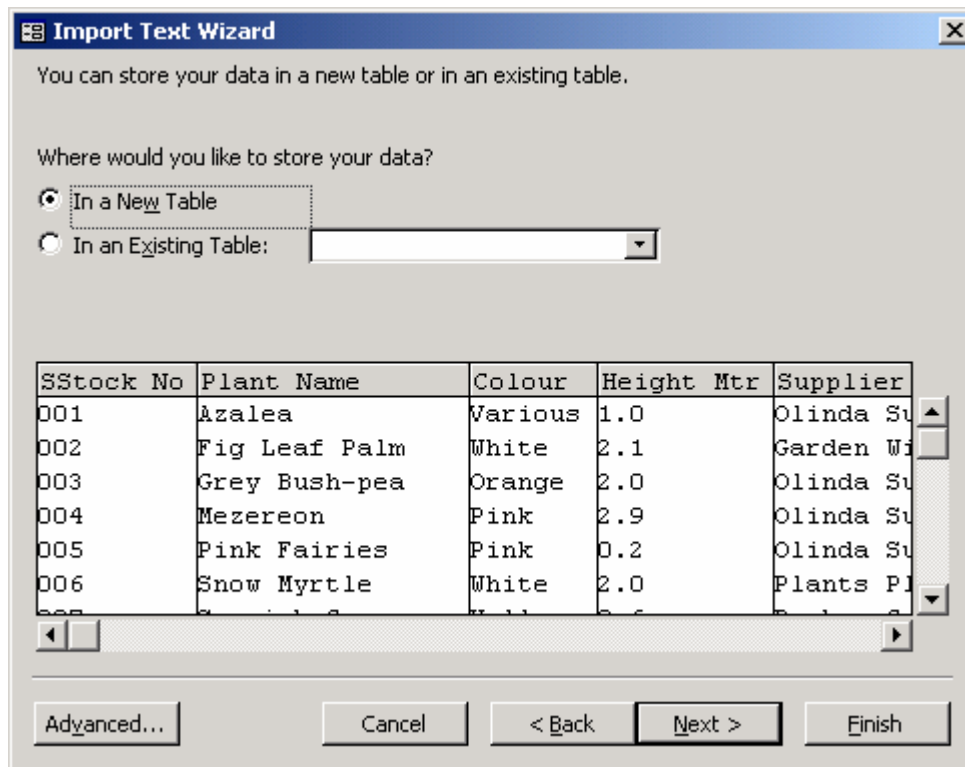
If your file is in fixed-width format, the wizard displays the window shown in the following illustration.



Instead of showing delimiting characters, the wizard offers a graphic representation of where it thinks each field begins. To change the definition of a field, you can click on and drag any line to move it. You can also create an additional field by clicking at the position on the display where fields should be separated. If the wizard creates too many fields, you can double-click on any extra delimiting lines to remove them. In the example shown above (Using Cust1.txt file downloads from Proj1), the wizard assumes that the **first name** is separate from the rest of the **Last name**. You can double-click on the line following the area code to remove it and group all the characters that make up the field **Full name** into a single field.

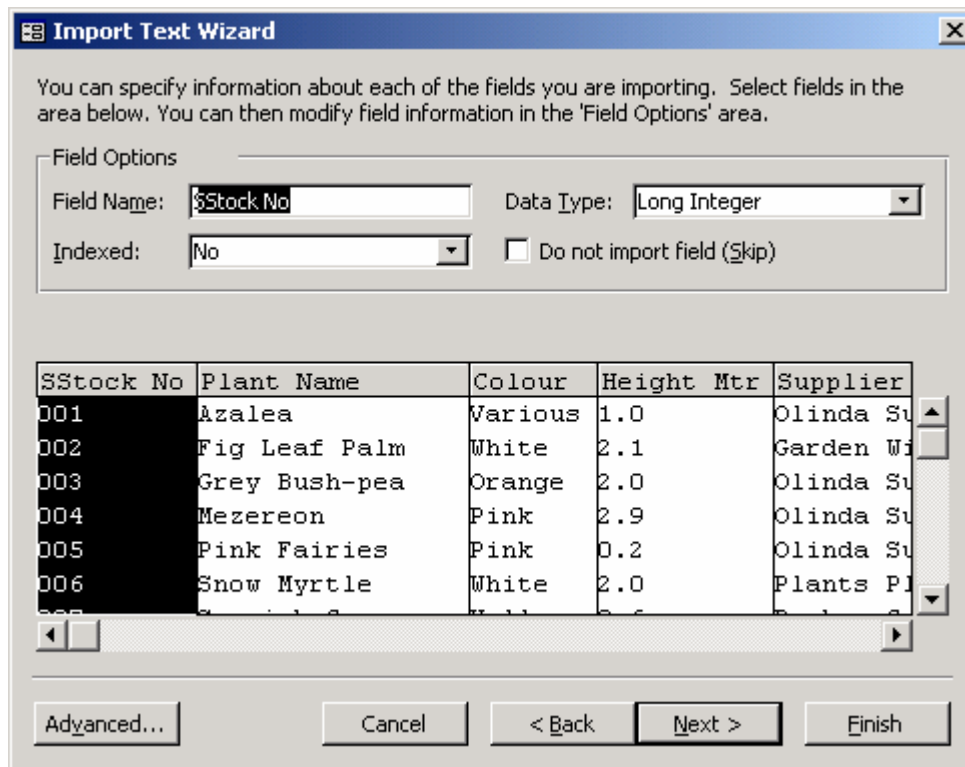
After you finish in this window, click Next to go to the next step.

2. In the next window, shown below, you specify whether you want to import the text into a new table or append the data to an existing table.

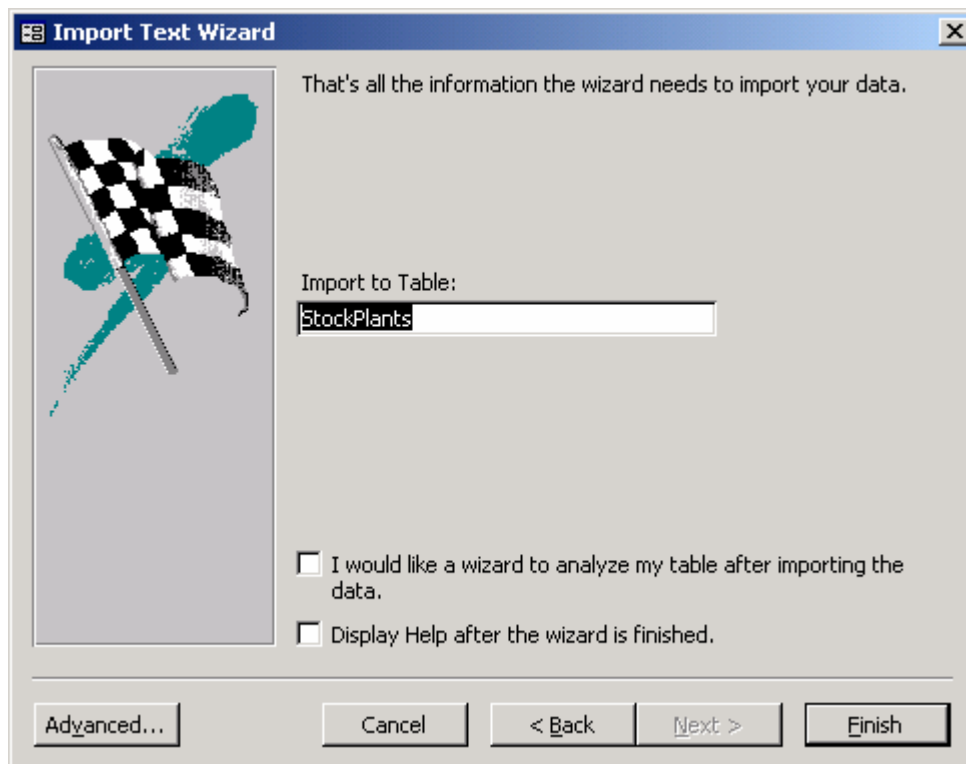


If you decide to create a new table, the wizard displays the window shown on the facing page. Here you can specify or confirm field names (you can change field names even if the first row in the text file contains names), select field data types, and set indexed properties. Click Next to go to the next window, where you can select a primary key, much as you did for spreadsheet files.

If you decide to append the data to an existing table, either the columns must exactly match the columns in the target table (left to right) or the file must be a delimited file that has column names in the first row that match column names in the target table.



3. In the final window of the wizard, you confirm the name of the new table or the target table and click Finish to import your data. Access displays a confirmation message box to show you the result of the import procedure. If the wizard encounters an error that prevents any data from being imported, it will reopen the final wizard window. You can click the Back button to return to previous settings to correct them.



Fixing Errors

While importing text files, you might encounter errors that are similar to those described earlier in the section titled "[Importing Spreadsheet Data.](#)" When you append a text file to an existing table, some rows might be rejected because of duplicate primary keys. Unless the primary key for your table is an AutoNumber field, the rows you append from the text file must contain the primary key fields and the values in those fields must be unique. For delimited text files, Access determines the data type based on the fields in the first several records being imported. If a number appears in a field in the first several records but subsequent records contain text data, you must enclose that field in quotation marks in at least one of the first few rows so that Access will use the Text data type for that field. If a number first appears without decimal places, Access will use the Number data type with the Field Size property set to Long Integer. This setting will generate errors later if the numbers in other records contain decimal places.

Defining an Import Specification

If you are likely to import the same fixed-width file often or if you want to be able to use a macro or a Visual Basic for Applications (VBA) procedure to automate importing a text file, you can use the Text Import Wizard to save an import specification for use by your automation procedures. To do so, use the wizard to examine your file, and verify that the wizard identifies the correct fields. Click the Advanced button to see an Import Specification window like the one shown here.

	Field Name	Data Type	Indexed	Skip
▶	Field1	Long Integer	No	<input type="checkbox"/>
	Field2	Text	No	<input type="checkbox"/>
	Field3	Text	No	<input type="checkbox"/>
	Field4	Double	No	<input type="checkbox"/>
	Field5	Text	No	<input type="checkbox"/>
	Field6	Text	No	<input type="checkbox"/>
	Field7	Text	No	<input type="checkbox"/>
*				<input checked="" type="checkbox"/>

For fixed-width specifications, you can define the field names, data types, start column, width, and indexed properties. You can also specify in the File Origin combo box whether the text file was created using a program running under MS-DOS or under Microsoft Windows. For fixed-width files, you don't need to make a Field Delimiter selection or a Text Qualifier selection; you use these options only to define a specification for delimited files. You can also specify the way Access recognizes date and time values and numeric fractions. Click the Save As button to save your specification and give it a name. You can also click the Specs button to edit other previously saved specifications.

Linking Text and Spreadsheet Files

Linking a text file or an Excel spreadsheet file is almost identical to importing these types of files, as discussed earlier in this chapter. As noted earlier, you can only read linked text files, but you can update and add new rows (but not delete rows) in Excel spreadsheet files.

To link a spreadsheet file or a text file, do the following:

1. Open the Access database to which you want to link the file. If that database is already open, switch to the Database window.
2. Choose the Get External Data command from the File menu, and then choose Link Tables from the submenu. Access opens the Link dialog box, which lists the types of files you can link.
3. Select Microsoft Excel or Text Files, as appropriate, in the Files Of Type drop-down list, and then select the folder and the name of the file that you want to link. If you're connecting over a network, select the logical drive that is assigned to the network server that contains the database you want. If you want Access to automatically connect to the network server each time you open the linked file, type the full network location in the File Name edit box instead of choosing a logical drive. For example, on a Windows NT network you might enter a network location such as this one:

(Driver name):\\FILESVR\EXCEL\SHARED\PLANTS.XLS

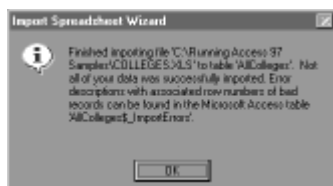
4. Click the Link button to start the Link Spreadsheet Wizard or the Link Text Wizard.
5. Follow the steps in the wizard, which are identical to the steps for importing a spreadsheet or text file, as described earlier in this chapter.

Linking SQL Tables

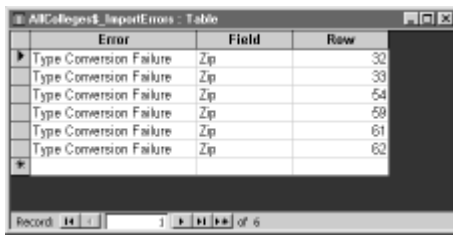
To link a table from another database system that supports ODBC SQL, you must have the ODBC driver for that database installed on your computer. (For details, see the *Building Applications with Microsoft Access 97* manual that comes with Access; also see the appendix to this book.) Your computer must also be linked to the network that connects to the SQL server you want, and you must have an account on that server. Check with your system administrator for information about correctly connecting to the SQL server from which you want to link a table.

To link an SQL table, do the following:

1. Open the Access database to which you want to link the SQL table. If that database is already open, switch to the Database window.
2. Choose the Get External Data command from the File menu, and then choose Link Tables from the submenu. Access opens the Link dialog box, which lists the types of files you can link.
3. Select ODBC Databases in the Files Of Type drop-down list. Access opens the Select Data Sources dialog box, shown earlier on page 319, in which you can select the data source that maps to the SQL server containing the table you want to link. Select a data source and click OK. Access displays the SQL Server Login dialog box for the SQL data source that you selected.



4. Enter your user ID and your password, and click OK. If you are authorized to connect to more than one database on the server and you want to connect to a database other than your default database, enter your user ID and password, and then click the Options button to open the lower part of the dialog box. When you click the Database text box, Access logs on to the server and returns a list of available database names. Select the one you want, and click OK. If you don't specify a database name and if multiple databases exist on the server, Access will prompt you to select the database you want. When Access connects to the server, you'll see the Link Objects dialog box, similar to the Import Objects dialog box shown earlier on page 321, which lists the available tables on that server.



Error	Field	Row
Type Conversion Failure	Zip	30
Type Conversion Failure	Zip	39
Type Conversion Failure	Zip	54
Type Conversion Failure	Zip	59
Type Conversion Failure	Zip	61
Type Conversion Failure	Zip	62

5. From the list of tables, select the ones you want to link. If you select a table name in error, you can click it again to deselect it, or you can click the Deselect All button to start over. Click the OK button to link to the SQL tables you selected.
6. If the link procedure is successful, the new table will have the name of the SQL table. If Access finds a duplicate name, it will generate a new name by adding a unique integer to the end of the name. For example, if you link to a table named Newcollege and you already have tables named Newcollege and Newcollege1, Access creates a table named Newcollege2

Modifying Linked Tables

You can make some changes to the definitions of linked tables to customize them for use in your Access environment. When you attempt to open the Table window in Design view, Access opens a dialog box to warn you that you cannot modify certain properties of a linked table. You can still click OK to open the linked table in Design view.

You can open a linked table in Design view to change the Format, Decimal Places, Caption, Description, and Input Mask property settings for any field. You can set these properties to customize the way you look at and update data in Access forms and reports. You can also give any linked table a new name for use within your Access database (although the table's original name remains unchanged in the source database) to help you better identify the table or to enable you to use the table with the queries, forms, and reports that you've already designed.

Changing a table's design in Access has no effect on the original table in its source database. Notice, however, that if the design of the table in the source database changes, you must relink the table to Access. You must also unlink and relink any table if your user ID or your password changes.

Unlinking Linked Tables

It is easy to unlink tables that are linked to your Access database. In the Database window, simply select the table you want to unlink and then press the Del key or choose Delete from the Edit menu. Access displays the confirmation message box. Click the Yes button to unlink the table. Unlinking the table does not delete the table; it simply removes the link from your table list in the Database window.

Adding Power with Select Queries

In the previous lesson, you learned about working with the data in your tables in Datasheet view. Although you can do a lot with datasheets—including browsing, sorting, filtering, updating, and printing your data—you'll find that you often need to perform calculations on your data or retrieve related data from multiple tables. To select a set of data to work with, you use queries.

When you define and run a *select query* (which selects information from the tables and queries in your database, as opposed to an *action query*, which inserts, updates, or deletes data), Microsoft Access creates a *recordset* of the selected data. In most cases, you can work with a recordset in the same way that you work with a table: you can browse through it, select information from it, print it, and even update the data in it. But unlike a real table, a recordset doesn't actually exist in your database. Access creates a recordset from the data in both your tables and your query at the time you run the query.

As you learn to design forms and reports later in this course, you'll find that queries are the best way to focus on the specific data you need for the task at hand. You'll also find that queries are useful for providing choices for combo and list boxes, which make entering data in your database much easier.

Note: The examples in this lesson are based on the tables and data from the Nursery Project sample database (NuseryShop.mdb)

To open a new Query window in Design view, click on the Queries tab in the Database window, and then click the New button to the right of the Query list. A dialog box opens that lets you either start a new query from scratch in Design view or select a Query Wizard. (You'll learn about Query Wizards later in this lesson.) To open an existing query in Design view, click on the Queries tab in the Database window (which, in this case, displays the Query list of the Entertainment Schedule database, as shown in [Figure 6-1](#)), select the query you want, and click the Design button.

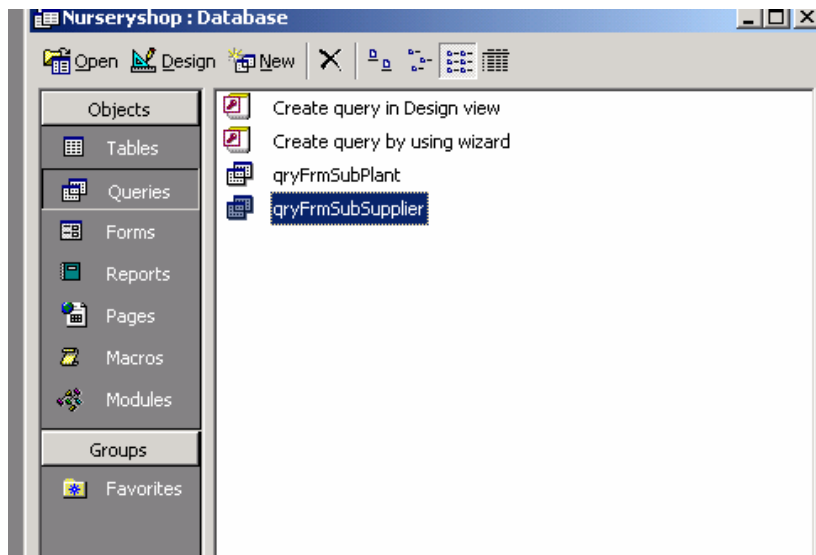


FIGURE 6-1. Opening a Query window in Design view from the Database window.

Figure 6-2 shows a query whose window has been opened in Design view. The upper part of the Query window contains field lists, and the lower part contains the design grid.

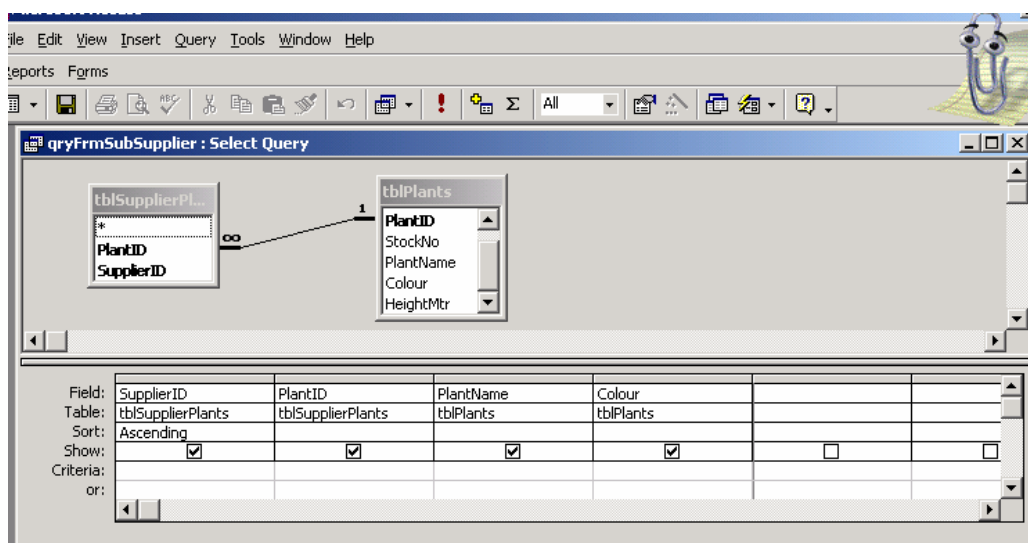


FIGURE 6-2. A query open in Design view.

<21>Selecting Data from a Single Table

One advantage of using queries is that they allow you to find data easily in multiple related tables. Queries are also useful, however, for sifting through the data in a single table. All the techniques you use for working with a single table apply equally to more complex multiple-table queries, remember the lesson last week begins by using queries to select data from a single table.

The easiest way to start building a query on a table is to open the Database window, select the table you want, and select Query from the New Object toolbar button's drop-down list. Do this now with the tblPlants table in the Nursery shop database, and

then select Design View in the resulting New Query dialog box. Click OK to open the window shown in **Figure 6-3**. If you can't see the Table row in the lower part of the Query window, choose the Table Names command from the View menu.

Note The New Object button "remembers" the last new object type that you created. If you've created only tables up to this point, you have to use the button's drop-down list to select New Query. Once you start a new query in this manner, the New Object button defaults to New Query until you use the button's drop-down list to create a different type of object.

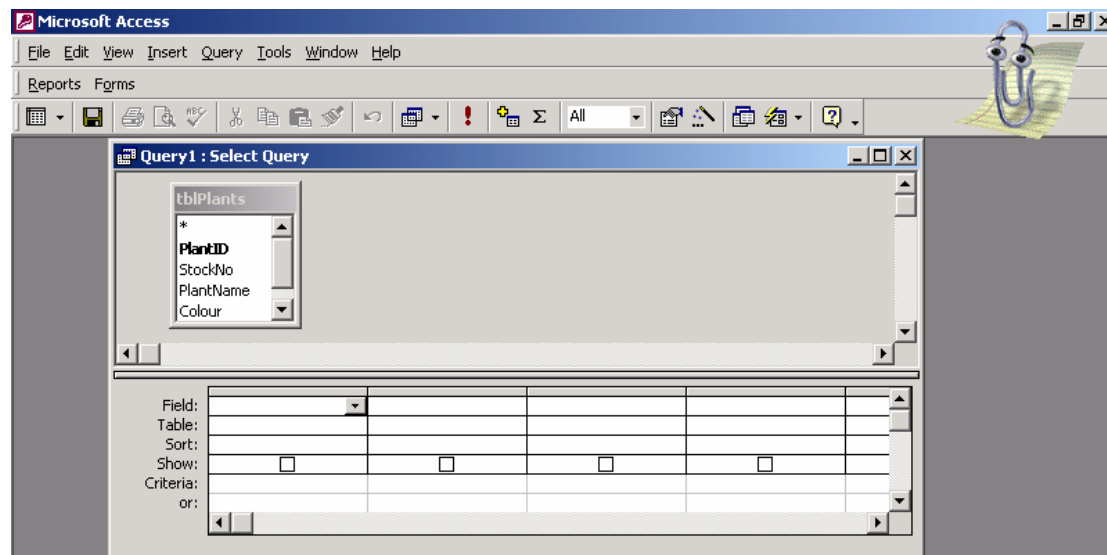


FIGURE 6-3. The Query window in Design view for a new query on tblPLants.

As mentioned earlier, the Query window in Design view has two main parts. In the upper part are field lists with the fields for the tables or queries you chose for this query. The lower part of the window is the design grid, in which you do all the design work. Each column in the grid represents one field that you'll work with in this query. As you'll see later, a field can be a simple field from one of the tables, a calculated field based on several fields in the tables, or a total field using one of the functions provided by Microsoft Access.

You use the first row of the design grid to select fields—the fields you want in the resulting recordset, the fields you want to sort, and the fields you want to test for values. As you'll learn later, you can also generate custom field names (for display in the resulting recordset), and you can use complex expressions or calculations to generate a calculated field.

Because you chose the Table Names command from the View menu, Access displays the table name (which is the source of the selected field) in the second row of the design grid. In the Sort row, you can specify whether Access should sort the selected or calculated field in ascending or in descending order.

Tip: It's a good idea to select the Show Table Names option on the Tables/Queries tab of the Options dialog box (choose the Options command from the Tools menu) whenever your query is based on more than one table. Because you might have the same field name in more than one of the tables, showing table names in the design grid helps to ensure that your query refers to the

field you intend.

In the Show row, you can use the check boxes to indicate the fields that will be included in the recordset. By default, Access includes all the fields you selected in the design grid. Sometimes you'll want to include a field in the query to allow you to select the records you want (such as the contracts for a certain date range), but you won't need that field in the recordset. You can add that field to the design grid so that you can define criteria, but you should deselect the Show check box beneath the field to exclude it from the recordset.

Finally, you can use the Criteria row and the rows labeled *Or* to enter the criteria you want to use as filters. Once you understand how a query is put together, you'll find it easy to specify exactly the fields and records that you want.

Specifying Fields

The first step in building a query is to select the fields you want in the recordset. You can select the fields in several ways. Using the keyboard, you can tab to a column in the design grid and press Alt-down arrow to open the list of available fields. (To move the cursor to the design grid, you can press the F6 key.) Use the up and down arrow keys to highlight the field you want, and then press Enter to select the field.

Another way to select a field is to drag it from one of the field lists in the upper part of the window to one of the columns in the design grid. In Figure 6-4, the Size field is being dragged to the design grid. When you drag a field, the mouse pointer turns into a small rectangle.

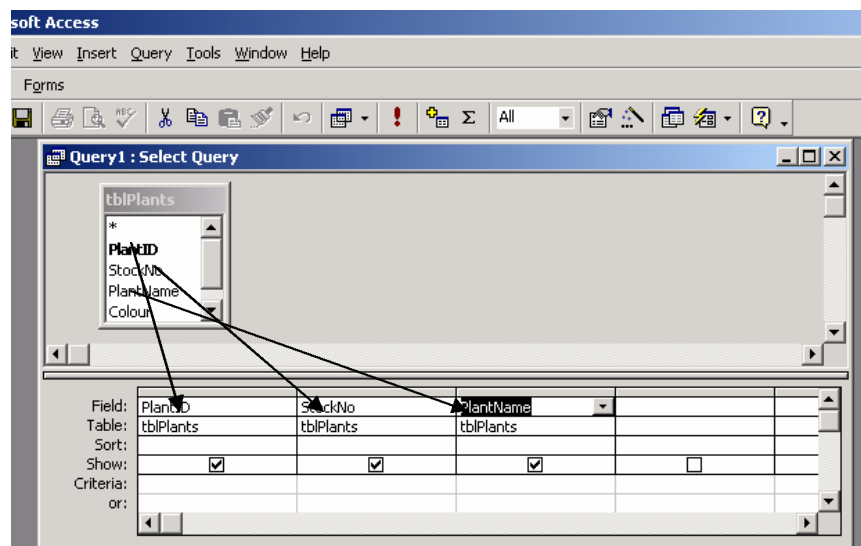


FIGURE 6-4. Dragging a field to a column in the design grid.

At the top of each field list in the upper part of the Query window (and also next to the first entry in the Field drop-down list in the design grid) is an asterisk (*) symbol. This symbol is shorthand for "all fields in the table or the query." When you want to include all the fields in a table

or a query, you don't have to define each one individually in the design grid (unless you also want to define some sorting or selection criteria for specific fields). You can simply add the asterisk to the design grid to include all the fields from a list. Note that you can add individual fields to the grid in addition to the asterisk in order to define criteria for those fields, but you should deselect the Show check box for the individual fields so that they don't appear twice in the recordset.

Tip: Another easy way to select all the fields in a table is to double-click on the title bar of the field list in the upper part of the Query window. This highlights all the fields. Then click in any of the highlighted fields and drag them to the field row in the design grid. While you're dragging, the mouse pointer changes to a multiple rectangle icon, indicating that you're dragging multiple fields. When you release the mouse button, you'll see that Access has copied all the fields to the design grid for you.

For this exercise, select PlantID, StockNo, and PlantName, from the tblPlants table in the Nursery database. If you switch the Query window to Datasheet view at this point, you'll see only the fields you selected from the records in the underlying table.

Setting Field Properties

In general, fields that are output by a query inherit the properties defined for the field in the table. You can define a different Description property (the information that is displayed on the status bar when you select that field in a Query window in Datasheet view), Format property (how the data is displayed), Decimal Places property (for numeric data), Input Mask property, and Caption property (the column heading). When you learn to define calculated fields later in this chapter, you'll see that it's a good idea to define the properties for these fields. If the field in the query is a foreign key linked to another table, Access propagates Lookup properties that you have defined in your table fields; however, you can use the properties in the Lookup tab on the query field properties to override them.

To set the properties of a field, click on any row of the field's column in the design grid, and then click the Properties button on the toolbar or choose Properties from the View menu to display the Field Properties window, shown in [Figure 6-5](#). Even though the fields in your query inherit their properties from the underlying table, you won't see those properties here. For example, the Format property for Size in tblPlants is Standard, and Decimal Places is 0, although neither value appears in the Field Properties window. Use the property settings in the Field Properties window to override any inherited properties and to customize how a field looks when viewed for this query. Try entering new property settings for the Size field, as shown in [Figure 6-5](#).

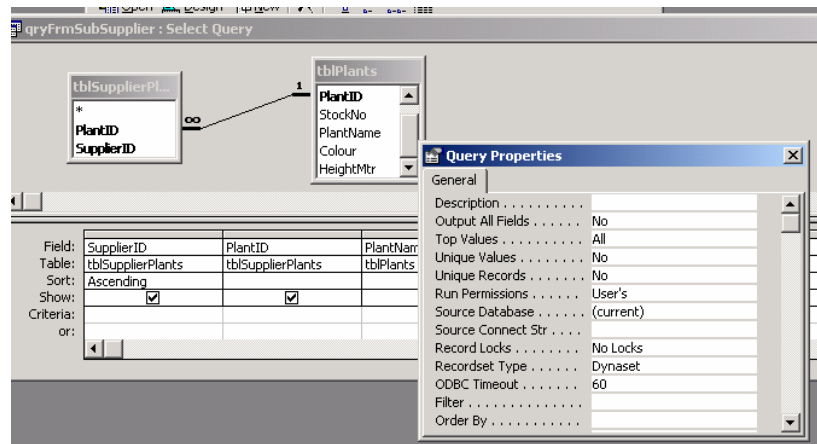


FIGURE 6-5. Setting properties for the Size field.

If you make these changes and switch to Datasheet view, you'll see a result similar to that shown in Figure 6-6. Notice that the Size column heading is now *SupplierID*, that one digit is displayed, and that the text on the status bar matches the new description, as shown in Figure 6-6.

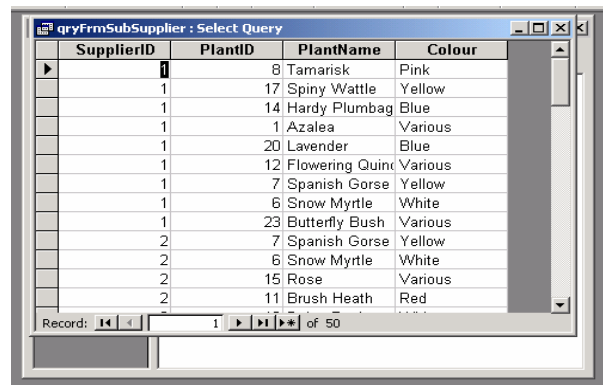


FIGURE 6-6. The Size field displayed with new property settings.

TIP: You'll notice that in Datasheet view, any new query you build using the `tblGroups` table will have no vertical gridlines and will use a serif font. These attributes are inherited from the table datasheet settings that you saw in the previous chapter. While you are in a query's Datasheet view, you can set the display format exactly as you would for a table in Datasheet view by using the commands on the Format menu. (See the previous chapter for details.) When you save the query, Access saves the custom query datasheet format settings.

Entering Selection Criteria

The next step is to further refine the values in the fields you want. The example shown in Figure 6-7 selects groups from the state of Washington.

Entering selection criteria in a query is similar to entering a validation rule for a field, which you learned about in last week. To look for a single value, as “Pink” simply type it in the Criteria row for the field colour you want to test. If the field

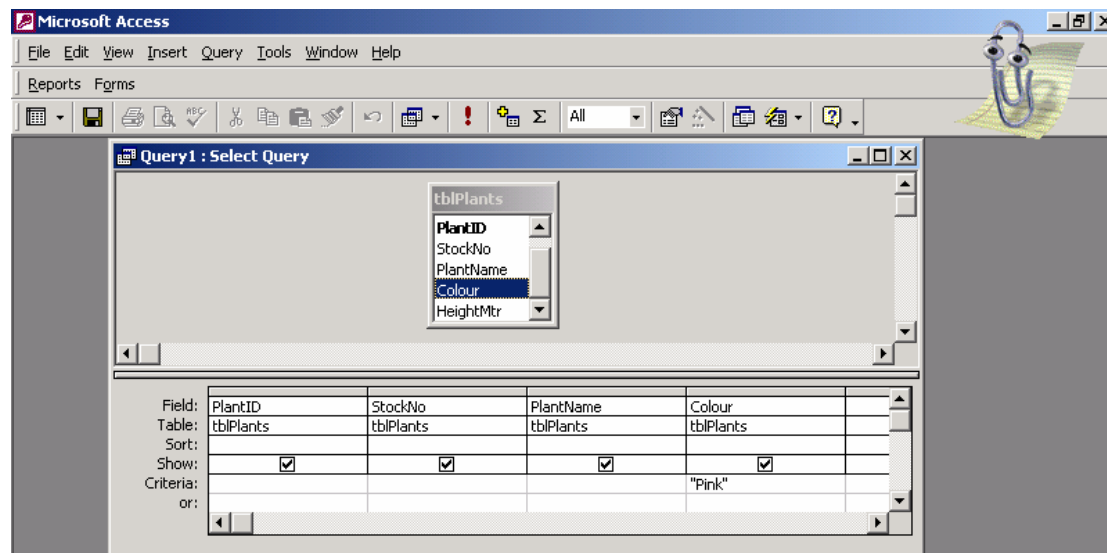


FIGURE 6-7.

A design grid that specifies "Pink" as a selection criterion.

you're testing is a text field and the value you're looking for has any blank spaces in it, you must enclose the value in quotation marks. Note that Access adds quotation marks for you around single text values. (In [Figure 6-7](#), *Pink* was typed, but the field shows "Pink" after Enter was pressed.)

If you want to test for any of several values, enter the values in the Criteria row, separated by the word *Or*. For example, specifying *Pink Or OR* searches for records for Yellow or Red. You can also test for any of several values by entering each value in a separate Criteria or Or row for the field you want to test. For example, you can enter *OR* in the Criteria row, *Pink* in the next row (the first Or row), and so on but you have to be careful if you're also specifying criteria in other fields, as explained below.

NOTE

It's a common mistake to get *Or* and *And* mixed up when typing a compound criteria for a single field. You may think to yourself, "I want all the entertainment groups in the state of Washington *and* California," and then type: *Pink And Yellow* in the Criteria row for the State field. When you do this, you're asking Access to find rows where (*Colour = "Pink"*) *And* (*Colour = "Yellow"*). Since a field in a record can't have more than one value at a time (it can't contain both the values "Pink" and "Yellow" in the same record), there won't be any records in the output. To look for all the rows for these two states, you need to ask Access to search for (*Colour = "Pink"*) *Or* (*Colour = "Yellow"*). In other words, type *Pink Or Yellow* in the Criteria row under the Colour field.

In the next section, you'll see that you can also include a comparison operator in the Criteria row so that, for example, you can look for values less than (<), greater than or equal to (>=), or not equal to (<>) the value that you specify.

AND & OR

When you enter criteria for several fields, all of the tests in a single Criteria row or Or row must be true for a record to be included in the recordset. That is, Access performs a logical AND operation between multiple criteria in the same row. So if you enter *Pink* in the Criteria row for Colour and <1.2 in the Criteria row for Height, the record must be for the Colour of pink *and* must have a height size of less than 1.2metre in order to be selected. If you enter *Pink Or OR* in the Criteria row for Colour and ≥ 1.2 *And* ≤ 1.5 in the Criteria row for Height, the record must be for the colour of pink or Yellow, *and* the group height must be between 1.2 and 1.5 meters inclusive.

Figure 6-8 shows the result of applying a logical AND operator between any two tests. As you can see, both tests must be true for the result of the AND to be true and for the record to be selected.

<u>AND</u>	<u>True</u>	<u>False</u>
<u>True</u>	True (Selected)	False (Rejected)
<u>False</u>	False (Rejected)	False (Rejected)

FIGURE 6-8.

The result of applying the logical AND operator between two tests.

When you specify multiple criteria for a field and separate the criteria by a logical OR operator, only one of the criteria must be true for the record to be selected. You can specify several OR criteria for a field, either by entering them all in a single Criteria row separated by the logical OR operator, as shown earlier, or by entering each subsequent criterion in a separate Or row. When you use multiple Or rows, if the criteria *in any one of the Or rows* is true, the record will be selected. Figure 6-9 shows the result of applying a logical OR operation between any two tests. As you can see, only one of the tests must be true for the result of the OR to be true and for the record to be selected.

<u>OR</u>	<u>True</u>	<u>False</u>
<u>True</u>	True (Selected)	True (Selected)
<u>False</u>	True (Selected)	False (Rejected)

FIGURE 6-9.

The result of applying the logical OR operator between two tests.

Let's look at a specific example. In Figure 6-10, you specify *WA* in the first Criteria row of the State field and <3 in that same Criteria row for the Size field. In the next row (the first Or row), you specify *OR* in the State field. When you run this query,

you get all the records for the state of Washington that also have a group size of less than 3. You also get any records for the state of Oregon regardless of the group size.

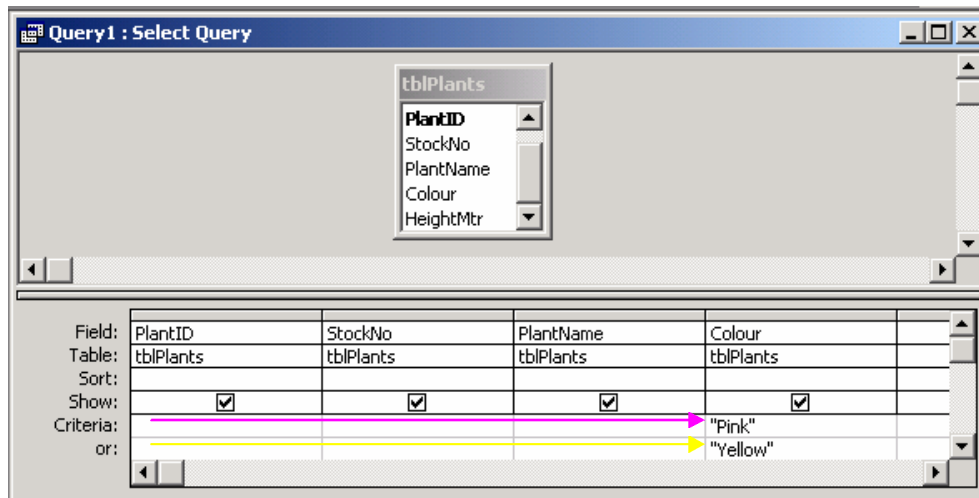


FIGURE 6-10.
A design grid that specifies multiple AND and OR selection criteria.

In [Figure 6-11](#), you can see the recordset (in Datasheet view) that results from running this query.

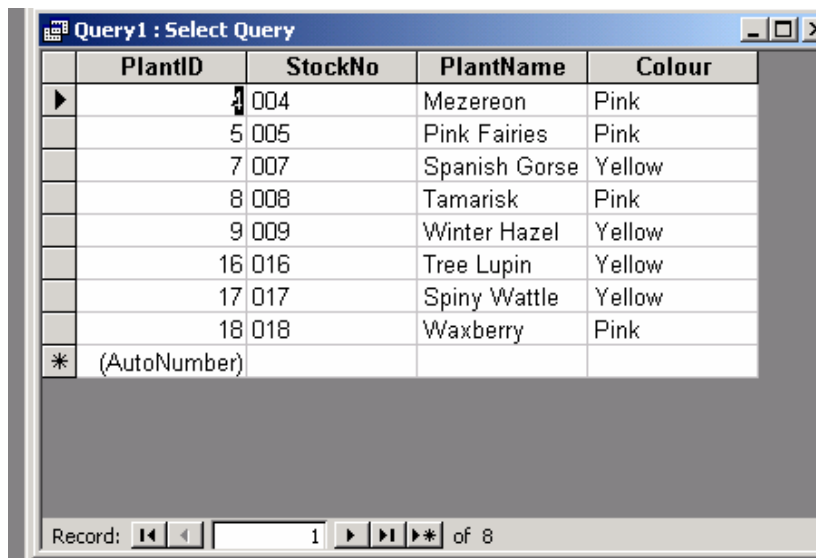


FIGURE 6-11.
The recordset of the query shown in [Figure 6-10](#).

If you also want to limit rows from groups in Oregon to those that have a size less than three, you must specify <3 again under Size in the first Or row that is, on the same row that filters for OR under State. Although this seems like extra work, this gives you complete flexibility to filter the data you want. You could, for example, include groups smaller than three members in Washington and groups smaller than five members in Oregon by placing a different criteria (<5) under Size on the row that filters for Oregon.

BETWEEN, IN, and LIKE

In addition to comparison operators, Access provides three special predicate clauses that are useful for specifying the data you want in the recordset:

BETWEEN	Useful for specifying a range of values. The clause <i>Between 10 And 20</i> is the same as specifying ≥ 10 And ≤ 20 .
IN	Useful for specifying a list of values, any one of which can match the field being searched. The clause <i>In ("WA", "CA", "ID")</i> is the same as <i>"WA" Or "CA" Or "ID"</i> .
LIKE	Useful for searching for patterns in text fields. You can include special characters and ranges of values in the Like comparison string to define the character pattern you want. Use a question mark (?) to indicate any single character in that position. Use an asterisk (*) to indicate zero or more characters in that position. The pound-sign character (#) specifies a single numeric digit in that position. Include a range in brackets ([]) to test for a particular range of characters in a position, and use an exclamation point (!) to indicate exceptions. The range [0-9] tests for numbers, [a-z] tests for letters, and [!0-9] tests for any characters except 0 through 9. For example, the clause <i>Like "?[a-k]d[0-9]*"</i> tests for any single character in the first position, any character from <i>a</i> through <i>k</i> in the second position, the letter <i>d</i> in the third position, any character from 0 through 9 in the fourth position, and any number of characters after that.

Suppose you want to find all entertainment groups in the city of Tacoma or in the town of Sumner that have between two and four members and whose name begins with the letter *B*. **Figure 6-12** shows how you would enter these criteria.

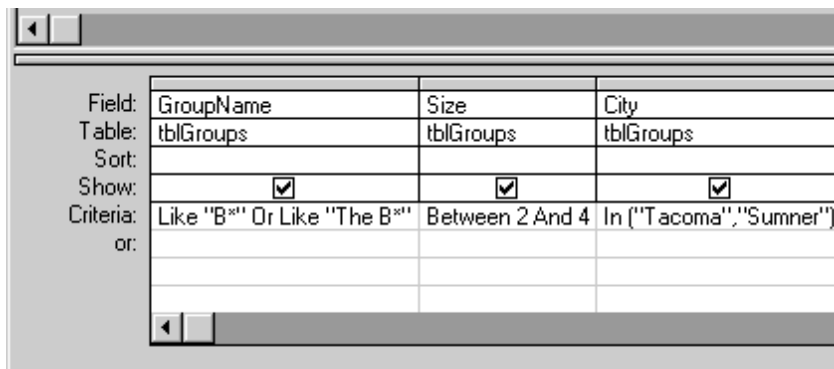


FIGURE 6-12.
A design grid that uses BETWEEN, IN, and LIKE.

Working with Dates and Times in Criteria

Access stores dates and times as double-precision floating-point numbers. The value to the left of the decimal point represents the day, and the fractional part of the number stores the time as a fraction of a day. Fortunately, you don't have to worry about converting internal numbers to specify a test for a particular date value because Access handles date and time entries in several formats.

You must always surround date and time values with pound signs (#) to tell Access that you're entering a date or a time. To test for a specific date, use the date notation that is most comfortable for you. For example, #April 15, 1997#, #4/15/97#, and #15-

Apr-1997# are all the same date if you chose English (United States) from the drop-down list on the Regional Settings tab of the Regional Settings Properties window. (You can display the Regional Settings Properties window by double-clicking on the Regional Settings icon in Control Panel.) Also, *#5:30 PM#* and *#17:30#* both specify 5:30 in the evening.

Access has several useful functions to assist you in testing date and time values. These are explained below with examples that use the *BeginningDate* field in the *tblContracts* table:

Day(date)	Returns a value from 1 through 31 for the day of the month. For example, if you want to select records with <i>BeginningDate</i> values after the 10th of any month, enter <i>Day([BeginningDate])</i> as a calculated field (see the next section), and enter <i>>10</i> as the criterion for that field.
Month(date)	Returns a value from 1 through 12 for the month of the year. For example, if you want to find all records that have a <i>BeginningDate</i> value of June, enter <i>Month([BeginningDate])</i> as a calculated field (see the next section), and enter <i>6</i> as the criterion for that field.
Year(date)	Returns a value from 100 through 9999 for the year. If you want to find a <i>BeginningDate</i> value in 1997, enter <i>Year([BeginningDate])</i> as a calculated field (see the next section), and enter <i>1997</i> as the criterion for that field.
Weekday(date)	As a default, returns a value from 1 (Sunday) through 7 (Saturday) for the day of the week. To find business day dates, enter <i>Weekday([BeginningDate])</i> as a calculated field (see the next section), and enter <i>Between 2 And 6</i> as the criterion for that field.
Hour(date)	Returns the hour (0 through 23). To find a scheduled start time before noon, enter <i>Hour([BeginningDate])</i> as a calculated field (see the next section), and enter <i><12</i> as the criterion for that field.
DatePart(interval, date)	Returns a portion of the date or time, depending on the interval code you supply. Useful interval codes are "q" for quarter of the year (1 through 4) and "ww" for week of the year (1 through 53). For example, to select dates in the second quarter, enter <i>DatePart("q",[BeginningDate])</i> as a calculated field (see the next section), and enter <i>2</i> as the criterion for that field.
Date	Returns the current system date. To select dates more than 30 days ago, enter <i><Date() - 30</i> as the criterion for that field.

Calculating Values

You can specify a calculation on any of the fields in your table and make that calculation a new field in the recordset. You can use any of the many built-in functions that Access provides. (See the examples above.) You can also create a field in a query by using arithmetic operators on fields in the underlying table to calculate a value. In a Supplier record, for example, you might have a *NumberStocks* field and a *PlantstPrice* (per each) field, but not the extended price (each price). You can include

that value in your recordset by typing the calculation in the field of an empty column in the design grid using the NumberOfPlants field, the multiplication operator (*), and the PlantsPrice field.

You can also create a new text (string) field by concatenating fields containing text, string constants, or numeric data. You create a string constant by enclosing the text in double or single quotation marks. Use the ampersand character (&) between fields or strings to indicate that you want to concatenate them. For example, you might want to create an output field that concatenates the LastName field, a comma, a blank space, and then the FirstName field.

The operators you can use in expressions include the following:

+	Adds two numeric expressions.
-	Subtracts two numeric expressions.
*	Multiplies two numeric expressions.
/	Divides the first numeric expression by the second numeric expression.
\	Rounds both numeric expressions to integers and divides the first integer by the second integer. The result is rounded to an integer.
^	Raises the first numeric expression to the power indicated by the second numeric expression.
MOD	Rounds both numeric expressions to integers, divides the first integer by the second integer, and returns the remainder.
&	Creates an extended text string by concatenating the first text string to the second text string. If either expression is a number, Access converts it to a text string before concatenating the expressions.

Try creating a query on the tblSuppliers table in the Nusery database that shows a field containing the Suppliername, followed by a single field containing the Contact person, a comma and a blank space, Phone number, Your expression should look like this:

[Contact] & ", " & [Phone]

you can join more by follow this join rules expression.

The Query window in Design view for this example is shown in [Figure 6-14](#). Notice that I clicked in the Field row of the column I wanted and then pressed Shift-F2 to open the Zoom window, where it is easier to enter the expression.

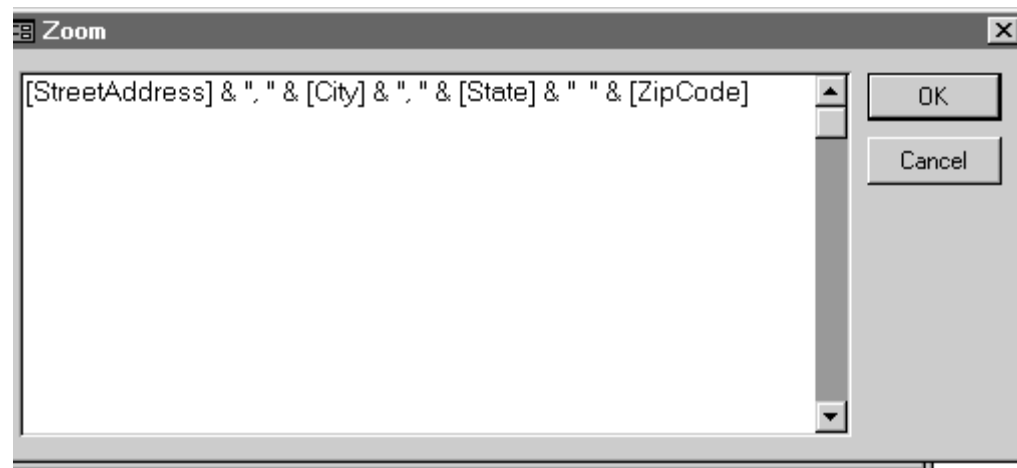


FIGURE 6-14.
Editing an expression in the Zoom window.

Try typing within the Expr1 field in Datasheet view. Because this display is a result of a calculation (concatenation of strings), Access won't let you update the data in this column.

Designing a Relational Database

Regardless of which data store you choose, designing the database structure is likely to be the most challenging part of building the relational-database solution. In order to understand how the tables in the database should be structured and how they should relate to one another, you must understand the data. Although it is not difficult to modify the data model while you are developing the solution, it is much more challenging once your customers are using the solution. Therefore, it is important to put as much effort as necessary into the process of designing the data model before you begin writing code.

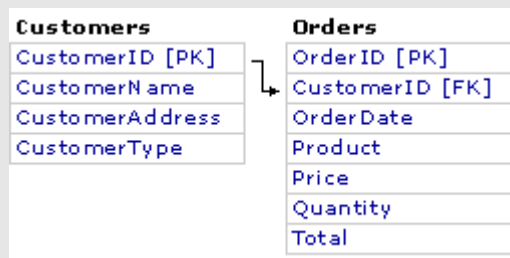
One way to start is to think of all the questions this database must answer. The answers become the columns (or fields) in your tables. How many corporate customers do we have by country? This could require the following columns: customer, country.

In addition to determining the columns in your tables, you must also set up the relationships between tables. Sometimes you may have what appears to be a straightforward one-to-many relationship that turns out to be much more complex.

The following example illustrates some of the issues you must consider when designing a relational database.

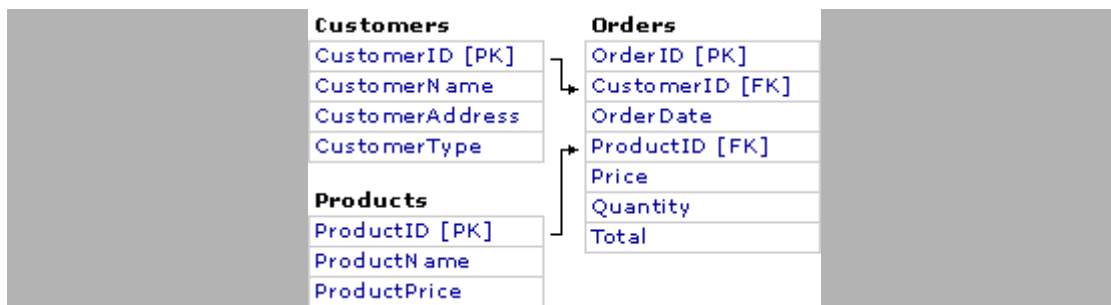
Note PK = Primary Key, FK = Foreign Key

Relational Database Design Example: Phase I



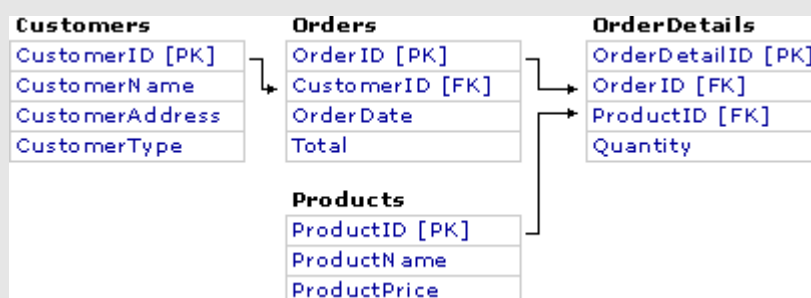
The simple one-to-many design with a Customers table with a single relationship to the Orders table has several limitations. For example, all the product information would have to be re-entered on each order. That would slow down the order entry process and could cause problems as products are added or changed.

Relational Database Design Example: Phase II



Adding a Products table to relate product information to the Order table is better, but there is still a major limitation. You can only place one order per order number. Therefore, another table is required to handle the order details.

Relational Database Design Example: Phase III



Adding an OrderDetails table and relating the Products table to this new table handles the relationship better. You can maintain and update the products separately, and each order can have one or many order details. Notice that total is still stored in the Orders table. This column could be updated using code, once the order total was calculated for good record keeping.

The design of your database itself enforces certain rules on the way users can enter data. For example, a user cannot violate a table's primary key by adding a duplicate record. In addition, you can establish custom validation rules using triggers that prevent users from entering invalid data.

Note Currently, Access Workflow Designer does not support multiple triggers within one table.

The Access Workflow Designer automatically tracks foreign-key constraints when you add a main table to the table hierarchy and creates a tree of detail and lookup tables. If you have other related tables for which there are no foreign-key constraints, you can add manually those as lookup tables using the Access Workflow Designer. For information about how Access Workflow Designer manages tables, see [Setting up a Table Hierarchy](#).

Retrieving, Analyzing, and Presenting Data

Once you have designed the data-storage and data-entry components of your solution, you should begin thinking about how to present and summarize the data in a format that makes sense to users. Although generally not as difficult as database design, determining what data users want to see and building reports to display the data in a usable format can be a challenging task. Preplanning your reports can also lead you to rethink or enhance your database design. Sometimes, the lack of a certain column required for grouping or sorting does not become obvious until the reports are being designed.

Here are some questions to ask yourself as you design the reporting component of a solution:

- Must the report be dynamically linked to the data source, or can it be a static report? If the report must display the most current data, it should be dynamically linked to the data source. On the other hand, if the data is not updated frequently or if the report must be re-created regularly because the structure of the underlying data source changes, you can create a static report.
- Must users interact with the data in the report, or can the report be read-only? If users must perform calculations on the data or manipulate the data to display it in novel ways, you may want to create the report in Microsoft Excel or use the Microsoft Office Web Components to create it in a Web page.
- Must users be able to view the report from a Web page or from within one of the Office applications?

An Access project can store forms, reports, data access pages, macros, and Microsoft Visual Basic® for Applications modules locally in your client solution file and use the OLE DB connection to display and work with the tables, views, relationships, and stored procedures that are stored on SQL Server. You create the forms, reports, macros, and Visual Basic for Applications modules in an Access project by using most of the same tools and wizards you use to create these objects in Access databases. This makes it possible for you to develop quickly client/server solutions that work directly against a SQL Server.

Note Although an Access project file uses an OLE DB connection to connect to a database, it can only use the Microsoft OLE DB Provider for SQL Server and can only connect to SQL Server 6.5 (with Service Pack 5) and SQL Server 7.0 databases. This is because the database creation and design tools in Access 2000 can only support SQL Server 6.5 or SQL Server 7.0 databases.

TIP: Access also makes it possible for you to create new SQL Server databases and provides a variety of visual tools to create and modify the design of tables, views, stored procedures, triggers, and database diagrams on your database server. The tables, views, and stored procedures you create, as well as SQL SELECT statements, are all valid data sources for Access forms, reports, and data access pages.

In addition to providing you with the ability to create and design client/server solutions from scratch, Access 2000 also includes the Upsizing Wizard, which makes it possible for you to convert an existing Access database to a client/server solution by creating a new SQL Server database linked to an Access client application.

With the addition of Access Workflow Designer, you can add workflow, offline replication, and security, as well as the ability to create a template of your database solution.

Advanced Query Design—And SQL select commands language

Underlying every query in Microsoft Access is the SQL database command language. Although you can design most queries using the simple Access design grid, Access stores every query you design as an SQL command. For advanced types of queries that use the results of a second query as a comparison condition, you need to know SQL in order to define the second query (called a *subquery*). Also, you cannot use the design grid to construct all the types of queries available in the product; you must use SQL for some of them.

In few weeks you have learned make-table, update, append, and delete queries in NURSERY project.

In this lesson explores and explains the various syntax elements in the Access variant of SQL, which you can use to build select, total, **crosstab**, **make-table**, **update**, **append**, and **delete queries**. You'll find brief examples that explain how to use each element. At the end of this chapter are several examples of complex queries that are implemented in the Entertainment Schedule database. This sample database also uses some of the more complex features of SQL in Access to accomplish tasks described in Part 6 of this book. As you become more familiar with SQL, you can learn a lot about the language and how it's implemented in Access by using the design grid to design a query and then switching to SQL view to see how Access translates the query into SQL statements.

A Brief History of SQL

In the early 1970s, IBM created a language called *Structured English Query Language* (SEQUEL) for a research project on relational database management systems. This language evolved into SEQUEL/2 and finally into *Structured Query Language* (SQL). Other companies became interested in the concept of relational databases and the emerging SQL interface. Relational Software, Inc., (now the Oracle Corporation) created a relational database product called Oracle in 1979. IBM released its first relational database product, called SQL Data System (SQL/DS), in 1981.

In 1982, the *American National Standards Institute* (ANSI), realizing the potential significance of the relational model, began work on a *Relational Database Language* (RDL) standard. By 1984, acceptance in the marketplace of such products as Oracle, SQL/DS, and IBM's DB2 caused the ANSI committee to focus on SQL as the basis for the new RDL standard. The first version of this standard, SQL-86, was adopted by both ANSI and the International Standards Organization (ISO) in October 1986. An update to SQL-86 covering integrity enhancements was adopted in 1989. The current standard, often referred to as "SQL2" or "SQL-92," reflects extensive work by the international standards bodies to both enhance the language and correct many missing, confusing, or ambiguous features in the original 1986 standard.

The standard as it currently exists is both in its core a common subset of the major implementations and in its entirety a superset of almost all implementations. That is, the core of the standard contains features found in virtually every commercial implementation of the language, yet the entire standard includes enhanced features that many vendors have yet to implement.

As mentioned in the previous chapter, the SQL Access Group consortium of database vendors has published what could be regarded as the "commercial standard" for SQL - a variant of the

language that can be "spoken" by (or mapped to) every major relational database product. An extended version of this Common Language Interface (CLI) is part of the draft SQL3 standard under consideration in 1996. Microsoft has implemented support for the CLI in the Open Database Connectivity (ODBC) application programming interface (API) for Windows. This allows vendors of applications for Microsoft Windows to connect to each other using the SQL Access Group standard. Microsoft Access connects to many databases via the ODBC standard and also "speaks" a major subset of the SQL Access Group's standard SQL.

SQL Syntax Conventions

The following table lists the SQL syntax conventions you'll encounter in this lesson

SQL Convention	Meaning
UPPERCASE	Uppercase letters indicate keywords and reserved words that you must enter exactly as shown. Note that Microsoft Access understands keywords entered in either uppercase or lowercase.
<i>Italic</i>	Italicised words represent variables that you supply.
Angle brackets <>	Angle brackets enclose syntactic elements that you must supply. The words inside the angle brackets describe the element but do not show the actual syntax of the element.
Brackets []	Brackets enclose optional items. If more than one item is listed, the items are separated by a pipe () character. Choose one or none of the elements. Do not enter the brackets or the pipe. Note that Access in many cases requires you to enclose names in brackets. In this chapter, when brackets are required as part of the syntax of variables that you must supply, the brackets are italicized, as in <i>[MyTable].[MyField]</i> .
Braces { }	Braces enclose one or more options. If more than one option is listed, the options are separated by a pipe () character. Choose one option from the list. Do not enter the braces or the pipe.
Ellipses ...	Ellipses indicate that you can repeat an item one or more times. When a comma is shown with an ellipsis, enter the comma between items.

You must enter all other characters, such as parentheses and colons, exactly as they appear in the syntax line.

SQL SELECT Syntax in Microsoft Access

The SELECT statement forms the core of the SQL database language. You use the SELECT statement to select or retrieve rows and columns from database tables. The SELECT statement syntax contains five major clauses, generally constructed as follows:

```
SELECT <field list>
FROM <table list>
[WHERE <row selection specification>]
```


**[GROUP BY <grouping specification>]
[HAVING <group selection specification>]
[ORDER BY <sorting specification>];**

Microsoft Access implements four significant extensions to the language: TRANSFORM, to allow you to build crosstab queries; IN, to allow you to specify a remote database connection or to specify column names in a crosstab query; DISTINCTROW, to limit the rows returned from the <table list> to rows that have different primary key values in the tables that supply columns in the <field list>; and WITH OWNERACCESS OPTION, to let you design queries that can be run by users who are authorized to use the query but who have no access rights to the tables referenced in the query.

The following sections in the lesson are a reference guide to the Access implementation of SQL for select, total, and crosstab queries. The language elements are presented in alphabetic order.

Notes:

When you build a query in SQL view, you can insert a carriage return between elements to improve readability. In fact, Access inserts carriage returns between major clauses when you save and close your query. The only time you might not include carriage returns in an SQL statement is when you're defining an SQL statement for a string literal in Visual Basic for Applications (VBA). VBA requires that a literal be defined as a single line in a procedure. (A literal is a value that is expressed as itself rather than as a variable's value or as the result of an expression.) In Access for Windows and Access 2K pro you can create a long literal within VBA by using string concatenation and line continuation characters to make a single SQL statement more readable in code.

Column-name

Specifies the name of a column in an expression.

Syntax:

[[/]{table-name|select-query-name|correlation-name}[/]].[field-name]

Notes:

You must supply a qualifier to the field name only if the name is ambiguous within the context of the query or subquery (for example, if the same field name appears in more than one table or query listed in the FROM clause).

The *table-name*, *select-query-name*, or *correlation-name* that qualifies the field name must appear in the FROM clause of the query or subquery. If a table or query has a correlation-name, you must use the alias name, not the actual name of the table or query.

You must supply the enclosing brackets only if the name contains an embedded blank. Embedded blanks and enclosing brackets are not supported in the ANSI standard.

Expression

Specifies a value in a predicate or in the select list of a SELECT statement or subquery.

Syntax:

[+|-] {*function* | [(*expression*)] | *literal* | *column-name*}

[{+|-|*|/|\|^|MOD|&} {*function* | [(*expression*)] | *literal* | *column-name*}]...

Notes:

Function—You can specify one of the SQL total functions: AVG, COUNT, MAX, MIN, STDEV, STDEVP, SUM, VAR, or VARP; however, you cannot use an SQL total function more than once in an expression. You can also use any of the functions built into Access or any function you define using VBA.

[(*expression*)]—You can construct an expression from multiple expressions separated by operators. Use parentheses around expressions to clarify the evaluation order. (See the examples later in this section.)

literal—You can specify a numeric or an alphanumeric constant. You must enclose an alphanumeric constant in single or double quotation marks. To include an apostrophe in an alphanumeric constant, enter the apostrophe character twice in the literal string or enclose the literal string in double quotation marks. If the expression is numeric, you must use a numeric constant. Enclose a date/time literal within pound (#) signs. A date/time literal must follow the ANSI standard mm/dd/yy (U.S.) format. (Note: Use the format specified in your Regional settings in the Windows Control Panel when entering date/time literals on the query design grid.)

column-name—You can specify the name of a column in a table or a query. You can use a column name only from a table or query that you've specified in the FROM clause of the statement. If the expression is arithmetic, you must use a column that contains numeric data. If the same column-name appears in more than one of the tables or queries included in the query, you must fully qualify the name with the query name, table name, or correlation name, as in [TableA].[Column1]. Although in ANSI SQL you can reference an *output-column-name* anywhere within an expression, Microsoft Access supports this only within the <field-list> of a SELECT statement. Access does not support references to named expression columns in GROUP BY, HAVING, ORDER BY, or WHERE clauses. You must repeat the expression rather than use the column-name. See "SELECT Statement" later in this chapter for details about output-column-name.

+ | - | * | / | \ | ^ | MOD—You can combine multiple numeric expressions with arithmetic operators that specify a calculation. If you use arithmetic operators, all expressions within an expression must evaluate as numeric data types.

&—You can concatenate alphanumeric expressions by using the special & operator.

Examples:

To specify the average of a column named COST, enter the following:

AVG(COST)

To specify one-half the value of a column named PRICE, enter the following:

(PRICE * .5)

To specify a literal for 3:00 P.M. on March 1, 1996, enter the following:

#3/1/96 3:00PM#

To specify a character string that contains the name *Acme Mail Order Company*, enter the following:

"Acme Mail Order Company"

To specify a character string that contains a possessive noun (requiring an embedded apostrophe), enter the following:

"Andy's Hardware Store"

or

`Andy"s Hardware Store'

To specify a character string that is the concatenation of fields from a table named Customer List containing a person's first and last name with an intervening blank, enter the following:

[Customer List].[First Name] & " " &

[Customer List].[Last Name]

FROM Clause

Specifies the tables or queries that provide the source data for your query.

Syntax:

```
FROM {table-name [[AS] correlation-name] |  
      select-query-name [[AS] correlation-name]} |  
      <joined table>, . . .  
      [IN <source specification>]
```

where <*joined table*> is

```
{(table-name [[AS] correlation-name] |  
  select-query-name [[AS] correlation-name] |  
  <joined table>}
```

```
{INNER | LEFT | RIGHT} JOIN  
  {table-name [[AS] correlation-name] |  
  select-query-name [[AS] correlation-name] |  
  <joined table>}
```

ON <join-specification>

Notes:

You can supply a correlation name for each table name or query name. You can use this correlation name as an alias for the full table name when qualifying column names in the <field-list>, in the <join-specification>, or in the WHERE clause and subclauses. If you're joining a table or a query to itself, you must use correlation names to clarify which copy of the table or query you're referring to in the select list, join criteria, or selection criteria. If a table name or a query name is also an SQL reserved word (for example, "Order"), you must enclose the name in brackets.

If you include multiple tables in the FROM clause with no JOIN specification but do include a predicate that matches fields from the multiple tables in the WHERE clause, Access in most cases optimizes how it solves the query by treating the query as a JOIN.

Example

```
SELECT *  
  FROM TableA, TableB  
  WHERE TableA.ID = TableB.ID
```

is treated by Access as if you had specified

```
SELECT *  
  FROM TableA  
  INNER JOIN TableB  
  ON TableA.ID = TableB.ID
```

You cannot update fields in a table by using a recordset opened on the query, the query datasheet, or a form bound to a multiple table query where the join is expressed using a table-list and a WHERE clause. In many cases you can update the fields in the underlying tables when you use the JOIN syntax.

When you list more than one table or query without join criteria, the source is the *Cartesian product* of all the tables. For example, *FROM TableA, TableB* instructs Access to search all the rows of TableA matched with all the rows of TableB. Unless you specify other restricting criteria, the number of logical rows that Access processes could equal the number of rows in TableA *times* the number of rows in TableB. Access then returns the rows in which the selection criteria specified in the WHERE and HAVING clauses evaluate to True.

Example:

To select information about customers and their purchases of more than \$100, enter the following (*qxmplCustomerOrder>100*):

```
SELECT tblCustomers.CustomerID, tblCustomers.FirstName,
tblCustomers.LastName, tblOrders.OrderDate,
tblOrderDetails.ISBNNumber, tblBooks.Title,
tblOrderDetails.Quantity,
tblOrderDetails.Discount,
tblBooks.SuggPrice,
CCur(CLng((tblOrderDetails.Quantity*
tblBooks.SuggPrice) *
(1-tblOrderDetails.Discount)*100)/100)
AS ExtPrice
FROM (tblCustomers
INNER JOIN tblOrders
ON tblCustomers.CustomerID =
tblOrders.CustomerID)
INNER JOIN (tblBooks
INNER JOIN tblOrderDetails
ON tblBooks.ISBNNumber =
tblOrderDetails.ISBNNumber)
ON tblOrders.OrderID = tblOrderDetails.OrderID
WHERE (((CCur(CLng((tblOrderDetails.Quantity*
tblBooks.SuggPrice) * (1-tblOrder
Details.Discount)*100)/100))>100));
```

Notes

In the [ANSI](#) SQL standard, you can also write the WHERE clause as WHERE ExtPrice > 100;. Microsoft Access does not support this syntax.

GROUP BY Clause

In a SELECT statement, specifies the columns used to form groups from the rows selected. Each group contains identical values in the specified column(s). In Access, you use the GROUP BY clause to define a total query. You must also include a GROUP BY clause in a crosstab query. (*See the TRANSFORM statement for details.*)

Syntax:

GROUP BY *column-name*,

Notes:

A column name in the GROUP BY clause can refer to any column from any table in the FROM clause, even if the column is not named in the select list. If the GROUP BY clause is preceded by a WHERE clause, Access creates the groups from the rows selected after it applies the WHERE clause. When you include a GROUP BY clause in a SELECT statement, the select list must be made up of either SQL total functions or column names specified in the GROUP BY clause.

Examples:

To find the largest order from any customer within each zip code, create the following two queries. (We'll use the first of these two queries as examples repeatedly throughout this chapter.)

Total functions AVG, COUNT, MAX, MIN, STDEV, STDEVP, SUM, VAR, VARP; HAVING Clause; Search-Condition; SELECT Statement; and WHERE Clause.

%%%%%%%%%stopped here 11/8/96

A query to calculate the total for each order (*qxmplOrderTotals*):

```
SELECT tblOrders.OrderID, tblOrders.OrderDate,
       tblCustomers.CustomerID,
       tblCustomers.City,
       tblCustomers.StateOrProvince,
       tblCustomers.PostalCode,
       Sum(([Quantity]*[SuggPrice])*
          (1-[Discount])) AS OrderTot
FROM (tblCustomers
      INNER JOIN tblOrders
      ON tblCustomers.CustomerID =
         tblOrders.CustomerID)
     INNER JOIN (tblBooks
                INNER JOIN tblOrderDetails
                ON tblBooks.ISBNNumber =
                   tblOrderDetails.ISBNNumber)
     ON tblOrders.OrderID =
        tblOrderDetails.OrderID
GROUP BY tblOrders.OrderID, tblOrders.OrderDate,
         tblCustomers.CustomerID,
         tblCustomers.City,
         tblCustomers.StateOrProvince,
         tblCustomers.PostalCode;
```

Build a query on the first query to find the largest order (*qxmplLargestOrderByZip*)

```
SELECT qxmplOrderTotals.PostalCode,
       Max(qxmplOrderTotals.OrderTot) AS
MaxOfOrderTot
FROM qxmplOrderTotals
GROUP BY qxmplOrderTotals.PostalCode;
```

To find the average and maximum prices for items in the book catalog by category name, enter the following (*qxmplCategoryAvgMaxPrice*):

```
SELECT tblCategories.Category,
       Avg(tblBooks.SuggPrice) AS AvgOfSuggPrice,
       Max(tblBooks.SuggPrice) AS MaxOfSuggPrice
FROM tblBooks
     INNER JOIN (tblCategories
                INNER JOIN tblBookCategories
                ON tblCategories.CategoryID =
                   tblBookCategories.CategoryID)
     ON tblBooks.ISBNNumber =
        tblBookCategories.ISBNNumber
GROUP BY tblCategories.Category;
```

HAVING Clause

Specifies groups of rows that appear in the logical table (an Access recordset) defined by a SELECT statement. The search condition applies to columns specified in a GROUP BY clause, to columns created by total functions, or to expressions containing total functions. If a group doesn't pass the search condition, it is not included in the logical table.

Syntax:

HAVING search-condition

Notes:

If you do not include a GROUP BY clause, the select list must be formed by using one or more of the SQL total functions.

The difference between the HAVING clause and the WHERE clause is that the WHERE search condition applies to single rows before they are grouped, while the HAVING search condition applies to groups of rows.

If you include a GROUP BY clause preceding the HAVING clause, the search condition applies to each of the groups formed by equal values in the specified columns. If you do not include a GROUP BY clause, the search condition applies to the entire logical table defined by the SELECT statement.

Examples:

To find the cities in which customers' average purchase is less than the average of purchases by all customers, create the following two queries.

A query to calculate the total for each order (*qxmplOrderTotals*):

```
SELECT tblOrders.OrderID, tblOrders.OrderDate,
tblCustomers.CustomerID,
tblCustomers.City,
tblCustomers.StateOrProvince,
tblCustomers.PostalCode,
Sum(([Quantity]*[SuggPrice])*
(1-[Discount])) AS OrderTot
```

Total functionsAVG, COUNT, MAX, MIN, STDEV, STDEVP, SUM, VAR, VARP;
GROUP BY Clause; Search-Condition; SELECT Statement; and WHERE Clause.

```
FROM (tblCustomers
INNER JOIN tblOrders
ON tblCustomers.CustomerID =
tblOrders.CustomerID)
INNER JOIN (tblBooks
INNER JOIN tblOrderDetails
ON tblBooks.ISBNNumber =
tblOrderDetails.ISBNNumber)
ON tblOrders.OrderID =
```

```
tblOrderDetails.OrderID
GROUP BY tblOrders.OrderID,
tblOrders.OrderDate,
tblCustomers.CustomerID,
tblCustomers.City,
tblCustomers.StateOrProvince,
tblCustomers.PostalCode;
```

Build a query on the first query to find the largest order (*qxmplUnderAvgCities*):

```
SELECT qxmplOrderTotals.City,
Avg(qxmplOrderTotals.OrderTot) AS
AvgOfOrderTot
FROM qxmplOrderTotals
GROUP BY qxmplOrderTotals.City
HAVING (((Avg(qxmplOrderTotals.OrderTot))
<(SELECT Avg([OrderTot])
FROM qxmplOrderTotals)));
```

To find the average and maximum order amounts for customers in the state of Washington for every month in which the maximum order amount is less than \$1,900, create the following two queries.

A query to calculate the total for each order (*qxmplOrderTotals*):

```
SELECT tblOrders.OrderID, tblOrders.OrderDate,
tblCustomers.CustomerID,
tblCustomers.City,
tblCustomers.StateOrProvince,
tblCustomers.PostalCode,
Sum(([Quantity]*[SuggPrice])*
(1-[Discount])) AS OrderTot
FROM (tblCustomers
INNER JOIN tblOrders
ON tblCustomers.CustomerID =
tblOrders.CustomerID)
INNER JOIN (tblBooks
INNER JOIN tblOrderDetails
ON tblBooks.ISBNNumber =
tblOrderDetails.ISBNNumber)
ON tblOrders.OrderID =
tblOrderDetails.OrderID
GROUP BY tblOrders.OrderID, tblOrders.OrderDate,
tblCustomers.CustomerID,
tblCustomers.City,
tblCustomers.StateOrProvince,
tblCustomers.PostalCode;
```

Build a query on the first query to find the largest order (*qxmplWAOrdersMax<1900*):

```
SELECT Month([OrderDate]) AS Month,
Avg(qxmplOrderTotals.OrderTot) AS
AvgOfOrderTot,
Max(qxmplOrderTotals.OrderTot) AS
MaxOfOrderTot
FROM qxmplOrderTotals
WHERE (((qxmplOrderTotals.StateOrProvince)="WA"))
GROUP BY Month([OrderDate])
HAVING (((Max(qxmplOrderTotals.OrderTot))<1900));
```

IN Clause

Specifies the source for the tables in a query. The source can be another Access database; a dBASE, Microsoft FoxPro, or Paradox file; or any database for which you have an ODBC driver. This is an Access extension to standard SQL.

Syntax:

```
IN <"source database name"
"> <[source connect string]
]>
```

Enter "*source database name*" and [*source connect string*]. (Be sure to include the quotation marks and the brackets.) If your database source is Access, enter only "*source database name*". Enter these parameters according to the type of database to which you are connecting, as follows:

Database Name	Source Database Name	Source Connect String
Access	"drive:\path\filename"	(none)
dBASE III	"drive:\path"	[dBASE III;]
dBASE IV	"drive:\path"	[dBASE IV;]
dBASE 5	"drive:\path"	[dBASE 5.0;]
Paradox 3.x	"drive:\path"	[Paradox 3.x;]
Paradox 4.x	"drive:\path"	[Paradox 4.x;]
Paradox 5.x	"drive:\path"	[Paradox 5.x;]
FoxPro 2.0	"drive:\path"	[FoxPro 2.0;]
FoxPro 2.5	"drive:\path"	[FoxPro 2.5;]
FoxPro 2.6	"drive:\path"	[Fox Pro 2.6;]
ODBC	(none)	[ODBC; DATABASE= <i>defaultdatabase</i> ; UID= <i>user</i> ; PWD= <i>password</i> ; DSN= <i>datasourcename</i>]

Notes:

The IN clause applies to all tables referenced in the FROM clause and any subqueries in your query. You can refer to only one external database within a query. If you need to refer to more than one external file or database, attach those files as tables in Access and use the logical attached table names instead.

For ODBC, if you omit the DSN= or DATABASE= parameters, Access prompts you with a dialog box showing available data sources so that you can select the one you want. If you omit the UID= or PWD= parameters and the server requires a UserID and password, Access prompts you with a login dialog box for each table accessed.

For dBASE, Paradox, and FoxPro databases, you can provide an empty string ("") for *source database name* and provide the path or

dictionary filename using the DATABASE= parameter in *source connect string* instead.

Examples:

To retrieve the Company Name field in the Northwind Traders sample database without having to attach the Customers table, enter the following:

```
SELECT Customers.[CompanyName]
FROM Customers
IN "C:\Program Files\Microsoft Office\Office\
SAMPLES\NORTHWIND.MDB";
```

To retrieve data from the CUST and ORDERS sample files distributed with dBASE IV, enter the following:

```
SELECT CUST.CUST_ID, CUST.CUSTOMER,
ORDERS.DATE_TRANS, ORDERS.PART_ID,
ORDERS.PART_QTY
FROM CUST
INNER JOIN ORDERS
ON CUST.CUST_ID = ORDERS.CUST_ID
IN "" [dBASE IV;DATABASE=C:\DBASE\SAMPLES;];
```

JOIN Operation

Much of the power of SQL derives from its ability to combine (join) information from several tables or queries and present the result as a single logical recordset. In many cases, Access lets you update the recordset of a joined query as if it were a single base table.

Use a JOIN operation in a FROM clause to specify how you want to link two tables to form a logical recordset from which you can select the information you need. You can ask Access to join only matching rows in both tables (called an *inner join*) or to return all rows from one of the two tables even when a matching row does not exist in the second table (called an *outer join*). You can nest multiple join operations to join, for example, a third table with the result of joining two other tables.

Syntax:

```
{table-name [[AS] correlation-name
]|
select-query-name [[AS] correlation-name
]|
<joined table>}
```

FROM Clause, HAVING Clause, Predicate: Comparison, Search-Condition, SELECT Statement, and WHERE Clause.

```
{INNER | LEFT | RIGHT} JOIN
{table-name [[AS] correlation-name
]|
select-query-name [[AS]
```

```

correlation-name] |
  <joined table>
ON <join-specification>

```

where *<joined table>* is the result of another join operation, and where *<join-specification>* is a search condition made up of predicates that compare fields in the first table, query, or joined table with fields in the second table, query, or joined table.

Notes:

You can supply a correlation name for each table or query name. You can use this correlation name as an alias for the full table name when qualifying column names in the select list or in the WHERE clause and subclauses. If you're joining a table or a query to itself, you must use correlation names to clarify which copy of the table or the query you're referring to in the select list, join criteria, or selection criteria. If a table name or query name is also an SQL reserved word (for example, "Order"), you must enclose the name in brackets.

Use INNER JOIN to return all the rows that match the join specification in both tables. Use LEFT JOIN to return all the rows from the first logical table (where *logical table* is any table, query, or joined table expression) joined on the join specification with any matching rows from the second logical table. When no row matches in the second logical table, Access returns Null values for the columns from that table. Conversely, RIGHT JOIN returns all the rows from the second logical table joined with any matching rows from the first logical table.

When you use only *equals* comparison predicates in the join specification, the result is called an *equi-join*. Access can graphically display equi-joins in the design grid but cannot display nonequi-joins. If you want to define a join on a nonequals comparison (<, >, <>, <=, or >=), you must define the query using the SQL view. When you join a table to itself using an equals comparison predicate, the result is called a *self-join*.

Examples:

To select information about books and their authors, sorted by book identifier (ISBN), enter the following (*qxmplBooksAndAuthors*):

```

SELECT tblBooks.*,
  [tblAuthors].[LastName] & ", " &
  [tblAuthors].[FirstName] &
  IIf(IsNull([tblAuthors].[MiddleInit]),Null," "
  & [tblAuthors].[MiddleInit] & ".")
  AS AuthorName,
tblAuthors.EmailAddress, tblAuthors.Photograph,
tblAuthors.Bio
FROM tblBooks
  INNER JOIN (tblAuthors
  INNER JOIN tblBookAuthor
  ON tblAuthors.AuthorID =
  tblBookAuthor.AuthorID)
  ON tblBooks.ISBNNumber =
  tblBookAuthor.ISBNNumber;

```

To find out which customers did not send in order number 25, enter the following (*qxmplNonEquiJoin*):

```
SELECT tblCustomers.CustomerID, tblCustomers.LastName,  
tblCustomers.FirstName, tblCustomers.MiddleInit  
FROM tblCustomers INNER JOIN tblOrders  
ON tblCustomers.CustomerID <> tblOrders.CustomerID  
WHERE (((tblOrders.OrderID)=25));
```

To find out which books haven't been ordered in the last 90 days, create the following two queries.

1. A query to calculate which books have been ordered in the last 90 days (*qxmplBooksInLast90Days*):
 2. SELECT DISTINCTROW tblBooks.ISBNNumber
 3. FROM tblBooks
 4. INNER JOIN (tblOrders
 5. INNER JOIN tblOrderDetails
 6. ON tblOrders.OrderID =
 7. tblOrderDetails.OrderID)
 8. ON tblBooks.ISBNNumber =
 9. tblOrderDetails.ISBNNumber
 10. WHERE (((tblOrders.OrderDate) Between Date()
 11. And Date()-90));

This query uses DISTINCTROW to see each book only once.

12. Build a query on the first query to find the unordered books (*qxmplUnorderedBooks*)
 13. SELECT tblBooks.ISBNNumber, tblBooks.Title
 14. FROM qxmplBooksInLast90Days
 15. RIGHT JOIN tblBooks
 16. ON qxmplBooksInLast90Days.ISBNNumber =

```
tblBooks.ISBNNumber  
WHERE (((qxmplBooksInLast90Days.ISBNNumber)  
Is Null));
```

To see a list of books and the amounts currently on order, sorted by book category, enter the following (*qxmplOrdersByCategoryAndTitle*):

```
SELECT tblCategories.Category, tblBooks.Title,  
tblOrderDetails.Quantity  
FROM tblCategories  
INNER JOIN (tblBooks  
INNER JOIN tblBookCategories  
ON tblBooks.ISBNNumber =  
tblBookCategories.ISBNNumber)  
INNER JOIN tblOrderDetails  
ON tblBooks.ISBNNumber =  
tblOrderDetails.ISBNNumber)  
ON tblCategories.CategoryID =  
tblBookCategories.CategoryID  
ORDER BY tblCategories.Category;
```

ORDER BY Clause

Specifies the sequence of rows to be returned by a SELECT statement or an INSERT statement.

Syntax:

```
ORDER BY {column-name |  
column-number [ASC | DESC]},
```

Notes:

You use column names or relative output column numbers to specify the columns on whose values the rows returned are ordered. (If you use relative output column numbers, the first output column is 1.) You can specify multiple columns in the ORDER BY clause. The list is ordered primarily by the first column. If rows exist for which the values of that column are equal, they are ordered by the next column in the ORDER BY list. You can specify ascending (ASC) or descending (DESC) order for each column. If you do not specify ASC or DESC, ASC is assumed. Using an ORDER BY clause in a SELECT statement is the only means of defining the sequence of the returned rows.

Examples:

To select customers who first did business in August 1996 or earlier and list them in ascending order by zip code, enter the following
(*qxmplCustomersWithOrdersBeforeAug1996*):

INSERT Statement, SELECT Statement, and UNION Query Operator.

```
SELECT tblCustomers.FirstName, tblCustomers.MiddleInit,  
tblCustomers.LastName, tblCustomers.City,  
tblCustomers.PostalCode  
FROM tblCustomers  
WHERE (((#9/1/96#)>=  
(Select Min([OrderDate]) From tblOrders Where  
tblOrders.CustomerID =  
tblCustomers.CustomerID)))  
ORDER BY tblCustomers.PostalCode;
```

Examples:

To specify a field named Customer Last Name in a table named Customer List, use the following:

```
[Customer List].[Customer Last Name]
```

To specify a field named StreetAddress that appears in only one table or query in the FROM clause, enter:

```
StreetAddress
```

To find all stores and all customers in the state of Washington and list them in descending order by zip code, enter the following (*qxmplCustomersAndStoresInWA*):

```
SELECT [tblCustomers].[LastName] & ", " &
      [tblCustomers].[FirstName] &
      IIf(IsNull([tblCustomers].[MiddleInit]),Null,
      " " & [tblCustomers].[MiddleInit] &
      ".") AS Name,
      tblCustomers.City, tblCustomers.PostalCode
FROM tblCustomers
WHERE tblCustomers.StateOrProvince = "WA"
UNION
SELECT tblStores.StoreName, tblStores.City,
      tblStores.PostalCode
FROM tblStores
WHERE tblStores.StateOrProvince = "WA"
ORDER BY 3 DESC;
```

PARAMETERS Declaration

Precedes an SQL statement to define the data types of any parameters you include in the query. You can use parameters to prompt the user for data values or to match data values in controls on an open form.

Syntax:

```
PARAMETERS {[parameter-name]
data-type}, ;
```

Notes:

If your query prompts the user for values, each parameter name should describe the value that the user needs to enter. For example, [Print invoices from orders on date:] is much more descriptive than [Enter date:]. If you want to refer to a control on an open form, use the format

```
[Forms]![Myform]![Mycontrol]
```

To refer to a control on a subform, use the format

```
[Forms]![Myform]![Mysubformcontrol].[Form]![ControlOnSubform]
```

Valid data type entries are as follows:

SQL Parameter Data Type Equivalent Access Data Type

Bit Yes/No

Binary Binary

Byte Byte

Currency Currency

DateTime Date/Time

FLOAT Double

Guid Replication ID

IEEEDouble Double

IEEESingle Single

INT[EGER] Long integer

Long Long integer

LongBinary OLE object

(continued)

SELECT Statement.

continued

SQL Parameter Data Type Equivalent Access Data Type

LongText Memo

REAL Single

Short Integer

SMALLINT Integer

Text Text

Value Value

VARCHAR Text

Example:

To create a parameter query that summarizes the sales and the cost of goods for all items sold in a given month, enter the following (*qxmplMonthSalesParameter*):

```
PARAMETERS [Year to summarize:] Short,  
           [Month to summarize:] Short;  
SELECT tblBooks.ISBNNumber, tblBooks.Title,  
       Format([OrderDate], "mmm", "yy") AS  
       OrderMonth,  
       Sum(CCur(CLng([Quantity]*[SuggPrice])) *
```

```
(1-[Discount])*100)/100) AS OrderTot
FROM tblOrders
  INNER JOIN (tblBooks
  INNER JOIN tblOrderDetails
  ON tblBooks.ISBNNumber =
  tblOrderDetails.ISBNNumber)
  ON tblOrders.OrderID = tblOrderDetails.OrderID
WHERE (((Year([OrderDate])=[Year to summarize:]) AND
  (Month([OrderDate])=[Month to summarize:])))
GROUP BY tblBooks.ISBNNumber, tblBooks.Title,
  Format([OrderDate],"mmmm", ""yy");
```

Next Week we discuss about Predicate: Exist; In ; Like; Null

Forms and Control tools

Forms are the primary interface between users and your Microsoft Access application. Form is one of 7 main objects in MS Access database.

You can design forms for many different purposes.

- **Displaying and editing data.** This is the most common use of forms. Forms provide a way to customize the presentation of data in your database. You can also use forms to change or delete data in your database or add data to it. You can set options in a form to make all or part of your data read-only, to fill in related information from other tables automatically, to calculate the values to be displayed, or to show or hide data on the basis of either the values of other data in the record or the options selected by the user of the form.
- **Controlling application flow.** You can design forms that work with macros or with Microsoft Visual Basic for Applications (VBA) procedures to automate the display of certain data or the sequence of certain actions. You can create special controls on your form, called *command buttons*, that run a macro or a VBA procedure when you click them. With macros and VBA procedures, you can open other forms, run queries, restrict the data that is displayed, execute a menu command, set values in records and forms, display menus, print reports, and perform a host of other actions. You can also design a form so that macros or VBA procedures run when specific events occur for example, when someone opens the form, tabs to a specific control, clicks an option on the form, or changes data in the form. . (*You will learn about using macros and VBA with forms to automate your application later.*)
- **Accepting input.** You can design forms that are used only for entering new data in your database or for providing data values to help automate your application.
- **Displaying messages.** Forms can provide information about how to use your application or about upcoming actions. Access also provides a MsgBox macro action and a *MsgBox* VBA function that you can use to display information, warnings, or error messages. (*You will learn about that later.*)
- **Printing information.** Although you should design reports to print most information, you can also print the information in a form. Because you can specify one set of options when Access displays a form and another set of options when Access prints a form, a form can serve a dual role. For example, you might design a form with two sets of display headers and footers, one set for entering an order and another set for printing a customer invoice from the order etc...

Create the forms object.

The Project Nursery sample database is full of interesting examples of forms. The rest of this project will take you on a tour of some of the major features of those forms. In this week you'll learn how to design and build forms for this database.

Begin by opening your Nursery database in Design view and clicking on the Forms tab in the Database window to see the list of available forms. Now you start to use the wizard to build an application, and later we will create main switchboard form to management to the Database window.)

Headers, Detail Sections, and Footers

You'll normally place the information that you want to display from the underlying table or query in the detail section in the centre of the Form window. You can add a header at the top of the window or a footer at the bottom of the window to display information or controls that don't need to change with each different record.

Subforms

Subforms are a good way to show related data from the "many" side of a one-to-many relationship. For example, the frmSupplier form shown earlier in Nursery shows one Plant and the many Suppliers that the suppliers have provided for Nursery shop. Another good example of a subform is the forms Suppliers show Plant or Plant show suppliers in the project you are doing .

This form looks pretty complicated, but it really isn't difficult to build. Because if the database is well designed, it doesn't take much effort to build the queries that allow the form to display information from five different tables. Most of the work of creating the form goes into selecting and placing the controls that display the data. To link a subform to a main form, you have to set only two properties that tell Access which linking fields to use. In Nursery database you'll build a subform and link it to a main form.

Pop-up Forms

Sometimes it's useful to provide information in a window that stays on top regardless of where you move the focus in your application. You've probably noticed that the default behavior for windows in Microsoft Windows 95 is for the active window to move to the front and for other windows to move behind the active one. One exception is the Help windows. In particular, the "How Do I" windows are designed to float on top of other windows so you can follow the step-by-step instructions when you move the focus to the window in which you're doing the work. This sort of floating window is called a *pop-up window*.

You can create forms in Access that open in pop-up windows. If you open any form in the Nursery application and then choose About from the Help menu, this opens the frmAbout (We will create a form shown in frmAbout, which is designed as a pop-up form. (You'll learn more about how to create custom menus for forms in "The Finishing fro project Nursery.") Switch to the Database window and open the frmAbout form to see how it behaves. Notice that if you click on the Database window behind it, the frmAbout form stays on top. Click the OK button on the form to close it.

Continuous Forms

You can create another type of form that is useful for browsing through a list of records when each record has only a few data fields. This type of form is called a *continuous form*. Rather than showing you only a single record at a time, a continuous form displays formatted records back to back, in the manner of a datasheet.

You can use the vertical scroll bar to move through the record display, or you can click the record number box and the buttons in the lower left corner of the form to move from record to record. As you might guess, the application uses this form to display the results of which information search that returns more than 10 rows.



FIGURE 9-1

this is a continuous form.

Modal Forms

As you add functionality to your application, you'll encounter situations in which you need to obtain some input from the user or convey some important information to the user before Access can proceed. Access provides a special type of form, called a *modal form*, that requires a response before the user can continue working in the application. This dialog box normally opens when you click the button on the main switchboard form and then click the Search button on the resulting Select form, but you can also open the form on which the dialog box is based directly from the Database window. You'll notice that as long as this dialog box is open, ***you can't select any other window or menu in the application.*** To proceed, you must either enter some search criteria and click the Search button or click the Cancel button to dismiss the form.

Using Controls

The information in a form is contained in *controls*. The most common control you'll use on a form is a simple text box. A text box can display data from an underlying table or query, or it can display data calculated in the form itself. You've probably noticed that many controls allow you to choose from among several values or to see additional content. You can also use controls to trigger a macro or a VBA procedure. These controls are discussed in the next five sections.

Option Buttons, Check Boxes, Toggle Buttons, and Option Groups

Whenever the data you're displaying can have only two or three valid values, you can use option buttons, check boxes, or toggle buttons to see or set the value you want in the field. For example, when there are two values, as in the case of a simple Yes/No field, you can use a check box to graphically display the value in the field. A check box that's selected means the value is "Yes," and a check box that's deselected means the value is "No."

To provide a graphical choice among more than two values, you can place any of these controls in a group. Only one of the controls in a group can have a Yes value. If you open the form, which have some option and click the available option buttons, you can see that when you click one button, the previously selected one resets. Access uses the relative numeric value of the control to determine the value in the underlying field. A VBA procedure tests the value when you click one of the controls and expands the form to reveal the date text boxes when appropriate.



FIGURE 9-2.

An option group on the form.

Uses List Boxes and Combo Boxes

When you want to display a list of data values that remains visible, a list box is a good choice. When you view objects in the Database window in detail list mode, you're looking at the tables, queries, forms, reports, macros, or modules in a list box. A list box can show a list of values you entered when you designed the control, a list of values returned by an SQL statement, the value of a field in a table or in a query, or a list of field names from a table or a query. In the example shown in sub form of Plant showing suppliers (the frmPlant form), the list includes the set of names from the tblSupplier table.

When you select a value from the list, you set the value of the control. If the control represents a field in the underlying table or query, you update that field. A list box like this one can use data from more than one field. You can, for example, display the more meaningful what suppliers can supply for you. List box also lets you select multiple entries by holding down the Shift key to select a contiguous range or by holding down the Ctrl key to select several non-contiguous entries. When you click the View button, a VBA procedure evaluates your choices and opens the specific form to display the selected details.

Combo boxes are similar to list boxes. The major difference is that a combo box has a text box *and* a drop-down list. One advantage of a combo box is that it requires space on the form for only one of the values in the underlying list.

Tab Controls

Some time, you should design the way to deal with the need to display lots of information on one form is to use a multiple-page form. Another way to organize the information on a single form is to use the tab control to provide what look like multiple folder tabs that reveal different information depending on the tab chosen—much like the Database window in Microsoft Access provides tabs for Tables, Queries, Forms, Reports, Macros, or Modules. In the Nursery database, we do not need to show some different contents. So we will create a form tab to see how the tab control displays only one of these pieces of information at a time.

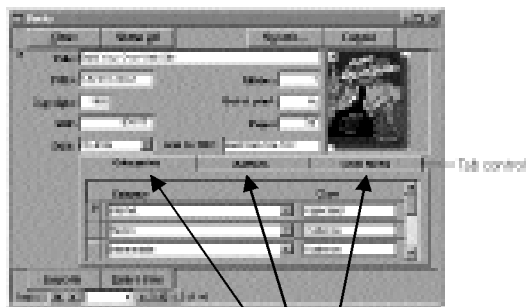


FIGURE 9-5
Information on the tab in the form.

You can click on the tab to see other information. Note that there's no programming required to implement tab selection and data display. (*We will demo for details about how to use the tab control.*)



FIGURE 9-6
Another tab displays different data in a complex form.

OLE Objects

In some forms you see the picture in the form. This picture is stored in a field in the table using Microsoft's object linking and embedding (OLE) technology. Place logo in the top part of the main switchboard form; on the other hand, is a picture that Access has stored as part of the form. The control you use to display a picture or any other OLE object is called an *object frame*. A bound object frame control is used to display an OLE object that is stored in a field in a table. An unbound object frame control is used to display an object that is not stored in a table.

When you include an object frame control on a form and bind the control to an OLE object in the database, you can edit that object by selecting it and then choosing the command at the bottom of the Edit menu that starts the object's application.

If the object is a picture, a graph, or a spreadsheet, you can see the object in the object frame control, and you can activate its application by double-clicking on the object. If the object is a sound file, you can hear it by double-clicking on the object frame control.

Note

You can select a picture and then edit it by choosing the Bitmap Image Object command from the Edit menu and then choosing Edit from the submenu.

When you double-click on the picture or select the picture and choose Bitmap Image Object from the Edit menu and then choose Edit from the submenu, Access starts the Paint application, in which the picture was created. In Windows 95, Paint is an OLE application that can "activate in place. You can still see the Access form, menus, and toolbars, but Paint has added its own tool-bars and menu commands. You can update the picture by using any of the Paint tools. You can paste in a different picture by copying a picture to the Clipboard and choosing the Paste command from Paint's Edit menu. After you make your changes, simply click in another area on the Access form to deactivate Paint and store the result of your edits in the object frame control. If you save the record, Access saves the changed data in your OLE object.

Command Buttons

Another important and useful control is the command button, which you can use to link many forms to create a complete database application. In the Nursery database, for example, most of the forms are linked to the main switchboard, shown in solution 4 from Downloads page, in which the user can click command buttons to launch various functions in the application. The advantage of command buttons is really quite simple they offer an easy way to trigger a macro or a VBA procedure. The procedure might do nothing more than open another form, print a report, or run an action query to update many records in your database. As you'll see when you get to the end of the project Nursery, you can build a fairly complex application using forms, reports, macros, and some simple VBA routines.

Access Tools in Designing Custom Multitable Forms

Expanding Your Form Design Repertoire

The controls that the Form Wizard adds to the forms it creates are only a sampling of the 17 *native control* objects offered by Access 2000. Native controls are built into Access; you also can add various ActiveX controls to Access forms. Until now, you used the Form Wizard to create the labels, text boxes, and subform controls for displaying and editing data in the Personnel Actions table. These three controls are sufficient for creating a conventional transaction processing form.

The remaining **14 controls** described in this chapter let you take full advantage of the Windows graphical user environment. You add controls to the form by using the Access Toolbox. List and combo boxes increase data-entry productivity and accuracy by letting you select from a list of predefined values instead of requiring you to type the value. Option buttons, toggle buttons, and check boxes supply values to Yes/No fields. If you place option buttons, toggle buttons, and check boxes in an option frame, these controls determine the numeric value of the option frame. The Image control supplements the Bound and Unbound Object Frame controls for adding pictures to your forms. Page breaks control how forms print. Access 2000's Tab control lets you create tabbed forms to display related data on forms and subforms in a space saving and more clearly organized fashion. Command buttons can execute Access VBA procedures.

Understanding the Access Toolbox

The Access 2000 Toolbox is based on the Toolbox that Microsoft first created for Visual

Basic. Essentially, the Access Toolbox is a variety of toolbar. You select one of the 20 buttons that appear in the Toolbox to add a native control, represented by that tool's symbol, to the form, so you can select a control, enable or disable the Control Wizards, or add a Microsoft or third-party ActiveX control to the form. When you create a report, the

Toolbox serves the same purpose—although tools that require user input, such as combo boxes, seldom are used in reports.

Control Categories

Three control object categories exist in Access forms and reports:





















- **Bound controls** are associated with a field in the data source for the form or subform. The data source can be a table or query. Bound controls display and update values of the data cell in the associated field of the currently selected record. Text boxes are the most common bound control. You can display the content of graphic objects or play a waveform audio file with a bound OLE object. You can bind toggle buttons and check boxes to Yes/No fields. Option button groups bind to fields with numeric values. All bound controls have associated labels that display the Caption property of the field; you can edit or delete these labels without affecting the bound control.

- **Unbound controls** display data you provide that is independent of the form's or subform's data source. You use the unbound OLE object to add a drawing or bitmapped image to a form. You can use lines and rectangles to divide a form into logical groups or simulate boxes used on the paper form. Unbound text boxes are used to enter data that isn't intended to update a field in the data source but is intended for other purposes, such as establishing a value used in an expression. Some unbound controls, such as unbound text boxes, include labels; others, such as unbound OLE objects, don't have labels. Labels also are unbound controls.
- **Calculated controls** use expressions as their source of data. Usually, the expression includes the value of a field, but you also can use values created by unbound text boxes in calculated control expressions.

The Toolbox

You use the Access Toolbox to add control objects to forms and reports. The Toolbox appears only in Design mode for forms and reports, and it appears only if you click the

Toolbox button on the toolbar or toggle the View, Toolbox menu choice. When the Toolbox is visible, the Toolbox menu choice is checked; Figure under this section, shows the Toolbox in its default mode—a two-column floating toolbar. You can select one of the 17 controls and and three other buttons, whose names and functions are listed in Table with figure

	Select Objects Changes mouse pointer to the object selection tool. Deselects a previously selected tool and returns the mouse pointer to normal selection function. Select Objects is the default tool when you open the Toolbox.
	Control Wizards Turns the Control Wizards on and off. Control Wizards aid you in designing complex controls, such as option groups, list boxes, and combo boxes.
	Label Creates a box that contains fixed descriptive or instructional text.
	Text box Creates a box to display and allow editing of text data.
	frame Creates a frame of adjustable size in which you can place toggle buttons, option buttons, or check boxes. Only one of the objects within an object group frame may be selected. When you select an object within an option group, the previously selected object is deselected.
	Toggle buttons Creates a button that changes from On to Off when clicked. The On state corresponds to Yes (-1), and the Off state corresponds to No (0). When used within an option group, toggling one button On toggles a previously selected button Off. You can use toggle buttons to let the user select one value from a set of
	Option Button Creates a round button (originally called a <i>radio button</i>) that behaves identically to a toggle button. Option buttons are most commonly used within option groups to select between values in a set where the choices are mutually exclusive.
	Check Box Creates a check box that toggles On and Off. Multiple check boxes should be used outside option groups so that
	Combo Box Creates a combo box with an editable text box where you can enter a value, as well as a list from which you can select a value from a set of choices.
	List Box Creates a dropdown list box from which you can select a value. A list box is simply the list portion of a combo box.
	Command Button Creates a command button that, when clicked, triggers an event that can execute an Access VBA event-handling procedure.
	Image Displays a static graphic on a form or report. This is not an OLE image, so you can't edit it after placing it on the
	Unbound Object Adds an OLE object created by an OLE server applications, such as Microsoft Chart or Paint, to a form or report.
	Bound Object Displays the content of an OLE field of a record if the field contains a graphic object.
	Page Break Causes the printer to start a new page at the location of the page break on the form or report. Page breaks don't appear in form or report Run mode.
	Tab Control Inserts a tab control to create tabbed forms. (The tab control looks like the tabbed pages you've seen in the Properties windows and dialogs throughout this book.) Pages of a tab control can contain other bound or
	Subform Adds a subform or subreport to a main form or report, respectively. The subform or subreport you intend to add must exist before you use this control.
	Line Creates a straight line that you can size and relocate. The color and width of the line can be changed by using the
	Rectangle Creates a rectangle that you can size and relocate. The border color, width, and fill color of the rectangle are
	More Controls Clicking this tool opens a scrolling list of ActiveX controls that you can use in your forms and reports. The ActiveX controls available through the More Controls list aren't part of Access; ActiveX controls are supplied as .ocx files with Office 2000, Visual Basic, and various third-party tool libraries.

Customizable Toolbars

At later of this lesson there are some demonstrates that Access toolbars include many shortcut buttons to expedite designing and using Access database objects. Access 2000, like most other con-temporary Microsoft applications, lets you customize the toolbars to your own set of preferences. Access 2000 stores customized toolbars in System.mdw. Toolbars that you create yourself are stored in a hidden system table—MSysCmdbars—in each database.

You can convert conventional floating design tools, such as the Toolbox, to conventional toolbars by the drag-and-drop method. To anchor the Toolbox as a toolbar, also called *docking the toolbar*, follow these steps:

1. Press and hold down the mouse button while the mouse pointer is on the Toolbox's title bar, and drag the Toolbox toward the top of Access's parent window. When the Toolbox reaches the toolbar area, the dotted outline changes from a rectangle approximately the size of the Toolbox into a wider rectangle only as high as a toolbar.
2. Release the mouse button to change the Toolbox to an anchored toolbar positioned below the standard Form Design toolbar.

You can add or delete buttons from toolbars with the Customize Toolbars dialog. To add form design utility buttons to the Toolbox toolbar (whether it's docked or floating), do the following:

1. Choose View, Toolbars, and Customize to display the Customize dialog. Alternatively, click the down arrow at the top left of the Toolbox, choose Add or Remove Buttons from the context menu, and select Customize from the button list.
2. Click the Commands tab, and select Form/Report Design from the Categories list. The optional buttons applicable to form design operations appear in the Commands list,
3. The most useful optional buttons for form design are control alignment and sizing buttons. Press and hold down the mouse button on the Align to Grid command, drag this button to the Toolbox toolbar, and drop it to the right of the Rectangle button. The right margin of the Toolbox toolbar expands to accommodate the new button (if you customize the Toolbox while it's floating, the window expands to accommodate the new button). You can drag the Align to Grid button slightly to the right to create a gap between the new button and the Rectangle button.
4. Repeat step 3 for the Size to Fit, Size to Grid, and Align Left commands, dropping each button to the right of the preceding button. You now have four new and useful design buttons available in your Toolbox.

The Customize dialog for toolbars provides the following additional capabilities:

- To remove buttons from the toolbar, open the Customize dialog; click and drag the buttons you don't want, and drop them anywhere off the toolbar.
- To reset the toolbar to its default design, open the Customize dialog. In the Toolbars list, select the toolbar you want to reset, and click the

Reset button. A message box asks you to confirm that you want to abandon any changes you made to the toolbar.

- To create a button that opens or runs a database object, open the Customize dialog, display the Commands page, and scroll the Categories list to display the All *Objects* items. When, for example, you select All Tables, the tables of the current database appear in the Commands list. Select a table name, such as Supplier, and drag the selected item to an empty spot on a toolbar. The ScreenTip for the new button displays Open Table 'Supplier'. (The same I did demo last week.)
- To substitute text or a different image for the picture on the buttons you add to a tool-bar, open the Customize dialog. Click the button you want to change with the right mouse button to display the button shortcut menu. Click Choose Button Face to display the Choose Button Face dialog. Click one of the images offered, or click the Text check box and type the text you want to display in the text box. To edit the button's image, click Edit Button Image.
- To create a new empty toolbar that you can customize with any set of the supplied buttons you want, open the Customize dialog and select Utility 1 or Utility 2 on the Toolbars page. If there's space to the right of an existing toolbar, the empty toolbar appears in this space. Otherwise, Access creates a new toolbar row for the empty tool-bar. The Utility 1 and Utility 2 toolbars and the changes you make to them are available in any Access database you open.
- To create a custom toolbar that becomes part of your currently open database, open the Customize dialog and click New on the Toolbars page. The New Toolbar dialog appears, requesting a name for the new toolbar (Custom 1 is the default). Access creates a new floating tool window to which you add buttons from the Commands page of the Customize dialog. You can anchor the custom tool window to the toolbar, if you want.
- To delete a custom toolbar, open the Customize dialog, select the custom toolbar on the Toolbars page, and click the Delete button. You are requested to confirm the deletion. The Delete button is disabled when you select one of Access's standard toolbars in the list.

Custom toolbars to which you assign names become part of your database application and are stored in the current database file; they are available only when the database in which they are stored is open. Built-in Access toolbars that you customize are stored in System.mdw and are available in any Access work session.

Creating Bound, Multiline, and Calculated Text Boxes

Access uses the following four basic kinds of text boxes:

- **Single-line text boxes** are usually bound to controls on the form or to fields in a table or query.
- **Multiline text boxes** are usually bound to Memo field types and include a vertical scroll bar to allow access to text that doesn't fit within the box's dimensions.
- **Calculated text boxes** obtain values from expressions that begin with = (equal sign) and are usually a single line. If you include a field value, such as [paScheduledDate], in the expression for a calculated text box, the text box is bound to that field. Otherwise, calculated text boxes are unbound. You cannot edit the value of a calculated text box.
- **Unbound text boxes** can be used to supply values—such as limiting dates—to Access VBA procedures. An unbound text box that doesn't contain a calculation expression can be edited.

The following sections show you how to create the first three types of text boxes.

Adding Text Boxes Bound to Fields

The most common text box used in Access forms is the single-line bound text box that makes up the majority of the controls for the frmSubPlant form of Project. To add a text box that's bound to a field of the form's data source with the field list window, do the following:

1. If necessary, click the Field List button of the toolbar to redisplay the field list.
2. Click and drag the PlantName field in the field list window to the upper-left corner of the form's Detail section. When you move the mouse pointer to the active area of the form, the pointer becomes a field symbol, but no crosshair appears. The position of the field symbol indicates the upper-left corner of the text box, not the label, so drop the symbol in the approximate position of the text box anchor handle.
3. **Drag** the text box by the anchor handle closer to the label, and decrease the box's width.
4. Small type sizes outside a field text box are more readable when you turn the **Bold** attribute on. Select the label beside the text box and click the **Bold** button. (When you select, that means you make the control active)
5. **Drag** the all fields from the list box to the form about 0.75 inch below the label, delete the label, and resize the text box to the approximate dimensions. When you add a text box bound to a memo field, Access automatically sets the Scrollbars property to Vertical.
6. Choose File, **save**, and type the name **frmSubplant** in the Form Name text box of the Save As dialog. Click OK.

Adding a Calculated Text Box and Formatting Date/Time Values

You can display the result of all valid Access expressions in a calculated text box. An expression must begin with = and may use Access functions to return values. As mentioned in the introduction to this section, you can use calculated text boxes to display calculations based on the values of fields. To create a calculated text box that displays the current date and time, do the following:

1. Close the field list and Properties windows. Click the Text Box tool in the Toolbox, and add an unbound text box at the right of the Form Header section of the form.
2. Edit the label of the new text box to read Date/Time:, and relocate the label so that it is adjacent to the text box. Apply the Bold attribute to the label.
3. Type =Now in the text box to display the current date and time from your computer's clock; Access adds a trailing parenthesis pair for you. Adjust the width of the label and the text box to accommodate the approximate length of the text.
4. Click the View button to change to Form view and inspect the default date format, MM/DD/YY HH:MM:SS AM/PM for Sydney area time zone. The default date format doesn't comply with Year 2000 (Y2K) standards unless you've selected Four-Digit Year

Formatting in the General page of the Options dialog.

Using Combo and List Boxes

Combo and list boxes both serve the same basic purpose by letting you pick a value from a list, rather than type the value in a text box. These two kinds of list boxes are especially useful when you need to enter a code that represents the name of a person, firm, or product.

You don't need to refer to a paper list of the codes and names to make the entry. The following list describes the differences between combo and list boxes:

- **Dropdown combo boxes and dropdown lists** consume less space than list boxes in the form, but you must open these controls to select a value. Combo boxes in Access are drop-down lists plus a text box, not traditional combo boxes that display the list at all times. You can allow the user to enter a value in the text box element of the dropdown combo list or limit the selection to just the members in the dropdown list. If you limit the choice to members of the dropdown list (sometimes called a *pick list*), the user can still use the edit box to type the beginning of the list value—Access searches for a matching entry. This feature reduces the time needed to locate a choice in a long list.
- **List boxes** don't need to be opened to display their content; the portion of the list that fits within the size of the list box you assign is visible at all times. Your choice is limited to values included in the list.

In the majority of cases, you bind the dropdown list or combo box to a field so that the choice updates the value of this field. Two-column controls are most commonly used. The first column contains the code

that updates the value of the field to which the control is bound, and the second column contains the name associated with the code. An example of when you want to display multiple-column, dropdown list is most useful is assigning supplier or plants to select one from the list (you are going to do in this week) as you see in the sample Designing Nursery project.

Using the Combo Box Wizard

Designing combo boxes is a more complex process than creating an option group, so you're likely to use the Combo Box Wizard for all the combo boxes you add to forms. Follow these steps to practice to use the Combo Box Wizard to create the dropdown list that lets you, for example select from a list kind of Plants

- 1. Open your database if it is not presently open. create a new form in design mode**
2. Click the Control Wizards button, if necessary, so that the wizards are turned on.
3. Click the Combo Box tool in the Toolbox. The mouse pointer turns into a combo box symbol while on the active surface of the form.(see a black square in the top left the form as I show you to make sure form active)
4. Click the Field List button to display the Field List window.
5. Drag the field to a position at the top and rightmost edge of the form's Detail section, The first Combo Box Wizard dialog appears.
- 6. You want the combo box to look up values in the Plant table, so accept the default option button and then click Next (see Figure 13.20). Your selection specifies Table/Query as the value of the Record Source property of the combo box. The second Combo Box Wizard dialog appears.**
7. Select Plant table from the list of tables in the list. Click Next to reach the third dialog.
8. You need the PlantName and Color fields of the Plant table for your combo box. PlantName serves as the bound field, and your combo box displays the PlantName field. Color is selected in the Available Fields list by default, so click the > button to move Color to the Selected Fields list. Selecte PlantName then click the > button again to move PlantName to the Selected Fields list.
9. Your Combo Box Wizard dialog appears. This selection specifies the SQL SELECT query that serves as the value of the Row Source property and populates the combo box's list.
10. Click Next to reach the fourth dialog. There are two fields, Which you want display in the dropdown list.
11. Accept default option then Click Finish.

Creating Bound, Multiline, and Calculated Text Boxes

Access uses the following four basic kinds of text boxes:

- **Single-line text boxes** are usually bound to controls on the form or to fields in a table or query.

- **Multiline text boxes** are usually bound to Memo field types and include a vertical scroll bar to allow access to text that doesn't fit within the box's dimensions.
- **Calculated text boxes** obtain values from expressions that begin with = (equal sign) and are usually a single line. If you include a field value, such as [paScheduledDate], in the expression for a calculated text box, the text box is bound to that field. Otherwise, calculated text boxes are unbound. You cannot edit the value of a calculated text box.
- **Unbound text boxes** can be used to supply values—such as limiting dates—to Access VBA procedures. An unbound text box that doesn't contain a calculation expression can be edited.

The following sections show you how to create the first three types of text boxes.

Adding Text Boxes Bound to Fields

The most common text box used in Access forms is the single-line bound text box that makes up the majority of the controls for the frmSubPlant form of Project. To add a text box that's bound to a field of the form's data source with the field list window, do the following:

7. If necessary, click the Field List button of the toolbar to redisplay the field list.
8. Click and drag the PlantName field in the field list window to the upper-left corner of the form's Detail section. When you move the mouse pointer to the active area of the form, the pointer becomes a field symbol, but no crosshair appears. The position of the field symbol indicates the upper-left corner of the text box, not the label, so drop the symbol in the approximate position of the text box anchor handle.
9. **Drag** the text box by the anchor handle closer to the label, and decrease the box's width.
10. Small type sizes outside a field text box are more readable when you turn the **Bold** attribute on. Select the label beside the text box and click the **Bold** button. (When you select, that means you make the control active)
11. **Drag** the all fields from the list box to the form about 0.75 inch below the label, delete the label, and resize the text box to the approximate dimensions. When you add a text box bound to a memo field, Access automatically sets the Scrollbars property to Vertical.
12. Choose File, **save**, and type the name **frmSubplant** in the Form Name text box of the Save As dialog. Click OK.

Adding a Calculated Text Box and Formatting Date/Time Values

You can display the result of all valid Access expressions in a calculated text box. An expression must begin with = and may use Access functions to return values. As mentioned in the introduction to this section, you can use calculated text boxes to display calculations based on the values of fields. To create a calculated text box that displays the current date and time, do the following:

5. Close the field list and Properties windows. Click the Text Box tool in the Toolbox, and add an unbound text box at the right of the Form Header section of the form.
6. Edit the label of the new text box to read Date/Time:, and relocate the label so that it is adjacent to the text box. Apply the Bold attribute to the label.
7. Type =Now in the text box to display the current date and time from your computer's clock; Access adds a trailing parenthesis pair for you. Adjust the width of the label and the text box to accommodate the approximate length of the text.
8. Click the View button to change to Form view and inspect the default date format, MM/DD/YY HH:MM:SS AM/PM for Sydney area time zone. The default date format doesn't comply with Year 2000 (Y2K) standards unless you've selected Four-Digit Year

Formatting in the General page of the Options dialog.

Using Combo and List Boxes

Combo and list boxes both serve the same basic purpose by letting you pick a value from a list, rather than type the value in a text box. These two kinds of list boxes are especially useful when you need to enter a code that represents the name of a person, firm, or product.

You don't need to refer to a paper list of the codes and names to make the entry. The following list describes the differences between combo and list boxes:

- **Dropdown combo boxes and dropdown lists** consume less space than list boxes in the form, but you must open these controls to select a value. Combo boxes in Access are drop-down lists plus a text box, not traditional combo boxes that display the list at all times. You can allow the user to enter a value in the text box element of the dropdown combo list or limit the selection to just the members in the dropdown list. If you limit the choice to members of the dropdown list (sometimes called a *pick list*), the user can still use the edit box to type the beginning of the list value—Access searches for a matching entry. This feature reduces the time needed to locate a choice in a long list.
- **List boxes** don't need to be opened to display their content; the portion of the list that fits within the size of the list box you assign is visible at all times. Your choice is limited to values included in the list.

In the majority of cases, you bind the dropdown list or combo box to a field so that the choice updates the value of this field. Two-column controls are most commonly used. The first column contains the code that updates the value of the field to which the control is bound, and the second column contains the name associated with the code. An example of when you want to display multiple-column, dropdown list is most useful is assigning supplier or plants to select one from the list (you are going to do in this week) as you see in the sample Designing Nursery project.

Using the Combo Box Wizard

Designing combo boxes is a more complex process than creating an option group, so you're likely to use the Combo Box Wizard for all the combo boxes you add to forms. Follow these steps to practice to use the Combo Box Wizard to create the dropdown list that lets you, for example select from a list kind of Plants

12. Open your database if it is not presently open. create a new form in design mode

13. Click the Control Wizards button, if necessary, so that the wizards are turned on.
14. Click the Combo Box tool in the Toolbox. The mouse pointer turns into a combo box symbol while on the active surface of the form.(see a black square in the top left the form as I show you to make sure form active)
15. Click the Field List button to display the Field List window.
16. Drag the field to a position at the top and rightmost edge of the form's Detail section, The first Combo Box Wizard dialog appears.

17. You want the combo box to look up values in the Plant table, so accept the default option button and then click Next (see Figure 13.20). Your selection specifies Table/Query as the value of the Record Source property of the combo box. The second Combo Box Wizard dialog appears.

18. Select Plant table from the list of tables in the list. Click Next to reach the third dialog.
19. You need the PlantName and Color fields of the Plant table for your combo box. PlantName serves as the bound field, and your combo box displays the PlantName field. Color is selected in the Available Fields list by default, so click the > button to move Color to the Selected Fields list. Selecte PlantName then click the > button again to move PlantName to the Selected Fields list.
20. Your Combo Box Wizard dialog appears. This selection specifies the SQL SELECT query that serves as the value of the Row Source property and populates the combo box's list.
21. Click Next to reach the fourth dialog. There are two fields, Which you want display in the dropdown list.
22. Accept default option then Click Finish.

Designing Access Report

In previous lessons you learned that you can format and print tables and queries in Datasheet view and that this technique is useful for producing printed copies of simple lists of information. You also learned that you can use forms not only to view and modify data, but also to print data including data from several tables. However, because the primary function of forms is to allow you to view single records or small groups of related records displayed on the screen in an attractive way, forms aren't the best way to print and summarize large sets of data in your database.

This lesson explains when you should use a report instead of another method of printing data, and it describes the features that reports object offer.

Uses of Reports

Reports are the best way to create a printed copy of information that is extracted or calculated from data in your database. Reports have two principal advantages over other methods of printing data:

- Reports can compare, summarize, and subtotal large sets of data.
- Reports can be created to produce attractive invoices, purchase orders, mailing labels, presentation materials, and other output you might need in order to efficiently conduct business.

Reports are designed to group data, to present each grouping separately, and to perform calculations. They work as follows:

- You can define up to 10 grouping criteria to separate the levels of detail.
- You can define separate headers and footers for each group.
- You can perform complex calculations not only within a group or a set of rows but also across groups.
- In addition to page headers and footers, you can define a header and a footer for the entire report.

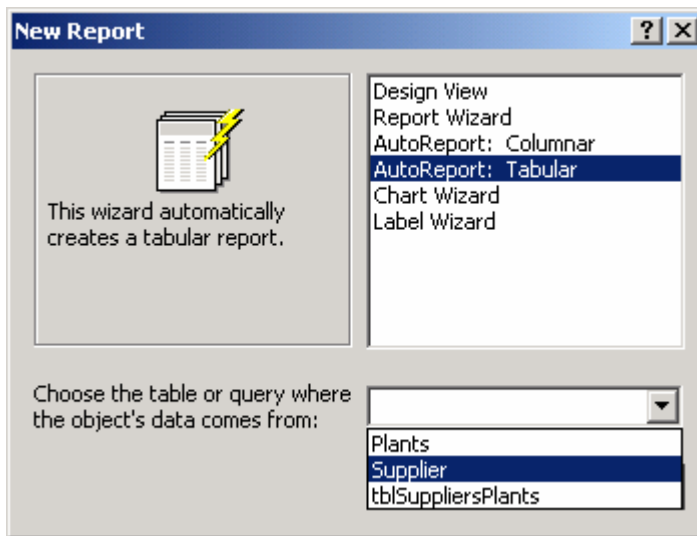
As with forms, you can embed pictures or graphs in any section of a report. You can also embed subreports or subforms within report sections.

A Tour of Reports

You can explore reports by examining the features of the sample reports in the Project database. A good place to start is the rptLetters report, (this integrated with a letter so you will learn more technics in one report. At first you should learn how to create a report by wizard to do that:

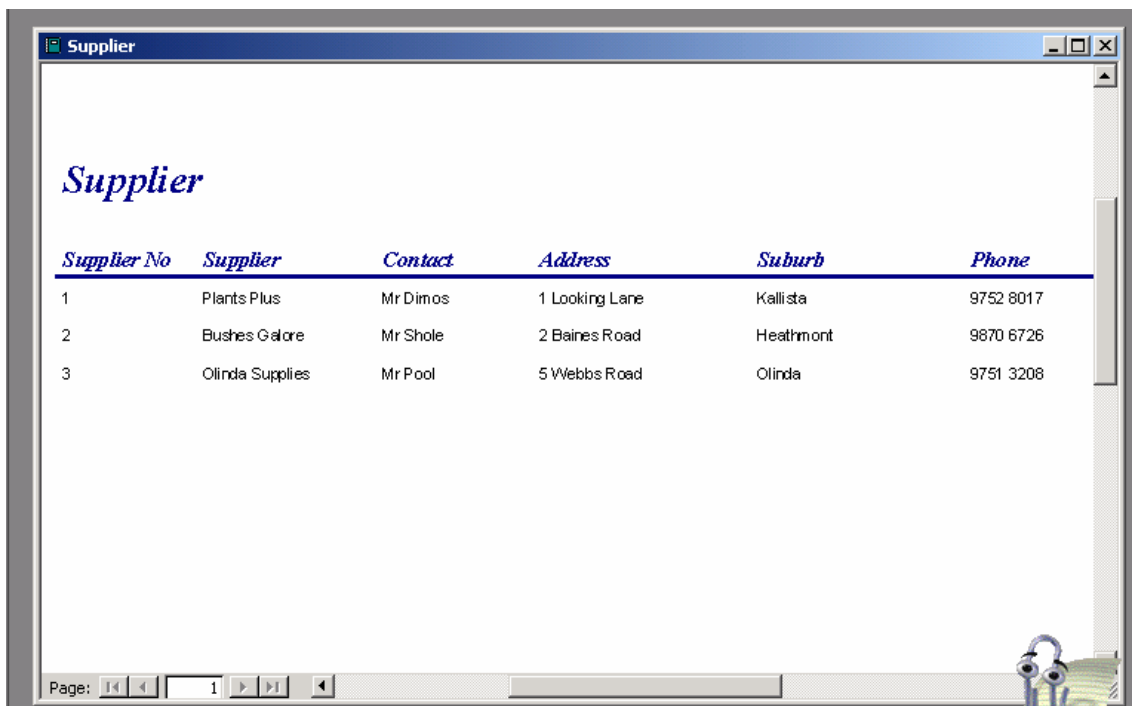
- Open the database in Design View and go to the Database window.
- Click on the Reports tab, and try to create a Report by wizard for any data in a table that you want to report and print out (*Using auto wizard tabular all*

default facilities and click Next button. Demo in class, Do not save this report).



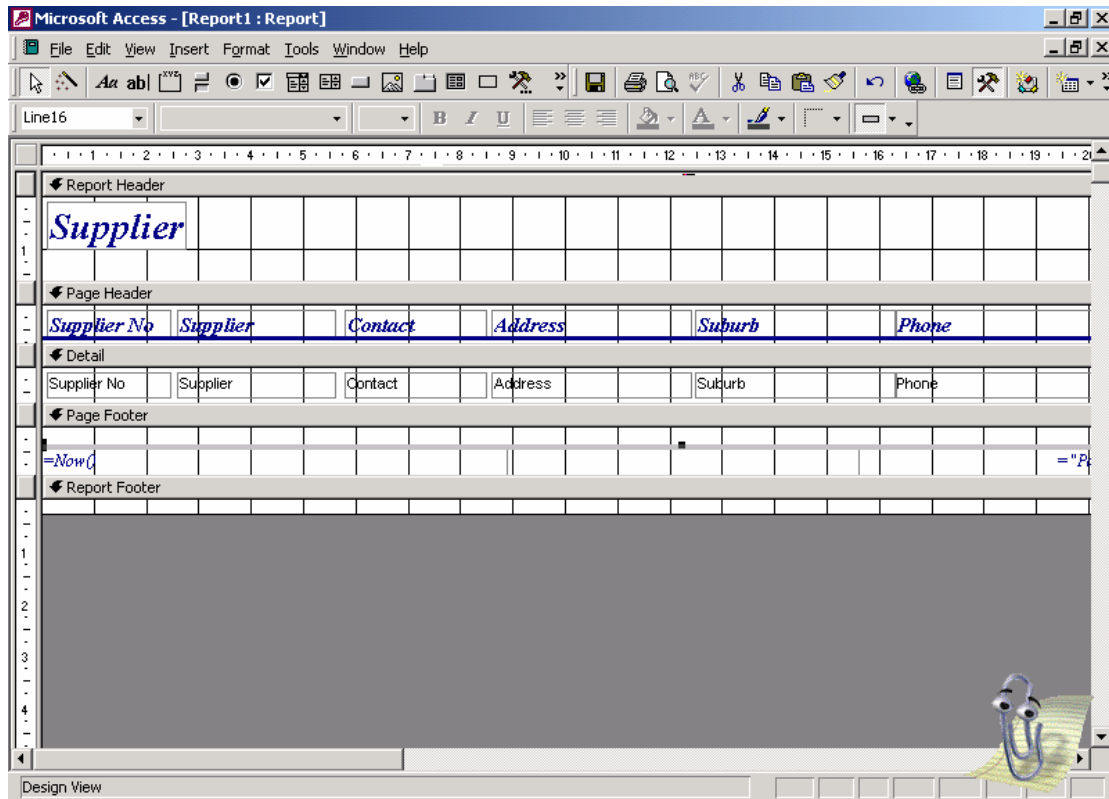
Report Wizard

- Click design view to see how wizard lay out the report. (See figure below)



Print Preview—A First viewing

The report is often based on the query, which brings together information from one or several tables. When the report opens in Print Preview, you'll see a view of the report as a document (see the figure above). But when you open the report in design view to modify or decorate the lay out for the



the report shows information for all stocks you have and come with a letter require the supplier to supply the new product if they've had. Start the application by opening frmFlash form, we will create later in this project), then Command buttons on the main switchboard form, you can choose 2 options to review all Suppliers or a specific Supplier on the Order Options form, and then click the Print button on the form, you'll see a dialog box form that offers to either print the order or print a mailing label. If you print the order, you'll see the Print Preview of only the order you were editing.

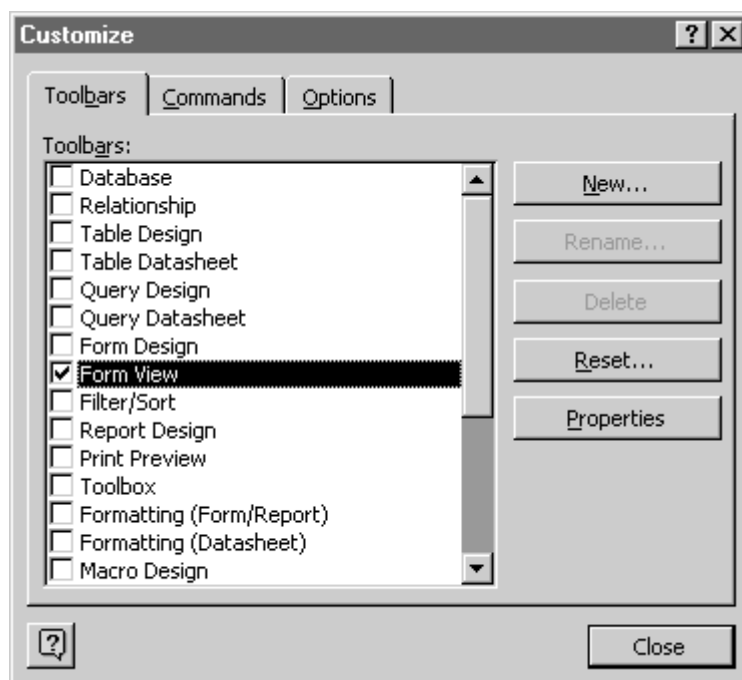
The Finishing Touches

You have almost all the forms and reports you need for the tasks you want to implement in your application, but you need some additional forms to make it easier to navigate and to provide a jumping-off place for all your tasks. Your application could also use a custom menu bar and a custom toolbar for most forms (and perhaps the reports) to add a professional touch. Finally, you need to set the startup properties of your database to let Microsoft Access know how to get your application rolling, and you need to perform a final compile of your VBA code to achieve maximum performance.

When your application is running, you probably won't want or need some of the Access design features. However, you might want some additional toolbar buttons on your form toolbar that provide direct access to commands, such as Save Record and Find Next

Defining a New Toolbar

To begin, open the Toolbars dialog box by choosing Toolbars from the View menu and then choosing Customize from the submenu. You can also click with the right mouse button on any open toolbar to open the toolbar shortcut menu and then choose Customize from that menu. The Customize dialog box with the Toolbars tab selected is shown in Figure below



The Toolbars tab of the Customize dialog box.

On the left side of the Customize dialog box you can see the names of all the built-in toolbars in Access. You can make any of the toolbars visible by selecting the check box next to the toolbar name.

Note:

If you scroll to the bottom of the Toolbars list in the Customize dialog box, you can find an entry for the standard built-in menu bar. As you'll learn later in this project, you can also build custom menus using this customize facility.

If you open one of the built-in toolbars in a context in which the toolbar would not normally be open, the toolbar remains open until you close it. For example, if you open the Customize dialog box while the focus is on the Database window and then open the Form Design toolbar, the toolbar remains open no matter what you are doing in Access. Likewise, if you close a toolbar in a context in which that toolbar is normally open (for example, if you close the Formatting toolbar in a Form window in Design view), that toolbar will remain closed until you open it again within the usual context or from the Customize dialog box.

If you have made changes to one of the built-in toolbars or menu bars, you can select it in the Customize dialog box and click the Reset button to return the toolbar to its default. Access prompts you to confirm this action so that you don't inadvertently erase any custom changes you've made.

Customize Dialog Box Options

If you click on the Options tab in the Customize dialog, you can find check boxes you can use to select large buttons, to display ScreenTips, and to display shortcut keys on ScreenTips. If you're working on a large monitor at a high resolution (1024×768 or 1280×1024), you might find the larger toolbar buttons easier to work with. The large buttons are approximately 50 percent wider and taller than the standard buttons. There's also a button on this tab to "animate" your menus when you open them an interesting effect if you have a fast graphics card on your PC.

Any new toolbar that you define is available only in the database that you had open at the time you created the toolbar. If you want to define a custom toolbar that is available in multiple databases that you work with on your computer, you must modify one of the built-in toolbars. You can use the two "blank" toolbars Utility 1 and Utility 2 to create a custom set of toolbar buttons that is available in any database. For example, you might want to build a "standard" custom form toolbar for all your databases using either Utility 1 or Utility 2. The only drawback to these two toolbars is that you cannot give them custom names. Note, however, that any change you make to a built-in toolbar is effective only for your workstation.

Click the New button in the Customize dialog box to begin defining a new toolbar. Access will prompt you for a name to use for your new toolbar. If you want to follow the example in this section, create a new toolbar in the Nursery Design database and give it a name like Nursery Design Toolbar. You'll see the name appear at the bottom of the Toolbars list, and an empty toolbar in the form of a tiny, gray window will open in the Access workspace. Select your new toolbar in the list, and click the Properties button to open the dialog box (Demonstrate in class)



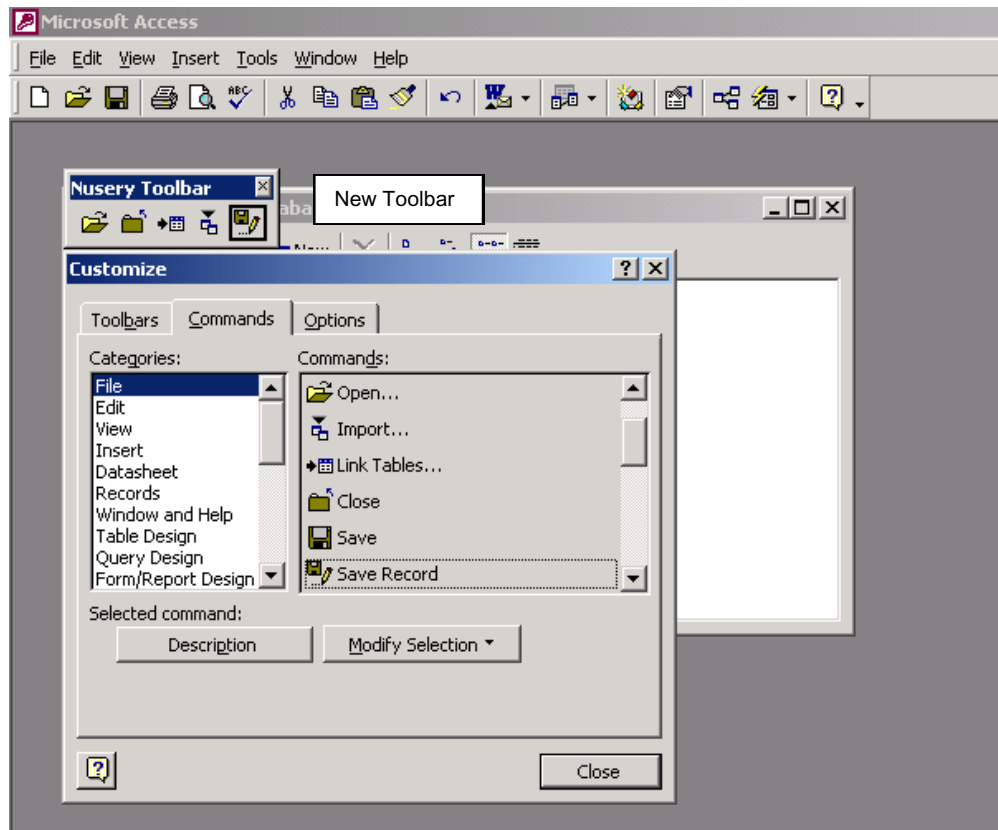
The Toolbar Properties dialog box.

In the Toolbar Properties dialog box, you can select any toolbar from the Selected Toolbar drop-down list. If you select a custom toolbar (not a built-in one), you can rename it and set its type. Because we're building a toolbar first, be sure the Type is set to Toolbar. If you want to, you can position your toolbar and then restrict where the user can move it. For example, you can dock the toolbar at the bottom of your screen and then set Docking to either Can't Change or No Vertical. For custom toolbars, you can check Show On Toolbars Menu to make the toolbar available on the list of toolbars displayed from the View menu's Toolbars command or when the user right-clicks on any toolbar or menu bar. You can't change this option for built-in toolbars. You can pick from additional options for whether this toolbar can be customized, resized, moved, or hidden. Finally, for built-in toolbars, you can click the Restore Defaults button to undo any changes you made. Close this dialog box to go to the next step.

Click the Commands tab in the Customize dialog box to reveal the list of available commands. On the left side of the dialog box is a list of all the command categories that Access provides. The buttons for that category of commands appear on the right side of the dialog box. If you want to see details about a command, select it in the Commands list and click the Description button. Access pops open a description of the command you chose over the bottom of the dialog box.

Customizing Your New Toolbar

After you build a toolbar, you can rearrange the buttons and add dividing lines between them. You can also change the button image, the label in the ScreenTip, and the button style. Finally, and perhaps most importantly, you can define a custom macro or function that you want Access to execute when you click the toolbar button. If you're not still in customize-toolbar mode, choose Toolbars from the View menu, and then choose Customize from the submenu to open the Customize dialog box, shown earlier in

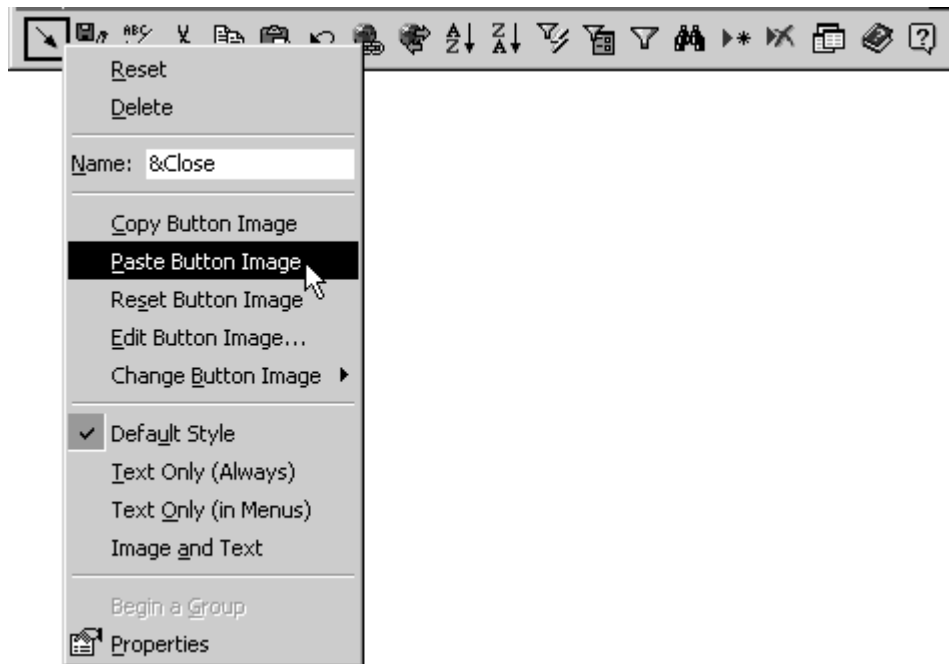


When you open the Customize dialog box, all toolbar buttons become editable. You can:

- Remove any button from any open toolbar (including any built-in toolbar) by clicking the button and dragging it off its toolbar.
- Move any button by clicking it and dragging it to a new location on its toolbar or another toolbar.
- Copy a button from one toolbar to another by holding the Ctrl key while you drag-and-drop.
- Add a button by dragging it from the Commands list to any toolbar.
- Right-click on a button to open a pop-up list of properties that you can modify to change the look of the toolbar button.
- Right-click on a button and choose Properties from the pop-up list to define a custom action for a toolbar button.

Creating Button Images

To make your custom toolbar look just like the Book Form Bar in the Microsoft Press Books database, the first thing you need to do is change the Close button to show an image instead of the word "Close". Right-click on the button to open the shortcut property list, shown in [figure](#) below, and change the style from Text Only to Default Style. Since the Close button has no image defined, you should see the button change to a blank square. If there's another button that has an image you want to use, you can right-click on that button to drop down the shortcut property list and choose Copy Button Image to place that button's bitmap on the Clipboard. Move to the button you want to change, right-click, and choose Paste Button Image.



Pasting a custom button image onto a toolbar button.

If you're good at visualizing button images by setting individual pixel colors on a 16-by-16 square, choose Edit Button Image from the shortcut property list to open a simple design window for the bitmap image. Finally, if you have a 16-by-16 pixel bitmap or icon file that you want to use as a button image, open that file in an image-editing program (for example, Microsoft Paint in the Windows 95 Accessories menu of the Start button), copy the bitmap to the Clipboard, and then choose Paste Button Image from the shortcut menu. In this case, I used an arrow that angles down from upper left to lower right. You can find this image on your sample disc, saved as *Close Arrow.bmp*.

Arranging Buttons

To make buttons easier to use, it's often useful to "cluster" buttons that perform similar functions by adding a dividing line between those clusters. To create a dividing line to the left of any button, right-click on the button and turn on the Begin A Group property for the button near the bottom of the shortcut menu.

Assigning Custom Actions

Last, but not least, you can define a custom macro or function that you want Access to run instead of the built-in action. If you remember from the previous chapter, many of the forms in the project database have a public custom help procedure that can be called from the *FormAssist* function in modUtility. I designed the function this way so that both the AutoKeys F1 macro and any toolbar button could call this one function that subsequently figures out what form is active and specifically calls that form's *FormHelp* subprocedure.

To define a custom action for a toolbar button, right-click on that button to open the shortcut properties menu and then choose Properties at the bottom of the list. You'll see the toolbar control Properties window open, as shown in figure below.



The toolbar control Properties window, where you can define a custom action for the control.

In the Properties window, you can choose any control on the toolbar from the drop-down list at the top of the window. As you can see, you can redefine the caption for controls that display text and the ToolTip (ScreenTip) text if you have ScreenTips enabled. The Shortcut Text property applies only to controls you define on menus. (*See the next section for details.*) You can also change the style of the control and define a help file and help context for the command. (If you've copied a built-in command as shown here, you probably want to leave the help pointers set as they are for the built-in help topic.) The Parameter and Tag properties are advanced settings for programmers who build their own command bars and commands using Visual Basic for Applications, which is not covered in this course

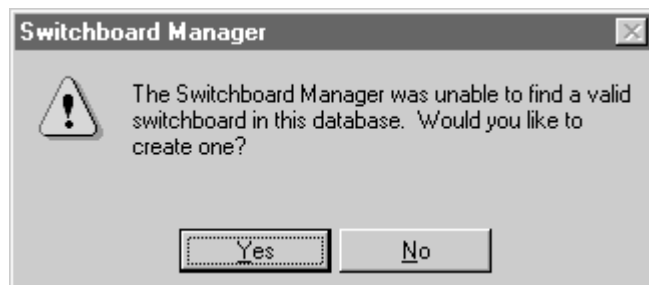
The critical setting in this window is the On Action property. You can set this property to a macro name or specify an expression by preceding the expression with an equals sign (=). This works exactly like the event properties in forms and controls. In this case, you want to call the *FormAssist* function, and the function requires no parameters.

Once you have finished building your toolbar, set the Toolbar property to point to this custom toolbar for any form that you want to open using this toolbar instead of the built-in Form View toolbar. You can also set the Toolbar property for reports to define the toolbar you want displayed when you open the report in Print Preview. The following section discusses how to build custom menu bars using similar techniques.

Using the Switchboard Manager

If your application is reasonably complex, building all the individual switchboard forms you need to provide navigation through your application could take a while. Access has a Switchboard Manager utility that helps you get a jump on building your switchboard forms. This utility uses a creative technique to handle all switchboard forms using a single form. It uses a driver table named Switchboard Items to allow you to define any number of switchboard forms with up to eight command buttons each.

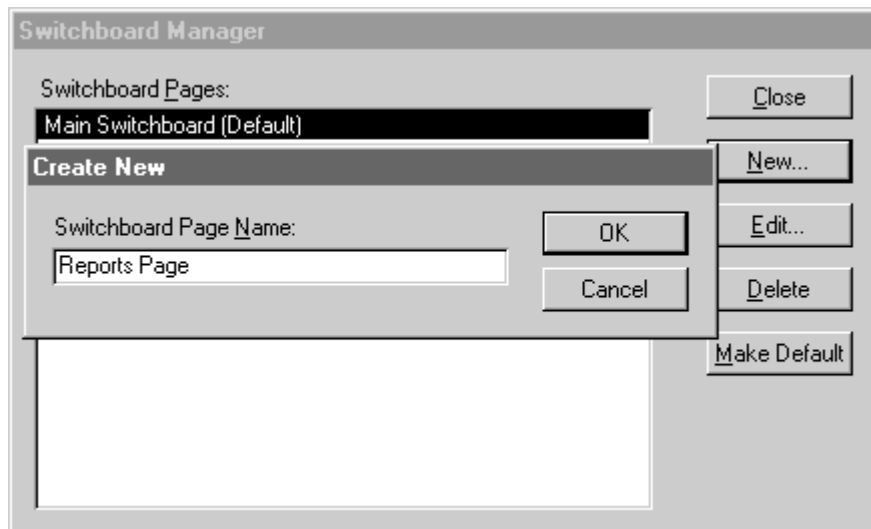
To start the Switchboard Manager, choose Add-ins from the Tools menu and then choose Switchboard Manager from the submenu. The utility will check to see whether you already have a switchboard form and a Switchboard Items table in your database. If you don't have these, the Switchboard Manager displays the message box shown in figure below. Which asks you if you want to build them.



The message box that appears if the Switchboard Manager does not find a valid switchboard form and table in your database.

After the Switchboard Manager builds a skeleton switchboard form and a Switchboard Items table (or after it establishes that you already have these objects in your database), it displays the main Switchboard Manager window. To build an additional switchboard form called a "page" in the wizard, click the New button and enter a name for the new switchboard form in the resulting dialog box, as shown in Figure below Click OK to create the form.

After you create the additional switchboard forms that you need, you can select one in the main Switchboard Manager window and click the Edit button to begin defining actions on the form. You'll see a window similar to the one shown in the background in Figure. Use this window to create a new action, edit an existing action, or change the order of actions. Figure, shows a new action being created. The Switchboard Manager can create actions such as moving to another switchboard form, opening a form in add or edit mode, opening a report, switching to Design view, exiting the application, or running a macro or a VBA procedure. When you create a new action, the Switchboard Manager places a command button on the switchboard form to execute that action.



Adding an additional switchboard form to the main switchboard form.

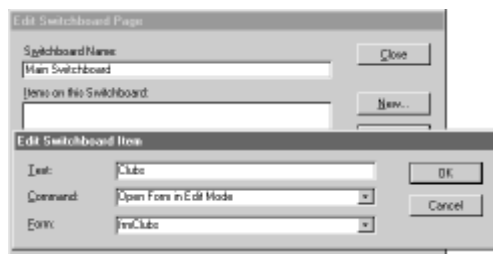


FIGURE 23-17.

Creating a new action on a switchboard form.

On the main switchboard form, you should create actions to open other forms. You should also consider creating an action to exit the application. On each subsequent form, you should always provide at least one action to move back through the switchboard form tree or to go back to the main switchboard form, as shown in Figure below



FIGURE 14

Creating an action to return to the main switchboard form from another switchboard form.

After you finish, the Switchboard Manager saves the main switchboard form with the name *Switchboard*. You can rename this form if you'd like to. If you want to rename the Switchboard Items table, be sure to edit the VBA procedures stored with the switchboard form so that they refer to the new name. You'll also need to change the record source of the form

Setting Startup Properties for Your Database

At this point, you know how to build all the pieces you need to fully implement your database application. But what if you want your application to perform a task automatically when you open your database? If you create a macro named *Autoexec*, Microsoft Access will always run it when you open the database (unless you hold down the Shift key when you open the database). However, a better way to start your application is to specify an opening form in the startup properties for the database. You can set these properties by switching to the Database window and then choosing Startup from the Tools menu. Access opens the Startup dialog box. Click the Advanced button to see the entire dialog box, as shown in Figure below.

