Chapter 5

Organizing and Manipulating the Data in Databases

INTRODUCTION

NORMALIZATION

First Normal Form

Second Normal Form

Third Normal Form

VALIDATING THE DATA IN DATABASES

Database Management Systems

Data Validation

EXTRACTING DATA FROM DATABASES: DATA MANIPULATION LANGUAGES (DMLS)

Creating Select Queries

Creating Action Queries

Guidelines for Creating Queries

Structured Query Language (SQL) and HyperText

Sorting, Indexing, and Database Programming

Online Analytical Processing (OLAP) and Data Mining

OBJECT-ORIENTED DATABASES, MULTIMEDIA DATABASES, AND DATA WAREHOUSES

Object-Oriented and Multimedia Databases

Data Warehouses

AIS AT WORK—RUN YOUR DATA WAREHOUSE LIKE A FINE RESTAURANT

SUMMARY

KEY TERMS YOU SHOULD KNOW

TEST YOURSELF

DISCUSSION QUESTIONS

PROBLEMS

CASE ANALYSES

Swan's Supplies

Bonnie P Manufacturing Company

Clifford Cohen University

BSN Bicycles II

Furry Friends Foundation II

REFERENCES AND RECOMMENDED READINGS

ANSWERS TO TEST YOURSELF

After reading this chapter, you will:

1. *Understand* the process of normalization.
2. *Be familiar with* database techniques for validating data inputs.
3. *Understand* the importance of extracting data from databases and AIS uses of such extractions.
4. *Know how to* create simple and multi-table queries using Microsoft Access.
5. *Understand* object-oriented and multimedia databases.
6. *Be familiar with* data warehouses and their uses in accounting applications.

"Why let valuable information remain untouched within your company databases? Data mining has a wide range of uses, from searching for unknown patterns in company databases to identifying fraudulent activity."

Raymond Landry Jr., et. al., "Grab your Picks and Shovels: There's Gold in Your Data" *Strategic Finance* Vol. 85, No. 7 (January 2004), p. 28.

INTRODUCTION

In theory, system developers should first design databases, using the techniques described in Chapter 4, and then construct them later. In practice, organizations create many commercial databases from collections of preexisting manual files, nonintegrated computerized files, personal or informal files, or the databases of acquired companies. Thus, the key databases of a company are typically in a state of continuous evolution, reevaluation, and revision.

The previous chapter introduced the concept of databases and discussed data modeling—the process of designing database records and tables. This chapter focuses on ways to use databases in AISs. We begin with a discussion of normalization and then look at several methods of querying Access database tables in practice. Finally, we examine a few special types of databases: object-oriented databases, multimedia databases, and data warehouses.

NORMALIZATION

Chapter 4 made clear that, without advanced planning, accounting data are likely to wind up in **flat files**—i.e., files with no sequence or order to them, except perhaps a chronological sequence. An example would be a file in which a professor enters the student grades of an examination in random order. Flat files make it almost impossible to find a particular record easily (because the records are not stored systematically), to link files to one another to provide information from related records, or to store file data efficiently. The databases of most AISs require more discipline than this.

Normalization is the process of examining and arranging file data in a way that helps avoid problems when organizations use or modify them later. Consider, for example, the long waiting period that a university student may encounter when applying for financial aid. If student records are not integrated in a complete database, the scholarship director will have to request a copy of the student's transcript from one source, financial records from other sources, and perhaps additional information from other sources—a time-consuming process.

In commercial applications, an organization generally knows what data are involved in a specific application. The challenge is to organize the data intelligently. Normalization is a methodology for accomplishing this objective. There are several levels of normalization, but we shall only examine the first three of them: first normal form, second normal form, and third normal form.

Social Security Number	Last Name	First Name	Phone Number	License Plate State	License Plate Number	Ticket Number	Date	Code	Fine
123-45-6789	Curry	Dorothy	(916)358-4448	CA	123 MCD	10151	10/15/10	A	$10
						10152	10/16/10	B	$20
						10121	11/12/10	B	$20
134-56-7783	Macon	Richard	(916)563-7865	CA	253 DAL	10231	10/23/10	C	$50
						12051	12/5/10	A	$10

FIGURE 5-1 A set of unnormalized parking ticket data.

First Normal Form

A database is in **first normal form (1 NF)** if all the record's attributes (data fields) are well defined and the information can thus be stored as a flat file. Interestingly enough, not every set of data automatically satisfies this requirement. For example, Figure 5-1 shows a set of university parking ticket data with repeating groups in its rightmost four columns. (Real parking tickets will contain many more data fields than shown here, but we will keep things simple to focus on normalization tasks.) Databases cannot store more than one value in the same data field (i.e., column) of the same record, so we must do something to overcome this limitation.

One solution to this problem is to use a separate record to store the information for each parking ticket. Figure 5-2 illustrates the results. For this file, the ticket number serves as the primary key. There are no repeating groups for any one column, so we can now store these records in a conventional computer file. Our data are now in first normal form.

Although we now have a well-defined file of student data, several problems remain. One difficulty is a large amount of *data redundancy* (i.e., the fact that much of the information in this file is repetitive). Another problem is that we have created an *insertion anomaly*—the fact that this database only stores information about students with parking tickets. Students with registered cars but no parking tickets will have no records in this file—a difficulty if school administrators also want to use this file for car-registration purposes. A third problem is a *deletion anomaly*—the fact that those students who pay their ticket fines will no longer have a car registration record on file.

Social Security Number	Last Name	First Name	Phone Number	License Plate State	License Plate Number	Ticket Number	Date	Code	Fine
123-45-6789	Curry	Dorothy	(916)358-4448	CA	123 MCD	10151	10/15/10	A	$10
123-45-6789	Curry	Dorothy	(916)358-4448	CA	123 MCD	10152	10/16/10	B	$20
123-45-6789	Curry	Dorothy	(916)358-4448	CA	123 MCD	10121	11/12/10	B	$20
134-56-7783	Mason	Richard	(916)563-7865	CA	253 DAL	10231	10/23/10	C	$50
134-56-7783	Mason	Richard	(916)563-7865	CA	253 DAL	12051	12/5/10	A	$10

FIGURE 5-2 The data of Figure 5-1 in first normal form.

Second Normal Form

To solve these problems, let us redesign our database into **second normal form (2 NF)**. A database is in second normal form if it is in first normal form and all the data items in each record depend on the record's primary record key. To satisfy this requirement for our student-parking ticket example, let us split our student information into two files—a ''Car Registration File'' and a ''Ticket File''—as shown in Figure 5-3. This approach not only results in a more efficient design but also eliminates much of the first file's data redundancy.

In our new Car Registration File (or table), what should serve as the primary key? At first glance, you might guess ''social security number.'' If students are only able to register one car, then this choice might be satisfactory. If students can register more than one car, then it makes more sense to use the license plate number as the primary key. Remember: the primary key must uniquely identify a record, and this would not be possible if one person (with one Social Security number) had two records in this table. Finally, we note that in an actual application, it is more likely that the license's ''State'' and ''Number'' data fields together would serve as the primary record key, but again we shall keep things simple here.

What about a primary key for our new Ticket File? In this table, the ''ticket number'' serves this purpose, while the student's license plate number serves as the foreign key. Again, recall from Chapter 4 that a foreign key enables a database to link appropriate records together—for example, to trace a particular parking ticket to the car's registered owner. It also enables database users to answer such questions as ''Does a particular student have any outstanding parking tickets?''

Car Registration File

Social Security Number	Last Name	First Name	Phone Number	License Plate (primary key) State	License Plate (primary key) Number
123-45-6789	Curry	Dorothy	(916)358-4448	CA	123 MCD
134-56-7783	Mason	Richard	(916)563-7865	CA	253 DAL
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

Ticket File

Ticket Number (primary key)	License Plate (foreign key) State	License Plate (foreign key) Number	Date	Code	Fine
10151	CA	123 MCD	10/15/10	A	$10
10152	CA	123 MCD	10/16/10	B	$20
10231	CA	253 DAL	10/23/10	C	$50
10121	CA	123 MCD	11/12/10	B	$20
12051	CA	253 DAL	12/5/10	A	$10
.	.	.	.	.	.
.	.	.	.	.	.

FIGURE 5-3 The data of Figure 5-2 in second normal form.

Third Normal Form

Although we are making headway in our database design, our goal is to create a database that is in **third normal form (3 NF)**. A database is in third normal form if it is in second normal form and contains no **transitive dependencies**. This means that the same record does not contain two data fields in which data field *A* determines data field *B*. A simple example is a student record that stores both the number of credits taken at a university and his or her class standing of ''junior'' or ''senior'' (assuming ''credits taken'' dictates ''class standing''). The Ticket File of Figure 5-3 also suffers from this problem because the ticket code data field (e.g., a code of ''A'') determines the amount of the fine (e.g., ''$10'').

One way to solve this problem is to store the data for parking fines in a new ''Parking Violations Code File'' as shown in Figure 5-4. This enables us to eliminate the redundant information (the Fine data field) in the Ticket File of Figure 5-3 and streamline our data. Figure 5-4 illustrates the results. The ticket codes (A, B, and so forth) in the Ticket File

Car Registration File

Social Security Number (primary key)	Last Name	First Name	Phone Number	License Plate State (primary key)	License Plate Number (primary key)
123-45-6789	Curry	Dorothy	(916)358-4448	CA	123 MCD
134-56-7783	Mason	Richard	(916)563-7865	CA	253 DAL
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.

Ticket File

Ticket Number (primary key)	License Plate State (foreign key)	License Plate Number (foreign key)	Date	Code
10151	CA	123 MCD	10/15/10	A
10152	CA	123 MCD	10/16/10	B
10231	CA	253 DAL	10/23/10	C
10121	CA	123 MCD	11/12/10	B
12051	CA	253 DAL	12/5/10	A
.	.	.	.	.
.	.	.	.	.

Parking Violations Code File

Code (primary key)	Fine	Explanation
A	$10	meter expired
B	$20	parking in no-parking zone
C	$50	no parking sticker
.	.	.
.	.	.

FIGURE 5-4 The data of Figure 5-3 in third normal form.

serve as foreign keys that link the information in the Ticket File to an entry in the Parking Violations Code File. We now have a database in third normal form.

Chapter 4 noted that databases tend to become complicated, with multiple files that are linked together with foreign keys. The database in Figure 5-4, for example, is more complex than our data in Figure 5-1, but it is also more efficient. For example, this database design will allow its users to (1) store the car registration information of all students, even if they do not have any parking tickets, (2) alter a student's name, phone number, or license plate by altering only one record in the Car Registration file—not several of them, as would be required using the flat file of Figure 5-2, and (3) easily change the fine amount for a parking ticket. Finally, this database design allows us to eliminate a lot of redundant information and therefore makes file storage more efficient.

VALIDATING THE DATA IN DATABASES

After data have been normalized, it remains to actually create database tables and records. Typically, this is done with a database management system.

Database Management Systems

A **database management system (DBMS)** is a separate software system that enables users to create database records, delete records, access specific information, query (select subsets of) records for viewing or analysis, alter database information, and reorganize records as needed. This section of the chapter explains how to perform some of these tasks in greater detail.

A DBMS is not a database. Rather, a DBMS is a set of separate computer programs that enable users to create, modify, and utilize database information efficiently, thus allowing businesses to separate their database system operations from their accounting system applications. This enables organizations to change record structures, query and report formats, video displays, and similar items without also having to reprogram the accounting software that accesses these database items. It also enables businesses to upgrade either system independently of the other one.

Examples of microcomputer DBMS packages include Access, Alpha 5, dQuery, Filemaker Pro, and Lotus Approach. Examples of DBMSs that run on client/server systems or mainframes include ADABAS, Microsoft SQL Server, DB2, Oracle, MySQL, Sybase, Ingrus, and Supra. Most microcomputer DBMSs are single-user systems, whereas others (for larger applications) are multiuser systems. Most of these larger systems are limited in how many concurrent users they support, the maximum number of transactions per day they can process, and so forth. Also, not every accounting package can interface with every database, so managers should make sure that any new accounting software they acquire can also read their existing databases, and vice versa.

Case-in-Point 5.1 IDC Company, a research company specializing in IT, estimates that the annual market for relational database management systems in the U.S. is $14.6 billion—and growing by nearly 10% per year. Reasons for such demand and growth include expanding international economies, new data warehousing applications, and increasing demand for such software by U.S. companies.[1]

[1]Source: Penny Crosman, "Database Bubble?" *Intelligent Enterprise* Vol. 9, No. 7 (July 2006), p. 15.

Data Validation

The data definition language (DDL) of a DBMS enables its users to define the record structure of any particular database table (i.e., the individual fields that each record will contain). For example, to create the record structure for the car registration file shown in the top portion of Figure 5-4, you might define the following data fields and characteristics:

Data Field	Date Type	Size	Required?
Social Security number	text	9	yes
Last name	text	50	yes
First name	text	30	yes
Home phone number	text	14	no
License plate state	text	2	yes
License plate number	text	10	yes

Chapter 4 explains how to define this record structure using Microsoft Access. Here, we will focus on the data-validation capabilities DBMSs.

Case-in-Point 5.2 Geographic information systems (GISs) rely heavily on databases to store information about streams, roads, utility installations, and similar geographic information. In regional planning environments—for example, water-resource planning—local districts often need to share data from different databases. One of the biggest problems: disparate proprietary record structures, which sometimes cost local governments hundreds of thousands of dollars to convert to consistent formats. Members of the Northern Kentucky GIS standards task force spent more than 600 hours grappling with this problem and developing new formatting standards.[2]

Mistakes in the important data fields of AIS databases are costly to a company in terms of the time and trouble required to correct them, as well as the potential inconvenience and confusion caused by such errors *until* they are corrected. Simple examples include typing ''4)'' instead of ''40'' for hours worked, ''NU'' instead of ''NY'' for the state code in a mailing address, or ''UPC'' instead of ''UPS'' for the shipper code. Although it is impossible to guard against every possible type of error, database designers can use the following tools of a typical DBMS to catch many of them.

Proper Data Types for Fields. Using Microsoft Access, one input control is inherent in the data type that you assign to a particular data field. For example, if you create a data field as a ''number'' data type, Access will reject all character inputs that are not numbers. Similarly, if you declare a data field as a ''date'' data type, Access will reject all input values (including alphabetic letters or punctuation marks) that cannot be part of a date. This is why it is often better to use data types *other than text* for data fields.

Input Masks. **Input masks** limit users to particular types of data in specific formats—for example, ''123-45-6789'' for a Social Security number,''(123) 456-7890'' for a telephone number, or ''8/9/10'' for a date. Although system designers use special symbols for the mask, the DBMS interprets these symbols as input requirements and acts accordingly. At data-entry time, the user will see just the formatted part of the mask—for example, ''___-__-___'' (see the ''Input Mask'' row in Figure 4-14 of the previous chapter). Input masks

[2]Source: Lisa Martin, et al., ''Sharing Data'' *Geospatial Solutions* Vol. 15, No. 3 (February 2005), pp. 26–31.

help users input data correctly in databases by indicating a general input format, thereby reducing data-entry errors. Such masks also enable the system to reject incompatible data—for example, a letter character mistakenly input in a numeric field.

Default Values. A third input control is to specify a **default value** for the data fields of new records. Examples include the number ''40'' for an hours-worked data field or a ''coach'' seating code of ''Y'' for an airline passenger. Again, such default values help guard against input errors as well as speed data entry.

Drop-Down Lists. You probably have already seen combo boxes on web pages that contain drop-down lists of choices. Databases like Access enable you to use similar boxes for your tables or forms (Figure 5-5). Although such boxes are convenient alternatives to typing data manually, they also control data-entry because they limit user entries to valid inputs. In Access, another advantage of using a combo box is that you can store the choices for it as the values of a separate table, adding to the flexibility of the database itself.

Validation Rules. One of the most versatile data entry controls is the ability to create custom validation tests using a **validation rule**. Using Microsoft Access, for example, you create such rules as a record-structure property of a data field. Figure 5-6 illustrates an example for the ''Fine Amount'' data field of the Parking Violations Code table in Figure 5-4. This (numeric) data field shows the amount of money that a person must pay for a particular parking violation. In Figure 5-5, the expression *Between 1 And 100* that appears in the properties window on the left side specifies the acceptable range of values. The error message in the message box on the right displays the ''Validation Text'' that you specify in this field's properties window. This is what will appear in a message box when a user attempts to enter a value (such as ''200'') that falls outside the allowable range.

Validation rules can be simple, such as the one in Figure 5-6, or much more complex. For example, Access also enables you to use mathematical computations, predefined functions, and logical operators to create more complex validation rules. An example is *Between 1 And 100 AND Not 77*, which means that the entry value must fall in the specified range and cannot be ''77.'' Another example is *Between [fldStartDate] and [fldEndDate]*, which means that the date entered must be between an employee's hire date and his or her termination date.

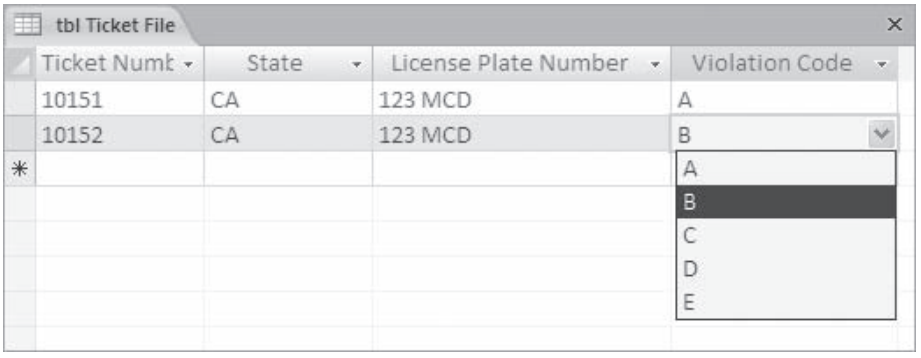


FIGURE 5-5 An example of a combo box at run time for the violation code of the Ticket file table.

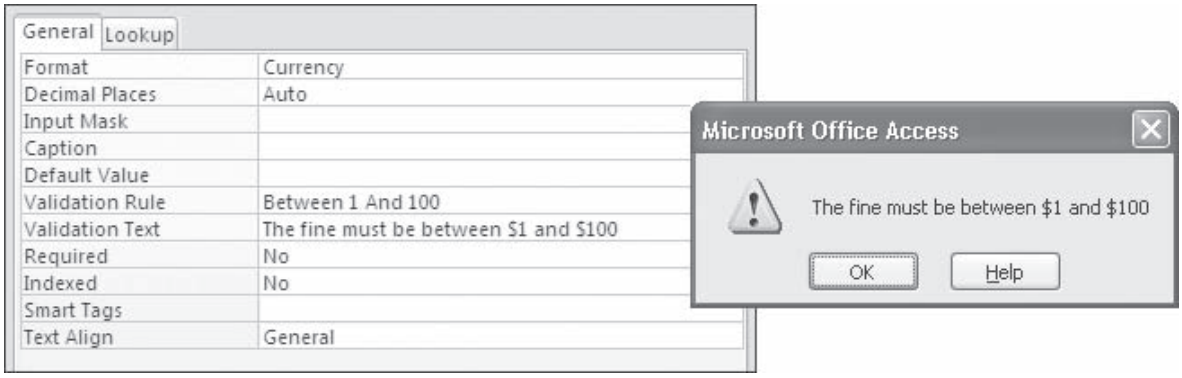


FIGURE 5-6 Left: The properties window for the Fine Amount data field of the Parking Violations table. Right: The error message that a user would see if he or she attempts to enter a value for this field that falls outside the specified range.

Case-in-Point 5.3 The consulting team had a massive task: consolidating the payroll records of 110,000 employees stored in five different systems into the database of a grocery chain's ERP system. Because the project was over budget, company executives instructed the team to ignore validating taxes for terminated employees. But some of those employees didn't *stay* terminated, and the validation costs that the team would have incurred for such validation was "chump change" compared to the cost of correcting the erroneous tax records for the rehires.[3]

Referential Integrity. A final data-entry control is to enforce **referential integrity** in relational database tables. This feature controls certain inconsistencies among the records in relational tables. Consider, for example, the possibility of *deleting* a parking violations code record from the third database table in Figure 5-4 (e.g., deleting the record for Code A—meter expired). We can't allow such a deletion because this would disrupt all the *references* to that record in the Tickets table. For the same reason, we can't allow new records in the ticket table to reference nonexistent codes in the parking violations code file—for example, a ticket with code "Z" (if such a code didn't exist). This would be a parking ticket for a nonexistent violation.

Database management systems make it easy to enforce referential integrity. In Access, for example, you simply check a box in the Edit Relationships dialog window at the time that you create the relationship—see Figure 5-7. This enforcement performs two vital functions. First, it does not allow record deletions in the "one" table of a one-to-many relationship. Second, it does not allow a user to create a new record in the "many" table of a one-to-many relationship that references a nonexistent record on the "one" side. Case 5–25 illustrates these concepts in more detail.

In Figure 5-7, note that the Edit Relationships window in Access provides two additional boxes that you might check: one that allows "cascade update related fields" and one that allows "cascade delete related records." These options enable you to override the referential integrity rules just described for parent records (although Access will warn you first). If you chose the first of these options, for example, you could delete a record in the

[3]Source: No author, "Anonymous Tales from the Front Lines: Do It Fast or Do It Right," *Infoworld* Vol. 28, No. 9 (February 26, 2006), p. 46.

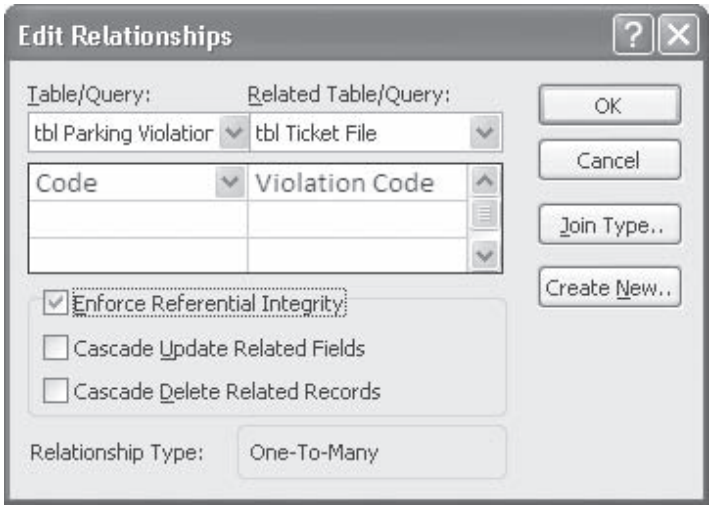


FIGURE 5-7 This dialog box appears when you first create a relationship in Microsoft Access. Clicking the check box to Enforce Referential Integrity does just that.

Parking Violations Code File, and Access would also delete the reference to that record in the records of the Ticket File. (This would not be desirable here, however—it would leave you with tickets in the Tickets File with no violation code in them.) The second option allows you to delete a parent record, even if there are matching child records for it. For example, if you delete a record in the Car Registration table, Access will then delete all the ticket records associated with that record (car) in the Ticket File table.

EXTRACTING DATA FROM DATABASES: DATA MANIPULATION LANGUAGES (DMLs)

The totality of the information in a database and the relationships of its tables is called the database **schema**. Thus, the schema is a map or plan of the entire database. Using the previous student-parking example, the schema would be all the information that a university might store about car registrations and parking tickets.

Any particular user or application program will normally be interested in (or might be limited to) only a subset of the information in the database. This limited access is a **subschema**, or *view*, in database parlance. For example, one subschema for our parking database might be the information required by the university registrar—e.g., the student's name, Social Security number, and outstanding parking tickets. (Many universities do not allow students to graduate with outstanding parking tickets.) Subschemas are important design elements of a database because they dictate what data each user needs, and also because they protect sensitive data from unauthorized access. This is one reason why a university might design several subschemas for its parking database that purposely exclude student Social Security numbers.

The terms *schema* and *subschema* describe a simple idea—the distinction between the design of a database on one hand and the uses of a database on the other. The goal is to design a database schema that is flexible enough to satisfy the subschema uses required of it. This design can make the difference between an AIS that barely works and an AIS that

provides a very real competitive edge to a profit-seeking business. Here are some ways of creating subschemas.

Creating Select Queries

The purchasing agent of a manufacturing company needs to know what inventory parts balances are now below their reorder points. A payroll manager wants to know which employees are eligible to receive year-end bonuses. A tax assessor is interested in those areas of the city that have experienced the most real estate appreciation.

What these applications have in common is the need for selective information from one or more database tables. **Queries** allow database developers to create customized subschemas. For example, using the student car registration database, you might want to (1) look up something about a specific student (e.g., the license plate number of his or her car), (2) change the information in a specific record (e.g., update a student's phone number), (3) delete a record (e.g., because a student sells his or her car), or (4) list file information selectively (e.g., prepare a list of all students with California license plates). A **dynaset** is a dynamic subset of a database that you create with such queries, and the purpose of a **data manipulation language** (DML) is to help you create such dynasets.

One-Table Select Queries.

A **select query** creates a dynaset of database information based on two types of user-specified criteria: (1) criteria that determine which records to include, and (2) criteria that determine which data fields to include *from* those records. Figure 5-8 illustrates a simple select query that displays particular information from a single table using Microsoft Access. This example asks the system to display the last name, first name, phone number, license plate state, and license plate number for all cars with California license plates.

You can create several types of queries with Access 2007. One is a simple *filter query* that references only one table. Another combines the information from several tables. A third type is an action query. We look at each of these queries in order.

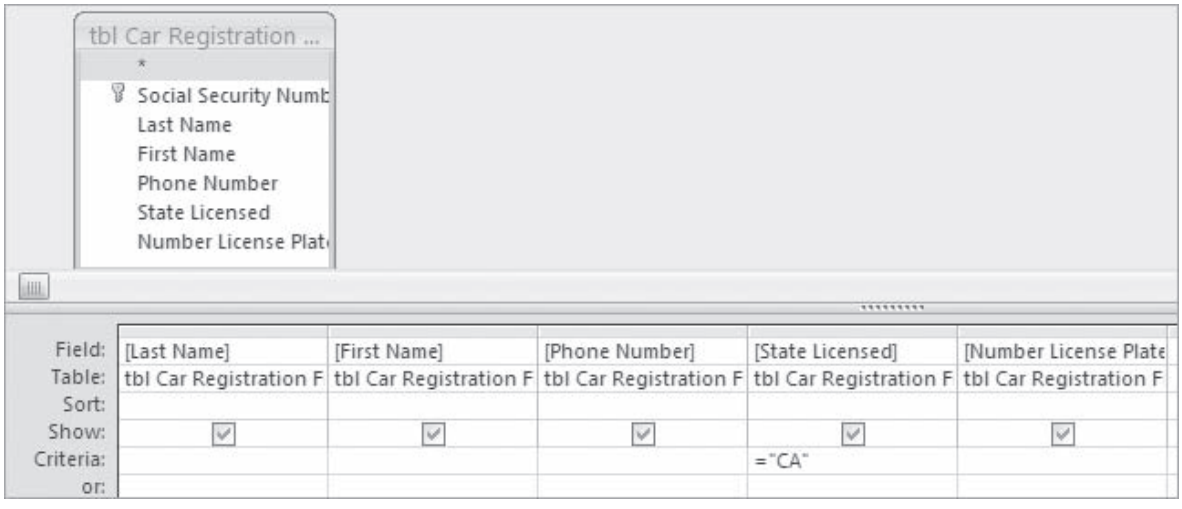


FIGURE 5-8 A simple query to select all car registrations with CA (California) license plates.

Single Criterion. To create a simple filter query, first click the Create option on the main menu bar. In the Create menu, click ''Query Design.'' Access will display a small dialog box that allows you to select the table(s) on which to base your query. To create the query in Figure 5-8, we only need the tbl Car Registrations File. The bottom portion of Figure 5-8 shows the layout in which to enter your data fields and the selection criteria for them.

Your next task is to select the data fields in each record you wish to display. One way to do this is to click on the first (left-most) cell in the Field row in the lower portion of the Query panel. An arrow will appear in this cell. Click on this arrow and a drop-down list of available data fields will appear. Select the field from this list that you wish to display in the current column (we selected Last Name in the figure). Continue across the panel until you have selected all the fields you need. Alternate methods of selecting data fields for queries in Access 2007 are (1) double clicking on the desired field name in the table list of the upper panel or (2) dragging the field name in the table list to the column.

Next, you must specify the selection criteria for the query. For example, to display only those records with California licenses enter ''=CA'' in the criteria box under ''State Licensed.'' You will see CA is now enclosed with quotation marks, which Access automatically adds for you. To specify criteria in general, all basic comparison operators are available—i.e., =(equals), <(less than), >(greater than), >=(greater than or equal to), <=(less than or equal to), and <>(not equal to).

You are now ready to run the filter. To do this, click the exclamation point with the word ''Run'' on the left portion of the main menu. The results of your query will appear as shown in Figure 5-9. You can toggle back and forth between design and run modes by clicking on the View option in the Results section of the left side of the main menu.

After you have created a query, most DBMSs enable you to save it in a separate file for later use, thus eliminating the need to rewrite it. This saves developer time as well as spares end users the work of creating such queries in the first place. The letters ''qry'' are the standard naming prefix for queries. Thus, as you can see on the tab in Figure 5-9, we named our query ''qryCalifornia License Plates.''

Multiple Criteria. It is also possible to specify multiple criteria in a query. For example, suppose you wanted a list of all car registrants whose cars had California license plates *and* whose last names were ''Curry.'' To create such a query in Access, simply type the name ''Curry'' in the ''Last Name'' column and in the *same* Criteria row as the ''CA.'' Access interprets criteria appearing in the same row as an ''and'' operation. The results will be all those records with last name ''Curry'' *and* whose license plate state is ''CA.'' Similarly, if you specify three criteria in the same row, then Access will find database records in the table satisfying all three requirements.

Home Create External Data Database Tools

Relationships | qryCalifornia License Plates

Last Name	First Name	Phone Numbe	State Licensed	Number License Plate
Curry	Dorothy	(916) 358-4448	CA	123MCD
Jones	Roberta	(987) 654-2132	CA	876JJH
Kerr	Stephen	(916) 764-3211	CA	498SEK
Mason	Richard	(916) 563-7865	CA	253DAL
Tajiri	Colleen	(916) 543-2211	CA	897ABC
*				

FIGURE 5-9 The result of the query in Figure 5-8.

Sometimes, you might want to search for records that satisfy alternate requirements—for example, car registrants whose cars have California license plates *or* whose last names are ''Curry.'' To create such a query in Access, use multiple lines at the bottom of the Query dialog box in Figure 5-8. The result of this query will be all records that satisfy *either* requirement. (The system will also include records satisfying both requirements.)

Multi-Table Select Queries. Many accounting applications require information that must be drawn from more than one database table. For example, suppose you wanted to create a report similar to the following:

Ticket Number	License Plate	Registered Car Owner	Listed Phone Number	Amount of Ticket
10151	CA 123 MCD	Dorothy Curry	(916) 358-4448	$10.00
10152	CA 123 MCD	Dorothy Curry	(916) 358-4448	$20.00
10231	CA 253 DAL	Richard Mason	(916) 563-7865	$50.00
etc.				

Notice that the information in this report comes from three different tables: the ticket number and license plate number come from the Tickets table, the registered car owner's name and phone number come from the Car Registrations table, and the amount of each ticket comes from the Parking Violations Code table. To create such a report, you must first join the tables using the Relationships window. Chapter 4 explains how to perform this task.

Your next step is to construct the query. Follow the steps outlined above for creating simple queries, being careful to select the data fields shown in Figure 5-10. The results should be similar to those shown in Figure 5-11.

The tasks performed by the query shown in Figure 5-10 are nontrivial. To appreciate this, imagine that you had to create the report above manually, using the information shown in Figure 5-4. If there were hundreds of parking tickets in a given week and thousands of car registration records, the work required for this job would be enormous. But a computerized DBMS using a DML can do this quickly and automatically in just a few seconds—an amazing feat if you think about it!

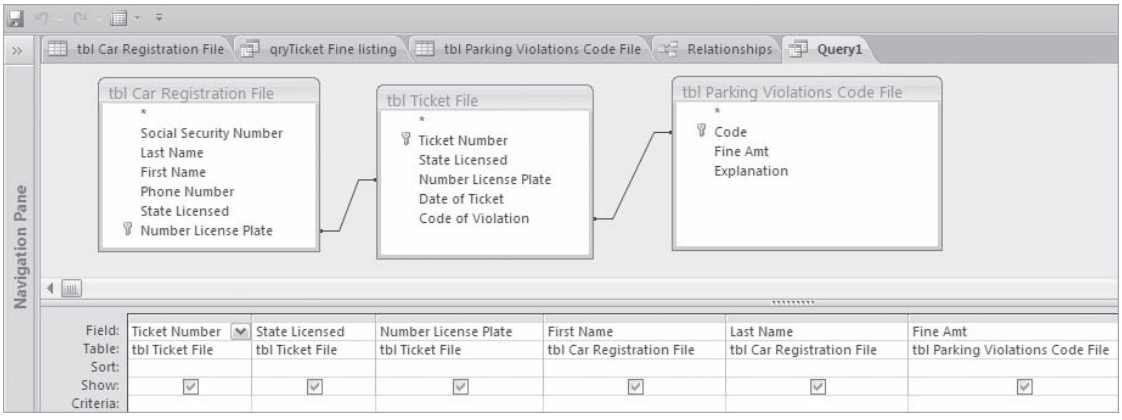
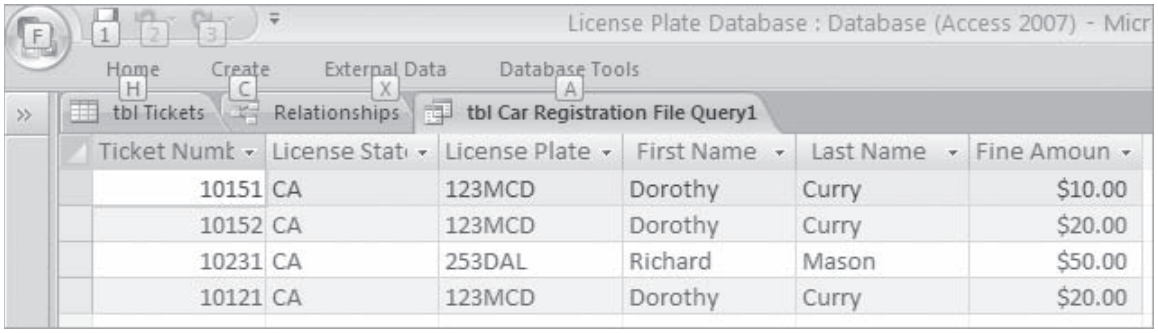


FIGURE 5-10 A multiple-table query.



Ticket Numb	License Stat	License Plate	First Name	Last Name	Fine Amoun
10151	CA	123MCD	Dorothy	Curry	$10.00
10152	CA	123MCD	Dorothy	Curry	$20.00
10231	CA	253DAL	Richard	Mason	$50.00
10121	CA	123MCD	Dorothy	Curry	$20.00

FIGURE 5-11 The results of the multiple-table query in Figure 5-10.

Creating Action Queries

Although most queries simply extract information from database tables, some accounting tasks require users to update, match, or delete multiple records in a single operation. Microsoft Access supports the **action queries** listed below. You can create any of these queries by selecting the appropriate choice from the New Query dialog box shown in Figure 5-12. (To launch this dialog box, select ''Query Wizard'' from the Query dialog box in the Access database main menu.)

1. ***Simple query Wizard*** does the same thing as described previously under ''One-table Select Queries.''
2. ***Crosstab queries*** enable you to perform a statistical analysis of the data in a table and provide the cross-tabulation results in a row-and-column format similar to a pivot table in a spreadsheet. For example, a crosstab query might show the average invoice amount for each vendor in a vendor table, or the average credit purchase for each customer living in a specified zip code.

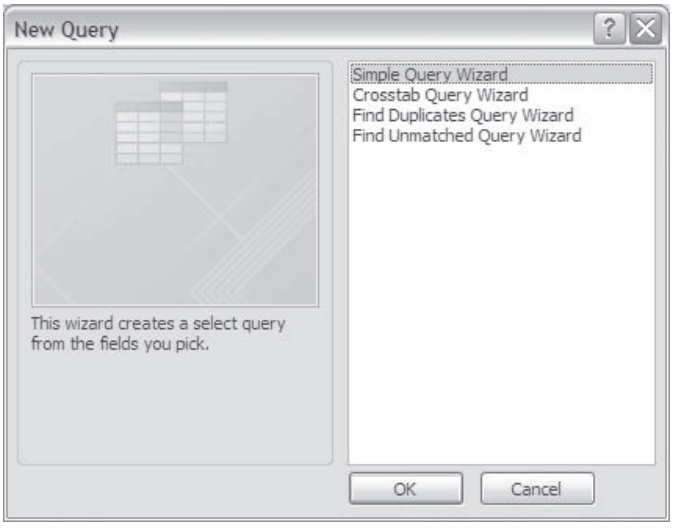


FIGURE 5-12 The Query Wizard screen of options available in Access.

3. ***Find-Duplicates queries*** enable you to find those records with duplicate entries in a specified field. Many common auditing tasks require such queries—for example, finding duplicate customer orders, finding employees with the same employee or Social Security number, or searching for different vendor records with the same address. Note that a simple select query might enable you to find *one example* of such duplicates. A find-duplicates query enables you to find *all duplicates* with a single query.
4. ***Find-unmatched queries*** enable you to find the records in one table with no matching record in another table. For example, such queries enable auditors to identify those payroll records with no matching employee records or identify those vendor invoices with no matching supplier record.
5. ***Delete queries*** enable you to delete table records selectively. Applications include the ability to delete the records of (1) employees who quit the organization, (2) students who drop out of school or graduate from school, or (3) inventory products no longer sold by the company.
6. ***Append queries*** enable you to append records from one table to the end of another table. Accounting examples include the ability to add the payroll records for the current period to a year-to-date table or to consolidate the employees from two departments into a single table.
7. ***Sum a column*** by clicking the Sum figure. The word ''Total'' will be added in the first column, and click under the ''Fine'' amounts and the column will be added together.
8. ***Update queries*** enable you to alter selected table records systematically. Accounting applications include the ability to raise all suggested retail prices of a particular product line by 10%, lower the salaries of all those employees with a low performance rating by 5%, or delete shipping charges for all customer purchases over a set limit.
9. ***Make-table queries*** enable you to create a new table from the records that you select in an existing table. For example, a university might want to create a separate table of all graduating seniors. A common accounting application is to create a separate table of all the records you are about to delete using a delete query.

Guidelines for Creating Queries

The preceding discussions explained various kinds of select queries and action queries. Here are some guidelines to help you create error-free queries using Microsoft Access:

1. ***Spell accurately and be sensitive to capitalization.*** The criteria for Access select queries are case sensitive. For example, you will not get matches if you specify California licenses as ''Cal'' or ''Ca'' in a criteria line if the entries in the underlying database table are ''CA.''
2. ***Specify AND and OR operations correctly.*** If you want a query to satisfy two conditions simultaneously (i.e., perform an AND operation), enter the criteria on the *same line* of your query. If you want a query to satisfy *either* of two conditions (i.e., perform an OR operation), place them on successive criteria lines.
3. ***Tables must be joined properly.*** If you wish to construct a multi-table query, the tables must first be joined properly in Access' Relationships window.
4. ***Name queries systematically.*** Query names should begin with the standard ''qry'' prefix. It also helps to assign mnemonic query names—for example, ''qryCustomers_in_California,'' ''qryGraduating_Seniors'' and so forth.

5. ***Choose data fields selectively.*** Double-clicking on the asterisk (*) in the data field list of a table (e.g., the first symbol in each of the three table lists in Figure 5-10) enables you to include *all* the data fields from that table in your query. Because most commercial database tables have many data fields, using this option can result in a large number of data fields (i.e., columns in the lower portion) of your query. This makes query design unwieldy.

Structured Query Language (SQL) and HyperText

In addition to using a DML in a DBMS, you can also access selected information from a database using a *data query language*. The American National Standards Institute (ANSI) has adopted standards for one such query language: **structured query language (SQL)**. This language is important because many relational databases such as *Access* support it. Figure 5-13 shows how you might construct the request for records with California license plates using SQL.

SQL is a useful tool for auditors. In Microsoft Access, the user points to a database table to include in a query. With SQL, the user specifies a table and fields, using commands such as FROM, SELECT, and WHERE. FROM identifies the table source, and SELECT chooses the data fields to include in the query. The WHERE command can specify criteria, such as "State = CA." An auditor could select files for review using these commands. For example, the WHERE command could refer to sales orders in excess of a specified dollar amount.

Yet another way of finding such information is with **hypertext**. DBMSs use hypertext by highlighting key words or display text in different-colored characters. Clicking on a keyword with your mouse directs the DBMS to move directly to that entry. One hypertext example is Apple's Hypercard for Macintosh microcomputers. Another example is **Hypertext Markup Language (HTML)**, the language for creating web pages. Hypertext systems are especially useful for researching technical materials in which you find it convenient to jump from subject to subject or web page to web page.

Case-in-Point 5.4 Wikipedia.org hosts one of the largest encyclopedias or compendiums of online information, containing (for example) almost 3 million articles in English. Most articles explain a term or concept such as "accounting information systems" in simple language, and include hyperlinks (in blue) to explanations of the technical terms contained in each article.[4]

It is also possible to store hypertext entries directly *in* databases. For example, you might want to store a "live" web address or email address *within* the records of a table.

```
SELECT (LastName, FirstName, PhoneNumber, LicPlateState, LicPlateNo)
FROM CarRegistrationFile
WHERE LicPlateState = CA;
```

FIGURE 5-13 An example of SQL instructions for the example of Figure 5-7. These instructions will list the last name, first name, phone number, license plate state, and license plate number of all cars with license plate state code "CA."

[4]Source: from the authors.

Conversely, you might want to store hypertext links within the body of database forms or other screen objects. Modern databases such as Access provide tools for performing such tasks. An additional version of hypertext is **eXtensible Markup Language (XML)**, which allows end users to create their own tagging standards as well. We discuss XML in greater detail in Chapter 12.

Sorting, Indexing, and Database Programming

In addition to accessing or listing records selectively, a DBMS also enables you to reorganize an entire table. One way to do this is by sorting records, which means physically rewriting records on a disk in the desired order. This is both time consuming and usually unnecessary. It is faster and easier to index your records (refer back to the last row of Figure 5-5), which merely creates a table of record keys and disk addresses that accomplishes the same purpose as sorting. Thus, when users specify ''sort'' in queries, Access does not physically reorder records but instead merely temporarily assembles the information for display purposes.

Finally, even the best DBMS software cannot anticipate every user's processing needs. For this reason, advanced DBMSs include programming tools that enable users to develop their own processing applications. One common requirement is for customized data-entry screens, which enable users to include better data descriptions and more detailed instructions on input screens. Similarly, programming languages (such as VBA for Microsoft Access) enable users to create custom processing routines—for example, to create their own data-validation routines. This end-user programming is important because it enables users to perform their own data processing without the technical assistance of IT professionals.

Online Analytical Processing (OLAP) and Data Mining

Although SQL enables users to extract data from one or more database tables, **online analytical processing (OLAP)** allows users to extract complex information that not only describes ''what'' happened, but helps explain *why* it happened. Several software developers now market OLAP packages. Examples include *Integration Server* (Arbor Software), *Holos* (Seagate Technology), *PowerDimensions* (SyBase), *Plato* (Microsoft), and *WhiteLight* (WhiteLight Systems). Some of these tools only work with specific databases, while others interface with several of them. Most allow end users to perform their own database analyses, including data mining (discussed shortly).

OLAP Features. An important feature of OLAP is the ability to conduct multidimensional analyses (Figure 5-14). One dimension may be ''time.'' Other dimensions might be ''customer,'' ''product,'' or ''geography.'' For example, OLAP can help you examine sales over time for a particular product in a specific geographical region. Another feature of OLAP is a ''drill-down'' capability. This allows you to examine data at increasing levels of detail. As an example, you can take sales for one quarter shown by geographical region and drill down to see sales for each state, and then for each major city within that state. Similarly, you can drill down sales from a product line to a specific product, and then to a specific product size or color. This type of analysis can provide the ''why'' behind what has happened.

OLAP has a variety of other helpful features. One is the ability to create **pivot tables**, which are two-dimensional statistical summaries of database information (and similar to the pivot tables of Microsoft Access or Excel). The example in Figure 5-14 is a two-dimensional

Sales Report—Best Multimedia
2nd Quarter 20xx

	Northeast	Northwest	Southeast	Southwest	Midwest	TOTAL
Total Sales						
CDs	**$50,000**	**$45,000**	**$37,000**	**$34,100**	**$34,000**	**$200,100**
Pop Rock	$30,500	$20,000	$22,000	$17,000	$19,000	$108,500
Jazz	$4,200	$7,500	$5,000	$4,100	$2,200	$23,000
Show Tunes	$8,200	$10,000	$4,800	$6,000	$4,700	$33,700
Rap	$7,100	$7,500	$5,200	$7,000	$8,100	$34,900
DVDs	**$80,800**	**$92,000**	**$78,000**	**$56,000**	**$60,200**	**$367,000**
Action	$12,000	$13,000	$11,000	$9,700	$9,000	$54,700
Classics	$14,000	$18,000	$25,000	$ 11,000	$7,000	$75,000
Comedy	$16,000	$17,000	$16,200	$13,400	$17,600	$80,200
Drama	$15,900	$17,100	$16,100	$13,400	$17,600	$80,100
Horror	$14,900	$17,900	$1,700	$1,500	$1,000	$37,000
Mystery	**$8,000**	**$9,000**	**$8,000**	**$7,000**	**$8,000**	**$40,000**
Suspense	$2,000	$3,800	$1,800	$3,000	$2,000	$12,600
Thrillers	$4,000	$4,000	$5,000	$3,000	$5,000	$21,000
True Crime	$2,000	$1,200	$1,200	$1,000	$1,000	$6,400
Software	**$22,000**	**$20,800**	**$19,700**	**$20,000**	**$25,000**	**$107,500**
Total	$152,800	$157,800	$134,700	$110,100	$119,200	$674,600

FIGURE 5-14 A pivot table showing a drilldown of sales totals by product type and region.

analysis of sales by product (on the vertical axis) and region (along the horizontal axis). Pivot tables enable users to choose what type of summary information to display (e.g., total sales, average sales, or maximum sales), as well as to change an overall selection category (e.g., change the period in which to view sales data). Further information about OLAP features, products, and reviews about OLAP software can be found at www.altaplana.com/olap and www.olapreport.com.

Data Mining. Closely connected to OLAP is the concept of **data mining**, which means using a set of data analysis and statistical tools to detect relationships, patterns, or trends among stored data. For example, data mining tools might enable an auditor to reveal suspicious payments by a governmental agency. Data mining also helps advertisers in cross-selling products or offering tie-in promotions, retailers decide product placements in their stores (e.g., placing snacks near the frozen pizza section), and sales managers increase customer satisfaction. Because data-mining tools can sift through massive amounts of corporate data to detect patterns, they can be particularly effective tools for firms seeking to better understand their customers and what they want or need.

Case-in-Point 5.5 Talbots is a leading retailer of women's wear whose legacy batch reporting system was unable to provide managers with timely information. To address this problem, the company recently installed SQL Server 2005—a data warehouse system from Microsoft—and additional software that transfers POS data such as purchases directly into the warehousing system in real time. The new software provides managers with up-to-the-minute

information on sales and inventory, thereby enabling them to make better staffing decisions as well as respond more quickly to changing market conditions.[5]

A wide variety of software tools now provide data mining capabilities. One possibility is to use the data-mining tools that already exist in OLAP software, database software, or artificial intelligence algorithms. Alternately, users can purchase specific software packages for data-mining tasks—for example, Darwin (Oracle), Intelligent Miner (IBM), Enterprise Miner (SAS), or Clementine (SPSS).

Although the most popular uses of data mining are related to sales and marketing, there are many accounting applications as well. One possibility is for auditors to use data mining to detect credit-card anomalies or suspicious behavior. For example, fraudulent credit card transactions may follow a pattern, such as an increase in the total amount of purchases immediately following a credit card theft or products with special characteristics (such as ones that can easily be sold). Another application is for investors to use data-mining tools to predict corporate bankruptcies. A third application is for government workers to use such tools to identify fraudulent claims for worker's compensation or excessive uses of welfare services.

OBJECT-ORIENTED DATABASES, MULTIMEDIA DATABASES, AND DATA WAREHOUSES

The databases that we have discussed so far are traditional ones that mostly handle text data (i.e., data that can be neatly organized and categorized according to the values stored in text or numeric data fields). Not all databases are this simple.

Object-Oriented and Multimedia Databases

An **object-oriented database (OODB)** is a database that contains both the text data of traditional databases and information about the set of actions that can be taken on these data fields. For example, a payroll file might contain not only traditional information about an employee, but also instructions that indicate how to compute an employee's net pay.

Case-in-Point 5.6 Geotagging means using an object-oriented database to store geographic information about entities of interest using a digital geographic map. In a typical application, the system allows users to indicate where they live or work, thereby helping individuals arrange for car pools. Flickr (flickr.com/map) allows camera users to upload pictures to indicate where the pictures were taken. Marketers suggest that such systems will soon allow advertisers to better target their customers.[6]

Many OODBs are **multimedia databases** that include graphics, audio information, and animation. These databases also typically store information about how to display graphics, how to play audio clips, and so forth. Multimedia databases are used by real estate brokers to store pictures and perhaps narrated tours of listed properties, by companies to train

[5]Source: (No author), ''Talbots to Boost Efficiency with Microsoft-based Retail Data Warehouse'' *Apparel Magazine* Vol. 47, No. 6 (February 2006), p. 18.

[6]Source: Rubel, Steve ''Location, Location, Location'' *Advertising Age* Vol. 77, No. 39 (September 25, 2006), p. 29.

FIGURE 5-15 The employee records of this security database contain both text data and the picture of each employee.

employees, by police departments to store ''mug shots'' and voice prints of prisoners, and by publishing houses to enhance the descriptions of everything from cookbooks to encyclopedias. Your employer might even use such a database to store your picture in an employee file (Figure 5-15).

Specialized accounting applications of multimedia databases include those that store the audio portions of audit interviews, pictures of important assets, or images of critical financial contracts. These ''unstructured objects'' require a new definition of what we mean by ''data'' and how we organize them. But OODB records can still be manipulated. For example, a speech still has such characteristics as ''speaker,'' ''subject,'' and ''length,'' and these characteristics can be used to search a database table and retrieve the desired object, whatever that might be.

Like multimedia databases, **multidimensional databases** store large quantities of data. The ultimate goal of such databases is to enable employees at various levels of an organization to define their own tables and reports in formats that are most useful to them. Some multidimensional databases accomplish this goal by combining data from several independent data sources. Others do so with unique data schemas, while still others do so by enabling users to scale existing data fields or otherwise define their own data fields. Because it is not clear how best to deliver quality data to organizational users, the subject of multidimensional database development is an active area of research.

Data Warehouses

Where feasible, it often makes sense to pool the data from separate applications into a large, common body of information called a **data warehouse**. The data in a data warehouse are rarely current. Rather, they are typically ''older information'' that were initially collected for other reasons during the conduct of normal operations and daily activities of an organization. For example, a sales transaction creates data that help management make decisions about production, cash availability, and so on. The sales transaction data also impact financial statements. Managers are learning, though, that much of the data gathered about sales and other operational activities can also be useful strategically. For example, in recording a sale, an AIS collects data about the customer, the product, the timing of the sale, and so on. This information can be helpful in predicting future sales of specific

products or by a certain category of customers. To obtain these benefits from the data collected, the data must be amassed in a central location—the data warehouse.

To be useful, the data in data warehouses should have the following characteristics: (1) free of errors, (2) defined uniformly, (3) span a longer time horizon than the company's transaction systems, and (4) optimized data relationships that allow users to answer complex questions—for example, queries requiring information from several diverse sources.

One advantage of a data warehouse is to make organizational information available on a corporate-wide basis. For example, with such an approach, the marketing representatives of a company might then gain access to the company's production data and thereby be better able to inform customers about the future availability of desired, but as yet unmanufactured, products. This idea is also central to the concept of an **enterprise-wide database** (i.e., a large repository of organizational data that comes from, and is available to, a wide range of employees). Another advantage is to facilitate data mining.

Case-in-Point 5.7 With more than 9 million customers, KeyBank (www.key.com) is the thirteenth largest bank in the United States. To help market financial products, the bank created a million-dollar DB2 data warehouse that allows its managers to determine what investments its customers prefer (e.g., CDs or mutual funds), and how best to sell products (e.g., direct mail or Internet). Bank officials credit the data warehouse, the decision tools that mine it, and the ability of different departments to share data for increasing customer contacts by 200%, and a 100% return on the investment in 14 months.

Building a data warehouse is a difficult job. The developers must first decide what data to collect, how to standardize and **scrub** (clean) the data to ensure uniform accuracy and consistency, and how to deal with computer records that typically begin in a non-normalized form. One reason for these difficulties is that the data in data warehouses may come from several sources—for example, an AIS in one case and a production application in another case. As a result, the same data element could have two different representations or values—for example, an eight-digit numeric product code in the AIS and a six-digit alphabetic character code in the production application. Similarly, one corporate division might capture sales daily while another collects the same data weekly. The developers must determine data standards in both cases, reconcile any discrepancies, and account for missing fields and misspellings. Another challenge is to build the data warehouse in such a way that users can access it easily and find answers to complex questions.

If data warehouses are so costly, difficult, and time consuming to develop, why do companies bother with them? The answer is that they generate many benefits in return, including increased employee access to valuable information, the ability to answer complex questions, and a potential return on investment that can exceed 400%.

Case-in-Point 5.8 Provident Central Credit Union is one of the largest credit unions in the United States. Recently, it created a data warehouse to make use of the rich transaction and customer data it gathers. The company plans to use the data in the warehouse to conduct one-to-one marketing campaigns for custom products and to improve its pricing and responsiveness to its membership of almost 96,000 customers. The data warehouse holds the answers to many complex questions such as, "Who are the most profitable customers?" and "How can we improve our customer relationships through product and service offerings?"[7]

Where corporate executives believe the rewards for building a data warehouse are not high, they can opt instead to build a **data mart**. Data marts are smaller than data

[7]Source: www.taborcommunications.com/dstar/01/0508/103021.html.

warehouses in storage size and typically focus on just one application area—for example, marketing data. However, in most other ways, they are similar to data warehouses.

AIS AT WORK
Run Your Data Warehouse Like a Fine Restaurant

What do a data warehouse and a fine restaurant have in common? According to Margy Ross and Ralph Kimball of the Kimball Group—a consulting company specializing in data warehousing—quite a bit. Here are several ideas:

In the restaurant:	Restaurant staffers create meal plans and then develop customer menus based on these offerings.
In the data warehouse:	Data warehouse developers provide menus of what data are available, typically through online display screens.
In the restaurant:	The physical layout is highly efficient, with concern for high levels of throughput.
In the data warehouse:	Thus, the same concern for efficiency and high levels of throughput applies. Patrons don't want to wait for data any more than restaurant patrons want to wait for their meals. This is why the data warehouse was created in the first place—to streamline the delivery of information to end-users.
In the restaurant:	Good meals depend upon quality raw materials.
In the data warehouse:	Good outputs also depend upon quality inputs. Thus, the system validates incoming data for such factors as (1) reasonableness, (2) integrity, and (3) value to end-users.
In the restaurant:	Professional employees staff the restaurant and use professional tools in their jobs.
In the data warehouse:	Professionals create, monitor, and develop the policies that govern operations—not patrons. This is no place for amateurs.
In the restaurant:	''Hard work'' and ''high quality'' are the watchwords of the operation, because what gets delivered to the customer can make or break the restaurant. Delivering consistent, high-quality meals is important because the restaurant's reputation depends on them.
In the data warehouse:	The same requirements of high data quality, integrity, and consistency are the order of the day. Consumers don't want half-baked (incomplete or flawed) information any more than they want half-baked food.
In the restaurant:	The kitchen itself is off limits to patrons. Security is important both for the safety of the customers and the kitchen staff.
In the data warehouse:	The same security concerns apply. Customers stay outside the preparation area, and are confined to the ''eating area.''
In the restaurant:	Managers often check with diners to ensure their satisfaction.
In the data warehouse:	Customer needs are also very important, and monitoring user satisfaction is just as desirable.

Source: Margy Ross and Ralph Kimball. ''Data Warehouse Dining Experience'' *Intelligent Enterprise* Vol. 7, No. 1 (January 1, 2004), pp. 12–14.

SUMMARY

- Databases must be designed carefully. The process of normalization enables designers to minimize data redundancy, insertion and deletion anomalies, and transitive dependencies. The goal is to develop a database that is at least in third normal form.
- Database management systems (DBMSs) enable users to create their own databases using data definition languages (DDLs) and to manipulate file data using data manipulation languages (DMLs).
- Designers use a variety of data-validation techniques to help ensure data accurancy and integrity. Examples include choosing data types carefully for data fields, using input masks, using default values, creating a wide variety of validation rules, and enforcing referential integrity.
- An important use of databases is to extract selected information, and Access provides a number of tools for constructing select queries and action queries. These tools allow users to extract data from a single table or from multiple tables. Following the guidelines in this chapter can help you avoid errors when creating such queries.
- Three additional ways of extracting information from databases are to use structured query language (SQL), online analytical processing (OLAP) tools, or hypertext.
- Users can also manipulate database information by sorting, indexing, using data mining tools, or performing specialized tasks with end-user programming languages.
- Object-oriented databases (OODBs) enable users to store both data and instructions on how the data should be displayed or computed. Multimedia databases are OODBs that enable users to store graphics, pictures, sound clips, and animation clips in addition to text data.
- Data warehouses typically combine the information from separate databases into large sets of cross-functional data repositories that can help businesses increase data-retrieval efficiency, output productivity, and long-term profitability.

KEY TERMS YOU SHOULD KNOW

action query
data definition language (DDL)
data manipulation language (DML)
data mart
data mining
data warehouse
database management system (DBMS)
default value
dynaset
enterprise-wide database
first normal form (1 NF)
flat file
hypertext
hypertext markup language (HTML)
multidimensional database
multimedia database
normalization
object-oriented database (OODB)
online analytical processing (OLAP)
pivot table
query
query wizard
referential integrity
schema
second normal form (2 NF)
select query
structured query language (SQL)
subschema
third normal form (3 NF)
transitive dependencies
validation rule
XML (extensible markup language)

TEST YOURSELF

Q5-1. A database is in third normal form (3 NF) if it is second normal form and:

a. All the data attributes in a record are well defined

b. All the data attributes in a record depend upon the record key

c. The data contain no transitive dependencies

d. The data can be stored in two or more separate tables

Q5-2. The difference between (1) a database management system (DBMS) and (2) a database, is:

a. Nothing—these terms are synonyms.

b. The first is hardware, the second is software

c. The first is program software, the second is proprietary data and related files

d. The first refers to a complete accounting system, the second refers to a subset of that

Q5-3. An example of a *validation rule* is:

a. An input value must be an integer

b. An input value must also have a default value

c. An input value must be between 0 and 40

d. You cannot delete parent records that have child records associated with them

Q5-4. To construct a select query in Microsoft Access in which you want to satisfy two conditions simultaneously—i.e., implement an *and operation*—you should:

a. Specify both criteria in *separate fields* of the same Criteria line of the query

b. Specify both criteria in the *same field* of the Criteria line of the query

c. Specify each criteria in *separate fields and in separate Criteria lines* of the query

d. Give up; this is not possible in Microsoft Access

Q5-5. To adjust the minimum wage of all payroll employees to the current federal level, you should use a(n):

a. Update query

b. Append query

c. Find minimums query

d. Tax expert

Q5-6. To identify all those employees receiving payroll checks but who have no matching record in a payroll master file, you should use a(n):

a. Auditor

b. Find unmatched records query

c. Cross-tabs query

d. Update query

Q5-7. All of the following are examples of DBMSs *except*:

a. Access

b. Oracle

c. DB2

d. SQL

Q5-8. All of the following are examples of action queries *except:*

a. Update query

b. Append query

c. Delete query

d. Find missing data query

Q5-9. The difference between (1) using an update query and (2) updating a single record is:

a. Nothing—these are the same thing

b. The first updates all selected records, the second only affects one record

c. The first updates more than one table, the second updates only one record

d. None of these is correct.

Q5-10. Which of these database tools is an accounting manager most likely to use to perform online, ''drill-down'' analyses?

a. Creating pivot tables

b. OLAP

c. HTML web pages

d. SQL

Q5-11. SQL is an example of:

a. A tool to perform online analytical processing

b. A database management system

c. A query language

d. A multimedia database

DISCUSSION QUESTIONS

5-1. What is the process of normalization? What levels are there, and why do database developers seek to normalize data?

5-2. What are database management systems? Are they the same as databases? Why are DBMSs classified as software and not hardware?

5-3. What are data definition languages (DDLs)? How are they related to DBMSs?

5-4. What is a record structure? When defining a record's structure, what is meant by the term ''data type?'' Give some examples of data types.

5-5. Why do database developers link tables together? How is this done using Access?

5-6. What is data validation? Why is it important? Give some examples of how to validate data inputs using Access.

5-7. What is a database schema? What is a database subschema? Give some examples of database schemas and subschemas for the payroll file of Figure 5-1.

5-8. What are data manipulation languages? How are these languages related to database management systems? How are these languages related to databases?

5-9. What is SQL? How is SQL like an Access query? How is it different?

5-10. What is online analytical processing? How is OLAP related to databases? What is a pivot table, and how are pivot tables and OLAP related?

5-11. What is the difference between ''sorting records'' and ''indexing records'' in a database?

5-12. What is ''data mining?'' How is data mining useful to profit-seeking companies? What are some accounting uses of data mining?

5-13. What are object-oriented databases? What are multimedia databases? How are these two types of databases alike? How are they different?

5-14. What are data warehouses? How are they like databases? How do they differ from databases?

5-15. Why would a company be interested in creating a data warehouse? Why would a company n*ot* be interested in creating a data warehouse?

PROBLEMS

5-16. Discuss both the advantages and disadvantages of using a computerized database system rather than a manual system for storing and processing accounting data. In your discussion, provide some specific accounting examples that illustrate your advantages and disadvantages.

5-17. What words are used to form each of the following acronyms?
(a) DBMS (b) DDL (c) DML (d) SQL (e) OLAP (f) OODB

5-18. The Wilmer Ruiz Corporation employs the individuals listed in the data shown in Figure 5-16. Use a DBMS to create a database of this information.

a. What record structure did you use for this database? Identify the names, widths, and other characteristics of each field you created.

b. List all employees in Department 5. Print this list.

c. List all employees with first name ''Brenda.'' Print this list.

d. List all those employees with pay rates over $6.50. Print this list.

e. List all those employees eligible for overtime (T = yes; F = no). Print this list.

5-19. Use the web to find business applications of data warehousing. Why do companies create data warehouses, and what are some accounting uses of such warehouses?

5-20. Use the web to find business applications of online analytical processing (OLAP). Why do companies use OLAP? What is the connection between OLAP and databases?

Record Number	Last Name	First Name	Social Security Number	Dept	Pay Rate	Over-time
1	ADCOX	NORMAN	901795336	1	6.50	Yes
2	KOZAR	LINDA	412935350	1	6.50	Yes
3	MCLEAN	KAY	405751308	1	7.50	No
4	CUNNINGHAM	TOM	919782417	3	7.50	Yes
5	DANIELS	PATRICIA	517351609	3	5.50	Yes
6	MCGUIRE	ANNE	201891647	3	5.50	Yes
7	REEDER	BRENDA	619294493	3	5.50	Yes
8	BLOOM	BRENDA	513321592	4	6.25	Yes
9	DAVIS	DENISE	517351608	4	5.50	Yes
10	DUFFY	LESLIE	314532409	4	8.50	No
11	HARPER	LINDA	615824130	4	5.75	Yes
12	MORGAN	MEREDITH	704563903	4	6.25	Yes
13	WELSH	KAREN	216253428	4	8.25	No
14	CHAPIN	GEORGE	203767263	5	7.50	Yes
15	FINN	JOHN	715386721	5	6.25	Yes
16	HALPIN	MARSHA	913541871	5	6.50	Yes
17	LAURIN	PHILIP	514484631	5	6.50	Yes
18	MIAGLIO	PEGGY	414224972	5	6.25	Yes
19	TURNER	BRENDA	713589164	5	8.50	No
20	ZORICH	MILDRED	504455827	5	6.50	Yes

FIGURE 5-16 Employees of the Wilmer Ruiz company.

Personnel File
Date: October 10, 20xx

	Employee Number	Score on Aptitude Test	Department ID	Current Pay Rate	Sex
BAKER, JEFFREY L.	1692	73	A	$7.50	M
BARRETT, RAYMOND G.	3444	53	B	7.45	M
BLISS, DONALD W.	6713	55	D	6.80	M
BOWERS, PAUL D.	2084	42	B	5.90	M
BUCHANAN, CINDY	3735	41	E	7.80	F
CHEUNG, WAI KONG	8183	55	C	7.80	F
CONRAD, MARK E.	8317	58	D	9.60	M
DAILY, REBECCA E.	2336	45	D	8.90	F
DRISCOLL, DAVID M.	5210	47	D	7.70	M
ERICKSON, KURT N.	2217	53	B	8.50	M
FRANTZ, HEIDI L.	6390	55	A	6.90	F
GARROW, SCOTT D.	8753	61	A	7.40	M
HARDENBROOK, LISA A.	7427	40	C	6.70	F
JACKSON, GREG W.	4091	67	D	8.90	M
LANGLEY, JERRY W.	3262	86	E	9.40	M
LUBINSKI, TRAVIS M.	3865	37	D	7.50	M
LYNCH, SHERENE D.	7857	66	D	8.90	F
MARKHAM, KYLE R.	6766	62	A	7.90	M
MCGUIRE, TANA B.	4052	55	A	9.20	F
MONACH, SHERI L.	8082	48	B	9.10	F
MOORE, MICHAEL S.	2431	67	E	8.50	M
NELSON, JOHN R.	5873	46	B	7.40	M
PAPEZ, PETER M.	7799	41	E	8.30	M
PETTINARI, DARIN M.	1222	56	B	8.40	M

FIGURE 5-17 Employee data for the Marcia Felix Corporation.

5-21. The information in Figure 5-17 is for the employees of the Marcia Felix Corporation. Use a DBMS software package to create a database for it.

a. What record structure did you design? Identify the names, widths, and other characteristics of each field in a typical record.

b. Sort these employees by department. Print this list.

c. Sort these employees by pay rate. Print this list.

d. Sort these employees by test score. Print this list.

e. Sort these employees by department and alphabetically by last name within department.

f. What is the average test score for these employees?

g. What is the average score for females? What is the average score for males?

h. What is the average pay rate for these employees?

i. What is the average pay rate for females? What is the average for males?

j. What females scored over 70 on their examinations? What males scored over 50?

5-22. Bonadio Electrical Supplies distributes electrical components to the construction industry. The company began as a local supplier 15 years ago and has grown rapidly to become a major competitor in the north central United States. As the business grew and the variety of components to be stocked expanded, Bonadio acquired a new computer and implemented an inventory control system and a computerized accounting sysem. Because of its operational importance, the inventory system has been upgraded to an online system, while all the other applications are operating in batch mode. Over the years, the company has developed or acquired more than 100 application programs and maintains hundreds of files.

Bonadio faces stiff competition from local suppliers throughout its marketing area. At a management meeting, the sales manager complained about the difficulty in obtaining immediate, current information to respond to customer inquiries. Other managers stated that they also had difficulty obtaining timely data from the system. As a result, the controller engaged a consulting firm to explore the situation. The consultant recommended installing a database management system (DBMS), and the company complied, employing Jack Gibbons as the database administrator.

At a recent management meeting, Gibbons presented an overview of the DBMS. Gibbons explained that the database approach assumes an organizational, data-oriented viewpoint, as it recognizes that a centralized database represents a vital resource. Instead of being assigned to applications, information is more appropriately used and managed for the entire organization. The operating system physically moves data to and from disk storage, and the DBMS is the software program that controls the data definition library that specifies the data structures and characteristics. As a result, both the roles of the application programs and query software, and the tasks of the application programmers and users are simplified. Under the database approach, the data are available to all users within security guidelines.

a. Explain the basic difference between a file-oriented system and a database management system.

b. Describe at least three advantages and at least three disadvantages of the database management system.

c. Describe the duties and responsibilities of Jack Gibbons, the database administrator.
(CMA Adapted)

CASE ANALYSES

5-23. Swan's Supplies (Normalizing Data)

Swan's Supplies is a wholesaler of sporting goods equipment for retailers in a local metropolitan area. The company buys sporting goods equipment direct from manufacturers and then resells them to individual retail stores in the regional area. The raw data in Figure 5-18 illustrate some of the information required for the company's purchase order system. As you can see, this information is characteristic of accounting purchase order

Purchase Order Number	Date	Customer Number	Customer Name	Customer Phone Number	Item Number	Item Description	Unit Cost	Unit	Quantity Ordered
12345	8/19/03	123-8209	Charles Dresser, Inc.	(752) 433-8733	X32655	Baseballs	$33.69	dozen	20
					X34598	Footballs	53.45	dozen	10
					Z34523	Bball Hoops	34.95	each	20
12346	8/19/03	123-6733	Patrice Schmidt's Sports	(673) 784-4451	X98673	Softballs	35.89	dozen	10
					X34598	Footballs	53.45	dozen	5
					X67453	Soccer balls	45.36	dozen	10

FIGURE 5-18 Some purchasing data for Swan's Sports Supplies.

systems but is not well organized. In fact, because of the repeating groups in the right-most columns, it cannot even be stored in a computer system.

Requirements

Store this data in a spreadsheet to make it easy to manipulate. Then perform each of the following tasks in turn:

1. Reorganize the data in first normal form and print your spreadsheet. Why is your data in first normal form?
2. Reorganize the data from part 1 into second normal form and print your spreadsheet. Why is your data in second normal form?
3. Reorganize the data from part 2 into third normal form and print your spreadsheet. Why is your data in third normal form?

5-24. Bonnie P Manufacturing Company (Data Validation Using a DBMS)

The payroll department at the Bonnie P Manufacturing Company has defined the following record structure for employee records.

Date Field	Data Type	Example
Last Name	Text	Kerr
First Name	Text	Stephen
Social Security number	Text	123-45-6789
Home phone number	Text	(987) 456-4321
Work phone extension	Number	123
Payrate	Currency	$12.34
Number of tax exemptions	Number	3
Department	Text	A

All fields are required. The employee's Social Security number serves as the record key. Work phone extensions are always greater than ''100'' and less than ''999.'' Pay rates are always at least $7.75 and no more than $29.85. The maximum number of tax exemptions allowed is ''10.'' Finally, there are only three departments: A, B, and C.

Requirements

1. Using a DBMS such as Access, create a record structure for the company.
2. Create data validation rules for as many data fields as you can. For each data validation rule, also create validation text that the system can use to display an appropriate error message. Create a list of such rules on a separate piece of paper.
3. Create employee records for yourself, and employees with the last names Anderson, Baker, and Chapman using data that you make up. Print this information.
4. Attempt to create one more record that violates a data validation rule. Create a screen capture of one or more violations, as suggested by your instructor.

5-25. Clifford Cohen University (Enforcing Referential Integrity)

Clifford Cohen University was founded as a small, liberal arts school just three years ago. Since that time, the institution has grown to the point where parking on campus is difficult and parking in illegal areas is common. Accordingly, the Board of Directors has reluctantly approved a policy requiring campus police to issue parking tickets.

Currently, the university requires students and faculty to register their cars with the parking office, which issues them parking decals that registrants must display inside the front windshield of their cars. At present, all record keeping at the parking office is done manually, severely limiting the ability of office personnel to create reports or perform meaningful statistical analyses about parking on campus. For example, it is currently not known how many students of each class (freshman, sophomore, etc.) register their cars or how many full-time faculty, part-time faculty, or clerical staff register their cars. The new policy of writing parking tickets will only add to this problem because it will require office staff to match parking tickets to student or faculty names. In addition, the Board of Directors would like an end-of-semester report indicating how many parking violations of each type (meter violation, invalid parking sticker, etc.) are issued by the campus police.

To help solve these problems, the University Board of Directors has hired you to create a computerized system for them. You realize that a database system might work for this, and accordingly propose a database of tables with record structures similar to those in Figure 5-4. The Board of Directors approves your plan, but asks that you create a small system to demonstrate its features before creating a full-blown system.

Requirements:

1. Use Microsoft Access (or an alternate DBMS designated by your instructor) to create the three tables illustrated in Figure 5-4. What data type did you specify for each data field in each table?
2. Create at least three records in the car registration table. Be sure to use your own name as one of the registrants. Also, create at least three records for the Parking Violations Code File. Make up your own fine amounts instead of using the ones shown in the figure.
3. For each record you create in the car registration file in step 2 above, create at least three parking tickets and input this information to the Tickets File. Thus, you should have at least nine records in this file. Be sure that at least one record in the Tickets File contains a reference to each of the records in the Parking Violations Code File (i.e., at least one person breaks every possible parking violation). Print copies of the records in each table for your instructor.
4. Attempt to create a record in the Ticket File that contains a nonexistent ticket code in the parking Violations Code File. Were you successful?
5. Link the tables together, following the directions in Chapter 4. (Be sure to check ''enforce referential integrity'' when you see the dialog box illustrated in Figure 4-19.) When you finish, your relationships window should resemble the one shown in Figure 5-10. What are the relationships among the records in the three tables? Print a copy of this window as documentation for your project.
6. Now return to the Tables portion of Access and display the Car Registration table. You should now see the plus symbols illustrated in Figure 4-18. Click on one of these symbols. Are you able to view the linked records?

7. Now again attempt to create a record in the ticket file that contains a nonexistent ticket code in the parking Violations Code File. Were you successful this time?
8. Finally, attempt to delete a record in the Parking Violations Code File. Why can't you do it?
9. If required by your instructor, create an example of the parking-violations-by-type report desired by the Board of Directors using the database you just created.

5-26. BSN Bicycles II (Creating Queries in Access)

Business has been growing at BSN Bicycles, and the store owners have been using their Access database to store information about their customers. Now that the store is a little more established, the owners are thinking more about how best to attract more customers to their store. One idea is to see where their current customers live. The owners also want a complete list of their credit customers.

Requirements:

1. If you have not already done so, create a database for BSN and the customer's table described in Case 4–25 in Chapter 4. Be sure to create at least 10 customer records for the company, including one with your name. Several of the customers should also live in the state of Virginia (VA) and several customers should have zip code "12345." The Virginia customers and the customers with zip code 12345 do not have to be the same.
2. If you have not already done so, create several invoices for your customers.
3. Create a query that selects all customers living in Virginia. Print your results.
4. Create a query that selects all customers living in zip code 12345. Print your results.
5. Create a query that selects all customers living in Virginia who also have zip code 12345. Print your results.
6. Create a query that selects all credit customers. (Hint: use the word "Yes" for the criteria in this query.) Print your results.

5-27. Furry Friends Foundation II (Creating Queries for Databases)

Recall from Case 4–21 in Chapter 4 that the Furry Friends Foundation is a nonprofit organization that finds homes for abandoned animals. The foundation has recently computerized some of its operations by storing its accounting data in a relational database. One reason for this was to enable it to more easily answer questions about donations. This portion of the case provides some examples of such questions and gives you practice creating database queries to answer them.

Requirements:

1. If you have not already done so, create the tables and relationships described in Case 4–21.

2. Using *Access* or similar software as required by your instructor, create three donations for yourself. You should donate to dogs in one contribution, cats in the second contribution, and unspecified (''other'') in the third contribution.
3. Create a query that selects all customers donating to cats. Print your results.
4. Create a query that selects all contributors who donated over $50. Print your results.
5. Create a query that selects all contributors who donated over $100 to dogs. Print your results.

REFERENCES AND RECOMMENDED READINGS

Adelman, Sid and Suzan, Dennis. ''Capitalizing the Data Warehouse'' *DM Review* Vol. 15, No. 7 (July 2005), pp. 26–32.

Anonymous. ''Anonymous Tales from the Front Lines: Do It Fast or Do It Right,'' *Infoworld* Vol. 28, No. 9 (February 26, 2006), p. 46.

Bange, Carsten. ''Commentary: The OLAP Revival'' *the OLAP Report* (2008), accessible at http://www.olapreport.com/Comment_OLAP_revival.htm.

Borthick, A. Faye and Donald Jones. ''Analyzing a Potential Warranty Call Center Budget Overrun: Using Database Queries to Solve Business Problems,'' *Journal of Information Systems* Vol. 19, No. 1 (Spring 2005), pp. 97–111.

Borthick, A. Faye, and Mary B. Curtis. ''Due Diligence on Fast Fashion Inventory Through Data Querying'' *Journal of Information Systems* Vol. 22, No. 1 (Spring 2008), pp. 77–93.

Calderon, Thomas G., John J. Cheh, and Il Woon Kim. ''How Large Corporations Use Data Mining to Create Value'' *Management Accounting Quarterly* Vol. 4, No. 2 (Winter 2003), pp. 2–13.

Crosman, Penny. ''Database Bubble?'' *Intelligent Enterprise* Vol. 9, No. 7 (July 2006), p. 15.

Fadairo, S. A., Rosemary Williams, Ronald Trotman, and Anthony Onyekelu-Eze. ''Using Data Mining to Ensure Payment Integrity'' *Journal of Government Financial Management* Vol. 57, No. 2 (Summer 2008), pp. 22–24.

Garbellotto, Gianluca. ''The Data Warehousing Disconnect'' *Strategic Finance* Vol. 89, No. 4 (October 2007), p. 59–61.

Helms, Michael. ''Spanning the Business'' *Best's Review* Vol. 105, No. 11 (March 2005), p. 44.

Keating, Barry. ''Data Mining: What is it and How is it Used?'' *Journal of Business Forecasting* Vol. 27, No. 3 (Fall 2008), pp. 33–35.

Kimball, Ralph, and Margy Ross. ''Differences of Opinion (on Data Warehousing)'' *Intelligent Enterprise* Vol. 7, No. 3 (March 6, 2004), pp. 16–18.

Lamoreaux, Matthew. ''Internal Auditor Used Computer Tool to Detect Worldcom Fraud'' *Journal of Accountancy* Vol. 204, No. 1 (July 2007), p. 35.

Landry, Raymond Jr., Roger Debreceny, and Glen L. Gray. ''Grab Your Picks and Shovels: There's Gold in Your Data'' *Strategic Finance* Vol. 85, No. 7 (January 2004), pp. 25–8.

Martin, Lisa, Trisha Bruce, J. Kyle Snyder, Steve Gay, and Ken Chaffman. ''Sharing Data'' *Geospatial Solutions* Vol. 15, No. 3 (February 2005), pp. 26–31.

Milligan, Jack. ''Data Chase'' *Banking Strategies* Vol. 81, No. 5 (September/October 2005), pp. 5–19.

Perry, Gail. ''At Your Fingertips: the Next Generation of Usage in Data Warehousing'' *Practical Accountant* (no volume or issue numbers), (May 2005), pp. 6, 23.

Ross, Margy, and Kimball, Ralph. ''Data Warehouse Dining Experience'' *Intelligent Enterprise* Vol. 7, No. 1 (January 1, 2004), pp. 12–14.

Slansky, Dick. ''Object-Oriented Databases: the Next Wave in Complex Data Management'' *Manufacturing Business Technology* Vol. 25, No. 6 (June 2007), p. 51.

Tittel, Ed. ''Managing Data with Database Tools'' *Certification Magazine*g Vol. 7, No. 8 (August 2005), pp. 20, 39.

ANSWERS TO TEST YOURSELF

1. **c** 2. **c** 3. **c** 4. **a** 5. **a** 6. **b** 7. **d** 8. **d** 9. **b** 10. **b** 11. **c**