

## 4.1 Introduction: What is ACCESS?

MICROSOFT ACCESS is a **relational database management system** (DBMS). At the most basic level, a DBMS is a program that facilitates the storage and retrieval of structured information on a computer's hard drive.

### 4.1.1 The role of database management systems

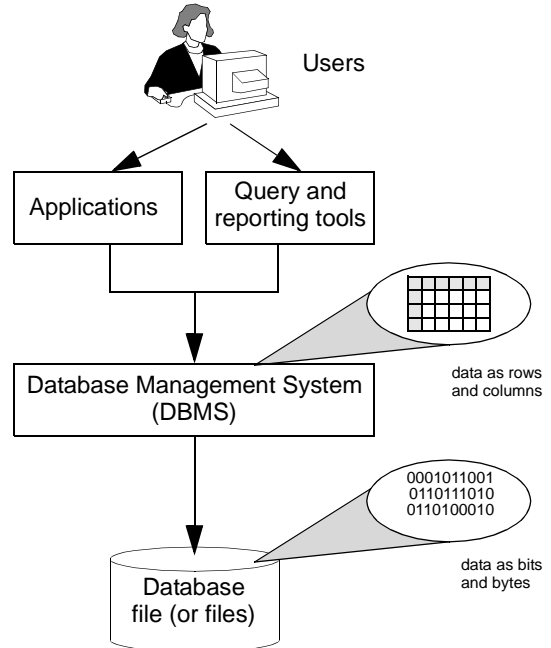
The role of the DBMS within an information system or application is shown in [Figure 4.1](#). As far as we are concerned, a **database** is an amorphous blob<sup>1</sup> of data stored in binary format (ones and zeros) on disk. To read and write to the database, a DBMS is required.



Despite the clear conceptual difference between a database (a blob of binary data) and a DBMS (software to manage one or more databases), we often refer to programs such as ACCESS and ORACLE as

<sup>1</sup> The term blob is used here in its informal sense (e.g., "a blob of gunk"). In database terminology, the acronym BLOB refers a special data type used to store Binary Large Objects (like graphics files or spreadsheet). We discuss different data types in [Lesson 5](#).

FIGURE 4.1: The role of the DBMS.



"databases". Although such usage is sloppy, the context in which the term is used is usually sufficient to eliminate any confusion.



An important feature of the arrows in [Figure 4.1](#) is that users do not interact with the DBMS directly. Instead, they use either a special-purpose application (e.g., a payroll program) or a query and reporting tool (e.g., CRYSTAL REPORTS) to view or modify the data. Similarly, the applications and query/reporting tools do not access the database directly. Instead, all requests for data are made to the DBMS and the DBMS software takes care of reading and writing data to and from the hard disk.

The primary advantage of this layered approach is that the DBMS is used to hide the complexities of low-level disk access from both the users and the application designers. In other words, the DBMS software provides an **abstraction**—instead of thinking about bits and bytes, end-users and designers can think in terms of **tables** of data consisting of **rows** and **columns**.

#### 4.1.2 Inside an ACCESS database file

Although the term “database” typically refers to a collection of related data “tables”, an ACCESS database file includes more than just tables. Indeed, an ACCESS “.mdb” file contains several different types of **database objects**:

- saved **queries** for organizing data;
- **forms** for users to interact with the data on screen;

- **reports** for organizing, summarizing, and printing data; and,
- **macros** and **VISUAL BASIC programs** for extending the functionality of database applications.

All these database objects are stored in a single file named **<file name>.mdb**.



When you are running ACCESS, a temporary “locking” file named **<file name>.ldb** is also created. You can safely ignore the \*.ldb file; everything of value is in the \*.mdb file.



## 4.2 Learning objectives

- identify the version of MICROSOFT ACCESS that you are using
- open and explore an existing database
- learn how to create a new database
- identify the database window and understand how the different database objects fit together
- get help from the on-line help system
- compact a database to save space



## 4.3 Exercises

### 4.3.1 Starting ACCESS

➔ To start ACCESS, you double click the ACCESS icon ( for version 8.0 and 7.0 or  for version 2.0) from within MICROSOFT WINDOWS.

If you are uncertain which version you are using, you can watch for the “splash” screen as the program loads. Alternatively, selecting **Help** → **About ACCESS** from the main menu will tell you everything you need to know.

### 4.3.2 Finding and using an existing database

In order to make the elements of [Figure 4.1](#) more concrete, we will start by finding and opening a sample database application provided by MICROSOFT called “NORTHWIND TRADERS”.

NORTHWIND TRADERS is a fictitious wholesaler of specialty foods to retailers around the world. Given its business environment, it is not surprising that the order entry system application built for NORTHWIND is similar to the one that you will build from scratch in these tutorials.



When you are stuck doing your own projects, it is sometimes convenient to take a look at the NORTHWIND TRADERS application to see “how MICROSOFT does

it.” However, keep in mind that MICROSOFT’S way is only one of many possible ways of accomplishing tasks.

The problem with opening the sample file is that its location depends on the precise manner in which MICROSOFT OFFICE was installed on your computer. To find the file, you can use one of three approaches:

- Look around in the file system directory in which ACCESS/OFFICE is installed.
- Use the “advanced find” feature of the “open file” dialog within ACCESS. This approach is described in more detail below.
- Use the search feature of your operating system. For example, in Windows 95/98/NT/2000, use **Start** → **Search/Find** → **Files or Folders** from the task bar.



Because of the filename limitations in WINDOWS 3.x, the sample application that ships with ACCESS version 2.0 is called **Nwind.mdb**. In more recent versions, it is called **Northwind.mdb**.

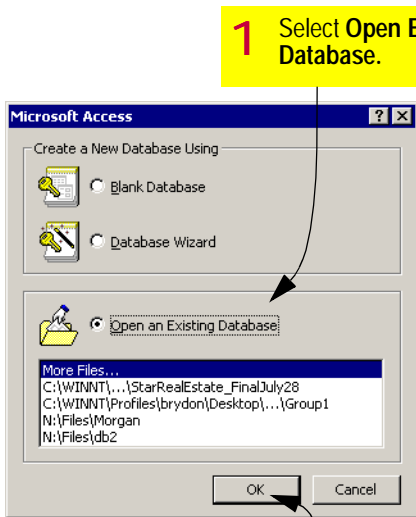
To find the file on your hard disk using the “advanced find” feature, do the following:

- ➔ After starting ACCESS, you should see a screen similar to that shown in [Figure 4.2](#).



Select **Open an Existing Database** and press **OK**.

FIGURE 4.2: Open an existing file from within ACCESS.



If you do not get the dialog in [Figure 4.2](#) or accidentally close the dialog window, you can select **File** → **Open** from the main menu at any time.

You should now have a dialog box with the title “Open”.

- Type “Northwind” into the field labeled **File name** and press the **Advanced** button, as shown in [Figure 4.3](#).
- Select the drive on which ACCESS is installed, indicate that you would like to search in the subfolders of the current folder, and press **Find Now** (see [Figure 4.3](#)).

Hopefully, the “advanced find” utility finds the location of the sample file. If not, you can revert to one of the other approaches above.



When installing ACCESS, you are given the option whether to install the sample databases. If you cannot find the NORTHWIND TRADERS database on your computer it is probably because it was never copied to your hard disk during installation. This problem is easily solved, however: re-run **setup.exe** from the OFFICE CD-ROM and indicate that you want the sample databases installed.

- When you have found the file, highlight it and press **Open**. Alternatively, you can simply double-click the file in the file dialog.

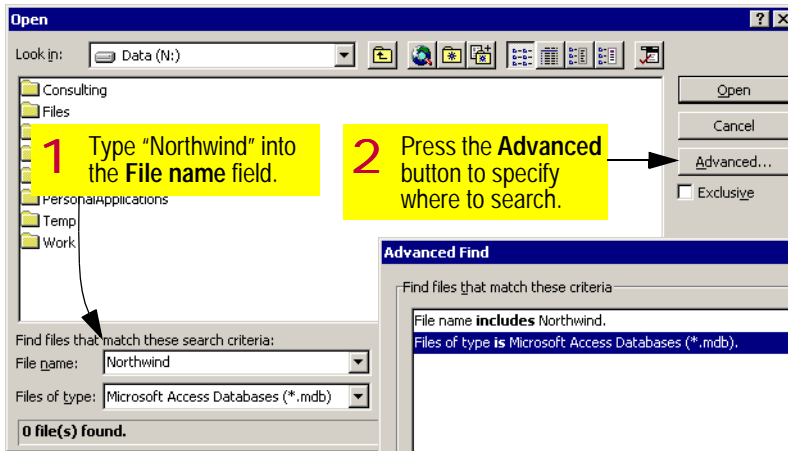
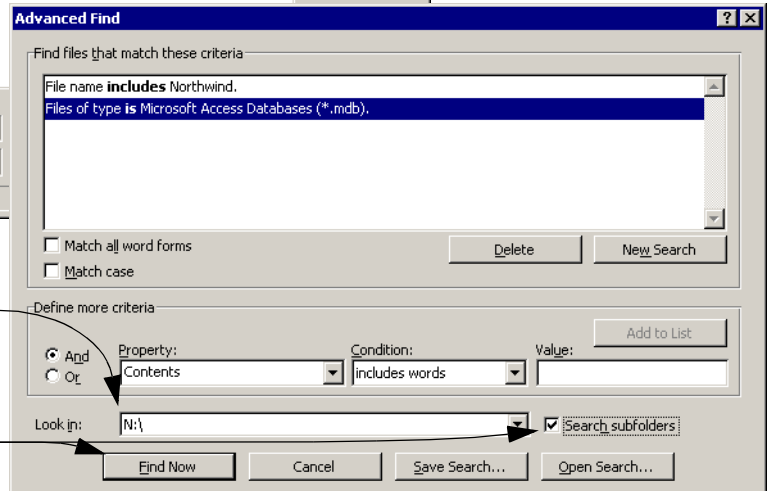


FIGURE 4.3: Use the “advanced find” feature within ACCESS to find the NORTHWIND TRADERS database.



3 Enter the letter of the drive on which ACCESS is installed.

4 Check the **Search Subfolders** check box and press **Find Now**.



If it seems like we are working through these steps in excruciating detail, it is because we are. As the lessons progress, the pace quickens. Eventually, you will simply be told *what* to do, not *how* to do

it. In these early stages, however, we are moving slowly.

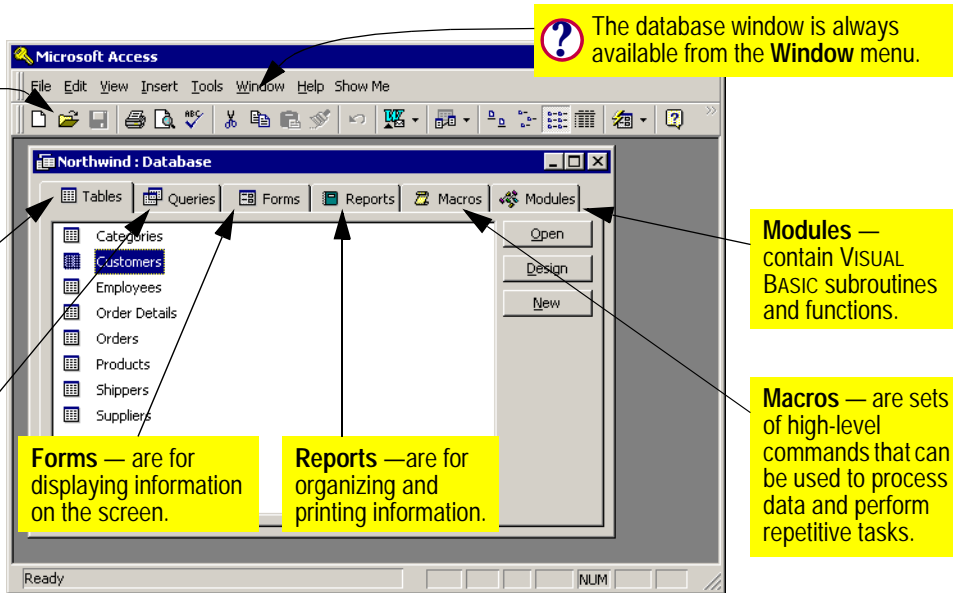


### 4.3.3 Exploring the NORTHWIND TRADERS database

Depending on whether you have ever opened the NORTHWIND TRADERS file, an introductory screen may appear.

➔ Close the introductory screen (if any). You should be left with the **database window**, as shown in [Figure 4.4](#).

FIGURE 4.4: The database window contains all the database objects for a particular application.



#### 4.3.3.1 Tables

In this section, you are going to take a brief look at the **Customers** table. Tables are where all



the data in a database is stored. As such, understanding the structure of tables is critical.

- ➔ Ensure the **Table** tab is selected in the database window and double-click the **Customers** table.

The **datasheet view** shows the data that is stored in the table (see Figure 4.5). Each column (or field) corresponds to an attribute of the entity and each row (or record) corresponds to an instance of the entity.



In the pre-database era, the logical structure of a data file was described in terms of **records** and **fields**. Thus, each customer would have one record in the customer file and the customer's phone number would be stored in the **PhoneNo** field. Although the terms "row" and "column" are preferred in the relational database context, "record" and "field" and are still widely used (even by relational DBMSs such as ACCESS). In these lessons, we will use the terms row/record and column/field interchangeably.

- ➔ Switch to the table's **design view** by selecting **View** → **Design View** from the main menu.

FIGURE 4.5: Open the **Customers** table.

The table fields appear as columns.

Customer ID	Company Name	Contact
ALFKI	Alfreds Futterkiste	Maria Anders
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo
ANTON	Antonio Moreno Taquería	Antonio Moren
AROUT	Around the Horn	Thomas Hardy
BERGS	Berglunds snabbköp	Christina Bergl
BLAUS	Blauer See Delikatessen	Hanna Moos
BLONP	Blondel père et fils	Frédérique Cite
BOLID	Bólido Comidas preparadas	Martín Somme
BONAP	Bon app'	Laurence Lebit
BOTTM	Bottom-Dollar Markets	Elizabeth Linci
BSBEV	B's Beverages	Victoria Ashwo
CACTU	Cactus Comidas para llevar	Patricio Simps
CENTC	Centro comercial Moctezuma	Francisco Cha
CHOPS	Chop-suey Chinese	Yves Moisson

The record for a particular customer appears as a row.

- ➔ Take a brief moment to observe the table's structure, as shown in Figure 4.6.



Although the datasheet view may resemble a spreadsheet, the information in a database is highly structured. For example, you cannot simply move to a



FIGURE 4.6: Switch to the table's design view to observe its structure.

1 Select View → Design View from the main menu to switch to table design view.

2 Note that each field has specific properties, such as data type, length, format, and so on.

Field Name	Data Type	Description
CustomerID	Text	Unique five-character code base
CompanyName	Text	
ContactName	Text	
ContactTitle	Text	
Address	Text	Street or post-office box.
City	Text	

Field Name	Data Type	Description
CustomerID	Text	Unique five-character code base
CompanyName	Text	
ContactName	Text	
ContactTitle	Text	
Address	Text	Street or post-office box.
City	Text	

Field Properties

Property	Value
Field Size	5
Format	
Input Mask	>LLLLL
Caption	Customer ID
Default Value	
Validation Rule	
Validation Text	
Required	No
Allow Zero Length	No
Indexed	Yes (No Duplicates)

A field name can be up to 64 characters long, including spaces. Press F1 for help on field names.

column labeled **PhoneNo** and type in some arbitrary text. The **PhoneNo** column, like other columns, has a predefined structure and data type that is enforced by the DBMS.

➔ Close the **Customers** table. You should be back at the database window.

Although it is possible to make changes to the data in datasheet view, this is not the way in which we will expect our users to interact with the data. Instead, we will use forms to create a



**user interface** that is friendlier and more functional.

#### 4.3.3.2 The user interface

To get a better sense of what a **database application** looks like, and to see how data is actually entered into the database, we can look at the order entry form for NORTHWIND TRADERS.

➤ Click on the **Forms** tab along the top of the database window and double-click the form called **orders**. You may have to resize the form to make it completely visible on your screen.

➤ Note some of the features of the form, as shown in [Figure 4.7](#).

FIGURE 4.7: Explore the order form for NORTHWIND TRADERS.

Special interface elements—in this case, a combo box—make it easier to enter data, such as the customer for the order.

Additional information about the customer pops-up automatically when the customer is selected.

The screenshot shows the 'Orders' form with the following data:

Product	Unit Price	Quantity	Discount	Extended Price
Spegesild	\$12.00	2	25%	\$18.00
Chartreuse verte	\$18.00	21	25%	\$283.50
Rössle Sauerkraut	\$45.60	15	25%	\$513.00
			0%	

Summary section:

Subtotal:	\$814.50
Freight:	\$29.46
Total:	\$843.96

Check boxes make it easy to select shipping method.

Sub-totals and totals are calculated automatically and are shown on the form.

Buttons are used to execute actions and show additional information.



- Close the order form.

#### 4.3.3.3 Queries and reports

The final type of database object we are going to consider on this whirlwind tour is a report. Reports are generally created in two steps. In the first step, a query is used to organize and filter the information that is required for the report. In the second step, a report template is created to display the information in a format fit for human consumption.

- Click on the Reports table along the top of the database window and double-click the report called **Products by Category**.
- Click on the report to zoom in and zoom out. Note that the report breaks out NORTHWIND's products and inventory out by product category and formats the information into columns.
- Close the report. Also, close the database window for NORTHWIND TRADERS.

Here ends our tour of the sample application. Hopefully, you now have a better idea of what a desktop database can do. Before we end this lesson, you will create a database file of your own and learn about some basic housekeeping issues.

#### 4.3.4 Creating a new database

- Select **File** → **New Database** from the main menu.
- In the “New” dialog box, select the **General** tab and “Blank Database”.



The “New” dialog does not exist in version of ACCESS prior to version 8. It permits you select the **Databases** tab and chose from a number of database design wizards. As indicated previously, a database design wizard may sound interesting, but its use contributes nothing to our pedagogical objectives.

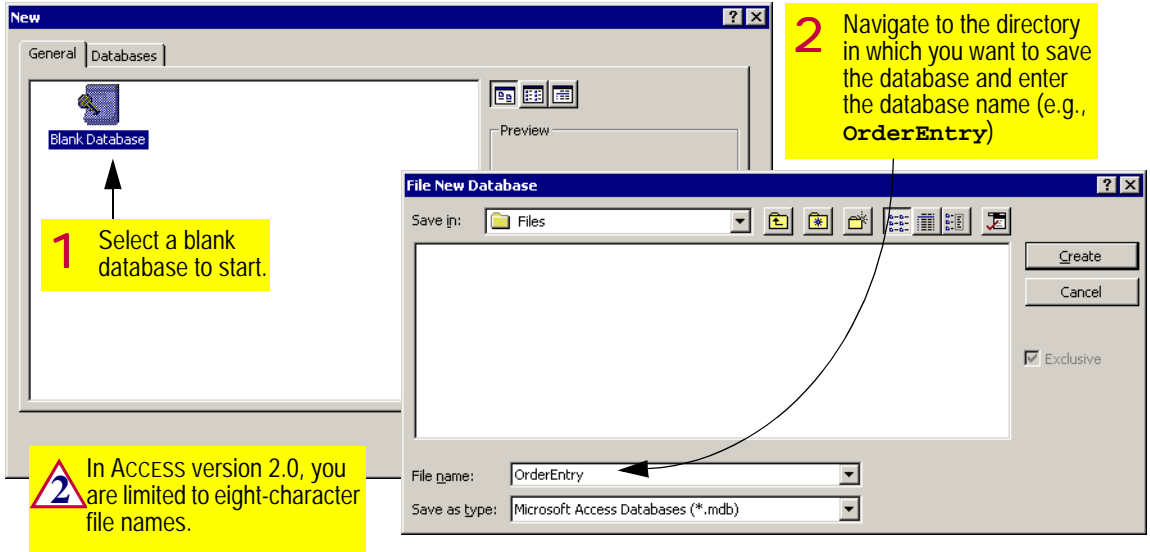
- Navigate to the directory in which you want your new database file to be saved and enter a name (e.g., **OrderEntry.mdb**) for the file, as shown in [Figure 4.8](#).

#### 4.3.5 Using the on-line help system

Commercial software relies increasingly on on-line help and documentation in lieu of printed manuals. As a consequence, experience navigating on-line help systems is essential for learning any new software. In this section, you will use ACCESS' on-line help system to learn how to perform a rather esoteric (but useful) task in ACCESS: compacting a database.



FIGURE 4.8: Create a new database and save it in a subfolder.



The term “on-line” is used here in the general sense to mean “computer-based”. The “connected to the Internet” connotation of the word is a relatively recent etymological development.

- ➔ Press **Help** → **Contents and Index** to invoke the on-line help system. Use the index to find information on compacting a database.

- ➔ Skim the contents of the entry.

### 4.3.6 Compacting your database

As the on-line help system points out, ACCESS database files can become highly fragmented and grow to become much larger than you might expect given the amount of data they contain. Compacting your database from time



to time eliminates fragmentation and can dramatically reduce the disk space requirement for the database.

➔ Follow the directions provided by the on-line help system to compact your database.



Since your database is completely empty at this point, compacting has little effect. The pedagogical objective of this section is to create awareness of the compacting feature so that you know what to do when your database balloons to a couple of megabytes for no apparent reason.



The compacting utility in MICROSOFT ACCESS helps use space within the “.mdb” file more efficiently. Compacting has nothing to do with general-purpose “compression” software such as WINZIP or PKZIP.

## 4.4 Discussion

### 4.4.1 Relationship between ACCESS and other databases

MICROSOFT ACCESS is a desktop DBMS. It is designed for use by a small number of users on a single machine and the design emphasis is on convenience rather than raw performance. Other notable desktop DBMSs include MICROSOFT

FOXPRO, BORLAND/INPRISE dBASE and PARADOX, LOTUS APPROACH, and FILEMAKER PRO.<sup>1</sup>

Because of its emphasis on convenience, ACCESS brings all the components in Figure 4.1 into a single, integrated package. Specifically, ACCESS provides:

- a database engine that takes care of reading and writing data to disk, optimizing queries, managing security, and so on;
- query and reporting tools for extracting and formatting specific data; and,
- a complete environment for creating and executing custom applications.

In contrast, many industrial-strength client/server database such as ORACLE, MICROSOFT SQL SERVER, and IBM DB2 are strictly database management systems. Although these vendors also sell supplemental tools for every imaginable query and reporting requirement, the supplemental tools are best seen as distinct products. In most cases, industrial-strength DBMSs are invisible to users and are accessed over a network by custom applications written in programming languages like C++ or VISUAL BASIC.

---

<sup>1</sup> The multiple offerings from MICROSOFT and BORLAND/INPRISE are the results of acquisitions over time. Large and vocal “installed bases” (rather than any substantive technological differences) appear to be the primary barrier to product-line rationalization.



Another class of DBMS software that is worth noting is **open-source** client/server databases such as `MySQL`, `Postgres`, and `Interbase`. These databases are freely available under open-source licenses and in some cases offer functionality and power that rival commercial products. `Interbase`, for example, was formerly a commercial product selling for about US\$10K per copy. Now it is free.

All three of the open-source databases mentioned above run on `Linux`, an open-source operating system. Thus, it is possible to run a sophisticated database server using software that costs nothing. The downside of open-source software in general is that it tends to be aimed at people who know what they are doing.

## 4.4.2 The many faces of ACCESS

`Microsoft` typically incorporates as many features as possible into its products. For example, the `ACCESS` product contains the following elements:

- a relational database system that supports two industry-standard query languages: **Structured Query Language (SQL)** and **Query By Example (QBE)**;
- a full-featured procedural programming language—essentially a subset of `Visual Basic`,

- a simplified procedural macro language that is unique to `ACCESS`;
- a rapid application development environment complete with visual form and report development tools;
- a sprinkling of object-oriented extensions; and,
- various wizards and builders to make development easier.

For new users, these “multiple personalities” can be a source of enormous frustration. The problem is that each personality is based on a different set of assumptions and a different view of computing. For instance,

- the relational database personality expects you to view your application as sets of data;
- the procedural programming personality expects you to view your application as commands to be executed sequentially;
- the object-oriented personality expects you to view your application as a collection of independent objects with state, methods, and events.

`Microsoft` makes no effort to provide an overall logical integration of these personalities (indeed, it is unlikely that such an integration is possible). Instead, it is up to you as a developer



to pick and choose the best approach to implementing your application.

Since there are often several vastly different ways to implement a particular feature in ACCESS, recognizing the different personalities and exploiting the best features (and avoiding the pitfalls) of each are important skills for ACCESS developers.

The advantage of these multiple personalities is that it is possible to use ACCESS to learn about an enormous range of information systems concepts without having to interact with a large number of “single-personality” tools, for example:

- ORACLE for relational databases
- POWERBUILDER for rapid applications development,
- SMALLTALK or JAVA for object-oriented programming.

Keep this advantage in mind as we switch back and forth between personalities and different computing paradigms.

### 4.4.3 Developing applications in ACCESS

In general, there are two basic approaches to developing an information system:

- in-depth systems analysis, design, and implementation,

- rapid prototyping (in which analysis, design, and implementation are done quickly and iteratively).

ACCESS provides a number of features (such as graphical design tools, wizards, and a high-level macro language) that facilitate rapid prototyping. Since you are going to build a small system and since time is limited, you will use a rapid prototyping approach to build your application. That is, rather than undertake an elaborate requirements specification and design phase, you are going to sketch some data models and dive right into building a prototype. You will then test the prototype, make changes, and repeat the cycle until you have developed a good solution to the business problem.

During the iterative development process, prototypes often become full of junk and remnants of failed approaches. In addition, prototypes do not have sophisticated error-handling code or internal documentation. Thus, it is often good practice to throw the prototype away at the end of the development process and use what you have learned during prototyping to build a clean system from the bottom up.



As you will discover, starting over from scratch is not as bad as it sounds—once you know what you are doing, you can rebuild a system very quickly.



## 4.5 Application to the project

➔ Play with the NORTHWIND TRADERS application until you have a good idea of what it does.



If you are worried about damaging or corrupting the sample database, you can make a copy under a different name (e.g., **myNorthwind.mdb**) and work with the copy instead.

➔ Ensure you have drawn an ERD (see [Lesson 3](#)) to model the data requirements of the system. The ERD will help you design tables in the next lesson.



If you take a closer look at the NORTHWIND application, you will notice that it contains more detail than the simple ERD we created in [Lesson 3](#). For example, the **Products** table contains information about, reorder quantity, whether the product has been discontinued, the supplier of the product, and so on. Although you are free to add additional tables and attributes to your kitchen supply application, the skills and principles are the same regardless of whether you build tables that have five fields or 50 fields. As a consequence, I recommend that you accept the unrealistically narrow scope of the

project in order to minimize the amount of time you waste repeating the same simple skills.