5 Database and Content Management

Running Case

It had been more work than Marla thought, but she had finally finished upgrading the existing computers in The 1881 and initiated a tablet loaner program where guests could borrow an iPad to use during their stay. Fresh from this success, Marlo's next move was to get a better idea of the guests who were staying at The 1881. The old Hotel Reservation and Management System (version 2.0) was stable enough, but did not provide the ability to analyze guest information other than sorting the list by name, address, and telephone number. The updated version of the hotel reservation and management software had more advanced features, but the owners of The 1881 indicated that it was too expensive to purchase now.

Marlo was stuck. How could she get better information about the B&B hotel's clients? She decided to investigate and found that the current Hotel Reservation and Management System was built on a relational database, something she had learned about in her introductory MIS course. She had learned that the database could export names, vacation dates, room numbers, and even services that the guests enjoyed. She needed to get administrative access to the database. She argued successfully that since she was exporting information that was already stored in the system, there was no impact on the current system and no chance for her to delete or modify existing data.

The exported data were provided in a format that could be read by a Microsoft Excel spreadsheet. She was thrilled the first time she opened the spreadsheet and was able to see the guests' names and other information. However, after a few minutes of exhilaration, she realized that there may be some challenges. She needed to learn more about how data were stored before she could do much with the data.

After a few weeks of working on the data, Marlo had created an interesting set of spreadsheets. In one spreadsheet, she had carefully listed all of the rooms in the B&B hotel. In another spreadsheet, she had a list of the services provided by the hotel. In yet another spreadsheet, she had a list of all of the activities that guests had been billed for over the past three years, by date, along with the names of guests who purchased them. Another spreadsheet contained alphabetically listed names and the addresses of almost all of the guests who had stayed at The 1881. It was a lot of information spread over four different spreadsheets. Her problem now was finding a way to get all of this data to make sense.

Racorn/Shutterstock

Luckily, she had kept her introductory MIS textbook, which included a chapter on database design—and even a tutorial on using Microsoft Access (a program that helps create relational databases). She read through the chapter on designing relational tables, and within a week she was able to combine the four separate spreadsheets she had created into a single database. She now had the ability to consider such questions as "How many of our guests stay more than twice a year?" for which the database could provide answers within seconds.

To prepare for her next meeting with the owners, Marlo used the database to prepare answers to such questions as the following: "Who are The 1881's most frequent guests?" "Which guests buy the most services?" "Which guests have stopped coming to The 1881?" She went into the meeting with a great feeling, knowing she was not only learning more about the B&B but also that she had developed valuable skills in analyzing information that would be used throughout her career.

Study Questions

Q1 **What is content?**

Q2 **How can content be organized?**

Q3 **What is the purpose of a database?**

Q4 **What does a database contain?**

Q5 **What is a DBMS, and what does it do?**

Q6 **What is a database application?**

Q7 **What is the difference between an enterprise DBMS and a personal DBMS?**

Q1 What Is Content?

Content can be difficult to define. In the broadest sense, content is something of value and can be considered an asset just like other items of property. It is often closely related to **intellectual property**, which, in Canada, is defined as a form of creative endeavour that can be protected through a trademark, patent, copyright, industrial design, or integrated circuit topography.[1] Content varies by industry. In the advertising industry, content refers to the pictures, commercials, and text used to promote ideas about products and services. In the publishing industry, content refers to words. In the banking industry, content is account information.

Before the advent of computers, content was only available on physical assets such as paper, photographs, or film. But computers create digital content that can be stored, and networks, such as the internet, can distribute this content. Organizations have databases that store large amounts of data related to customers, employees, orders, and so on. Organizations also store a lot of other content. Word-processing documents (.doc, .txt, .pdf), spreadsheets (.xls), and presentations (.ppt) are a part of everyday work. Other content might include webpages (.htm, .html), text from blogs, Twitter, or discussion boards, graphics (.jpg, .gif, .bmp, .png, etc.), video files and video logs (.WMV, .AVI, and .MPG), audio files (.WAV, .MP3, .ACC, and .WMA), and even geographical information available through such applications as Google Earth. The expanding volume of content and the growing number of formats in which it is provided can make it difficult for individuals and corporations to effectively utilize that content. Managing content is, therefore, an important challenge for businesses to understand and appreciate.

The challenge today is not collecting and distributing content but also in presenting it appropriately for various stakeholders inside and outside of organizations. A company's website has become an important source of content for both customers and employees. Students who concentrate in marketing recognize that websites help to brand organizations. Websites have also become a critical part of customer support. It was hard enough to manage data and information when it was exclusively contained within a company and used only by employees—as information has become more available to other stakeholders, it has become increasingly difficult to manage the increased volume, format, and presentation choices for content.

Q2 How Can Content Be Organized?

The challenge in content management is indexing or cataloguing the right information, processing and storing it, and then getting it to the right person in the right format at the right time. One way of thinking about content management is to separate the *management* of content from the *presentation* of content. We learned in the previous chapter that all data in computing systems are represented by bytes. Content management focuses on how to efficiently and effectively store and process these bytes. Content presentation focuses on how best to present data to the person using the system.

The management of many types of content has traditionally been handled through organizational database management systems (DBMSs). DBMSs are central to the management of content data, and we will learn more about them later in the chapter. The presentation of content has gone through changes as company websites have matured. In the early days of the web, employees might have been

[1] You can read more about intellectual property at www.cipo.ic.gc.ca.

able to post content directly to a company's website. This practice did not provide a consistent look and feel and left the company at risk if incorrect data were posted. As organizational websites became more complex, employees could not be expected to keep up with all of the changes. The presentation of content in organizations today is increasingly handled through a series of steps supported by software. **Content management systems (CMSs)** have been developed to help companies organize this process.

When an employee wants to place some content on the organization's website, he or she will access the web CMS. The web CMS of a company is usually located on its website server. The employee typically loads the raw content into the web CMS. A copyeditor then reviews the document and makes any needed changes. He or she then passes the content on to layout artists, who prepare the content for presentation. The content and presentation are stored with the help of a DBMS. The manager in charge of the website will then review the content and presentation and publish the work to the public website. The web CMS helps manage each step of this process and enables a company to standardize the look and feel of a website and control the information available to customers and employees.

CMSs have also evolved. They have grown beyond their original role of simply organizing documents for corporate websites. These systems now actively seek out documents located across an organization and automatically manage access to this content. Media files, word-processing documents, html pages, and many other documents can all be categorized and searched by CMSs. This capability allows for the increased organization of a wider range of a corporation's data assets. Current CMSs also handle document archiving and the increased use of electronic files for document management. OpenText (see box below), a Canadian company located in Waterloo, Ontario (www.opentext.com), and EMC, a U.S. company (www.emc.com), are examples of these CMSs.

OPENTEXT: FROM SPINOFF TO MARKET LEADER

How does a small spinoff company from the University of Waterloo grow to become Canada's largest software company and the world leader in enterprise content management systems? It all started with a project to bring the *Oxford English Dictionary (OED)* into the computer age. The *OED* had become so large that it was unwieldy to update. Researchers at the University of Waterloo, with funding help from the Canadian government, worked to build full-text indexing and string-search technology for the *OED*. The project resulted in a product that was close to a web-based search engine—in 1989, years before web search engines were well known. OpenText was started in 1991. The company continued to develop increased functionality in the search engine through 1995. When management believed that the market for search engines no longer looked promising, the company turned to document management systems. (Astute students may note that Google, the current leader among internet search engines, was founded in 1998 and has a value almost 50 times greater than that of OpenText. In business, as in many areas, timing is everything.) Web-based document management systems proved to be a lucrative market, and OpenText grew from a company of 20 employees in 1995 to a company of more than 5000 employees with over $1.3 billion in sales supporting 100 million users in 114 countries by 2014. The company has continued its rapid growth and is recognized as the market leader in enterprise content management.

Source: You can find out more about the history of OpenText at www.opentext.com/corporate/our_history.html.

Watch

Go to MyMISLab to watch a video about the purpose of a database.

Watch

Go to MyMISLab to watch a video about database processing.

Q3 What Is the Purpose of a Database?

A database keeps track of things. When most people become aware of this, they wonder why we need a special technology for such a simple task. Why not just use a list? And if the list is long, can it just be put in a spreadsheet?

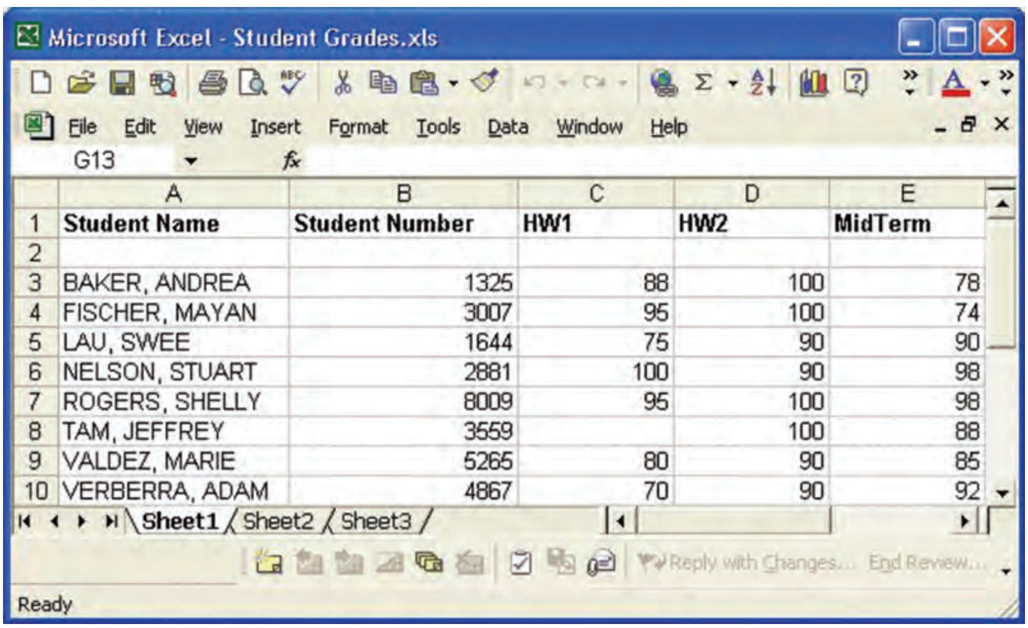
Many professionals do keep track of things using spreadsheets. If the structure of the list is simple enough, there is no need to use database technology. The list of student grades in Figure 5-1, for example, works perfectly well in a spreadsheet.

Suppose, however, that the professor wants to track more than just grades. He or she may want to record email messages as well. Or, perhaps, the professor wants to record both email messages and office visits. There is no place in a spreadsheet, such as the one in Figure 5-1, to record these additional data. Of course, the professor could set up a separate spreadsheet for email messages and another for office visits, but that awkward solution would be difficult to implement because it does not provide all the data in one place.

Instead, the professor may want a form similar to the one shown in Figure 5-2. With it, he or she can record student grades, emails, and office visits all in one place. Technically it might be possible to create a similar form like this in a spreadsheet, but with a database, it is much easier to develop and maintain.

The key distinction between Figures 5-1 and 5-2 is that the list in Figure 5-1 is about a single theme or concept—student grades. The list in Figure 5-2 has multiple themes—it shows student grades, emails, and office visits. We can create a general rule from these examples: Lists that involve a single theme can be stored in a spreadsheet; lists that involve multiple themes require a database. We will learn more about this general rule later in this chapter.

To summarize, the purpose of a database is to keep track of things that involve more than one theme.



	A	B	C	D	E
1	**Student Name**	**Student Number**	**HW1**	**HW2**	**MidTerm**
2					
3	BAKER, ANDREA	1325	88	100	78
4	FISCHER, MAYAN	3007	95	100	74
5	LAU, SWEE	1644	75	90	90
6	NELSON, STUART	2881	100	90	98
7	ROGERS, SHELLY	8009	95	100	98
8	TAM, JEFFREY	3559		100	88
9	VALDEZ, MARIE	5265	80	90	85
10	VERBERRA, ADAM	4867	70	90	92

Figure 5-1
A List of Student Grades

Source: Microsoft Excel

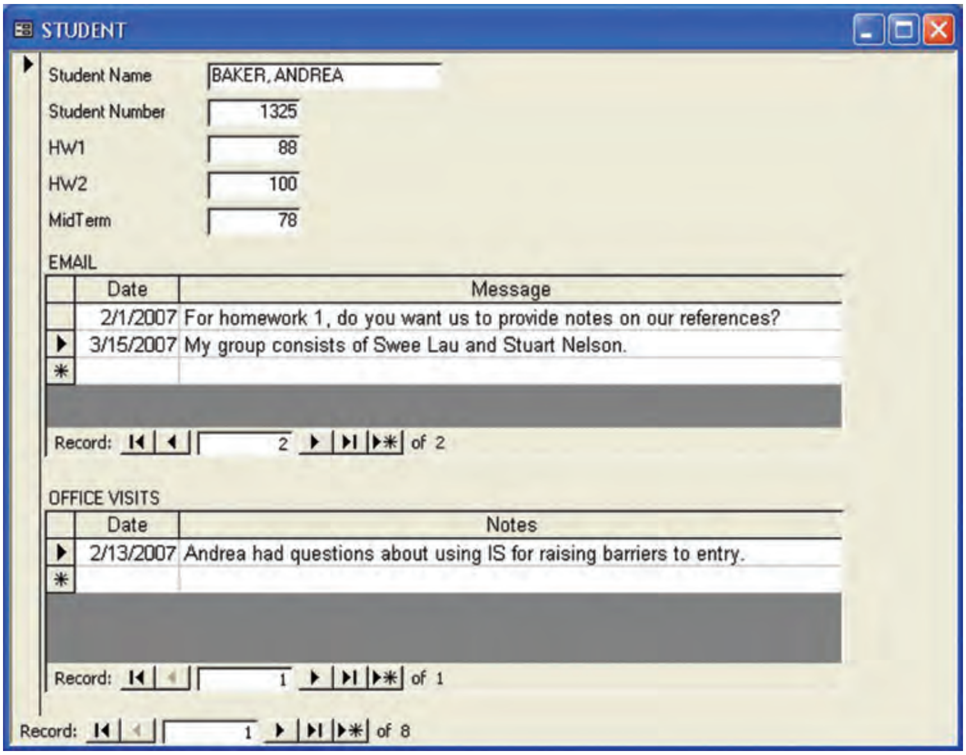


Figure 5-2

Student Data Shown in Form of a Database

Source: Microsoft Excel

Q4 What Does a Database Contain?

Database design is a specialized skill that everyone in the field of management information systems (MIS) should understand and any business student who plans to work with corporate data should be familiar with. You will learn the basics in this chapter. In addition, two extensions to this chapter, "Database Design" and "Using Microsoft Access," contain more in-depth information on this topic.

A **database** is a self-describing collection of integrated records. To understand this definition, you first need to understand the terms illustrated in Figure 5-3. As you learned in Chapter 4, a **byte** is a character of data. Bytes are grouped into **columns**, such as *Student Number* and *Student Name*. Columns are also called **fields**. Columns or fields, in turn, are grouped into **rows**, which are also called **records**. In Figure 5-3, the collection of data for all columns (*Student Number, Student Name, HW1, HW2*, and *MidTerm*) is called a *row* or a *record*. Finally, a group of similar rows or records is called a **table** or a **file**. From these definitions, you can see that there is a hierarchy of data elements, as shown in Figure 5-4.

It is tempting to continue this grouping process by saying that a database is a group of tables or files. This statement, although true, does not go far enough, however. As shown in Figure 5-5, a database is a collection of tables *plus* relationships among the rows in those tables, *plus* special data, called *metadata*, that describe the structure of the database. By the way, the cylindrical symbol represents a computer disk drive. It is used in diagrams, such as that in Figure 5-5, because databases are very often stored on magnetic disks.

Relationships Among Records

Consider the terms on the left side of Figure 5-5. You know what *tables* are. To understand what is meant by *relationships among rows in tables*, examine Figure 5-6. It shows

Figure 5-3
Student Table (also called a *file*)

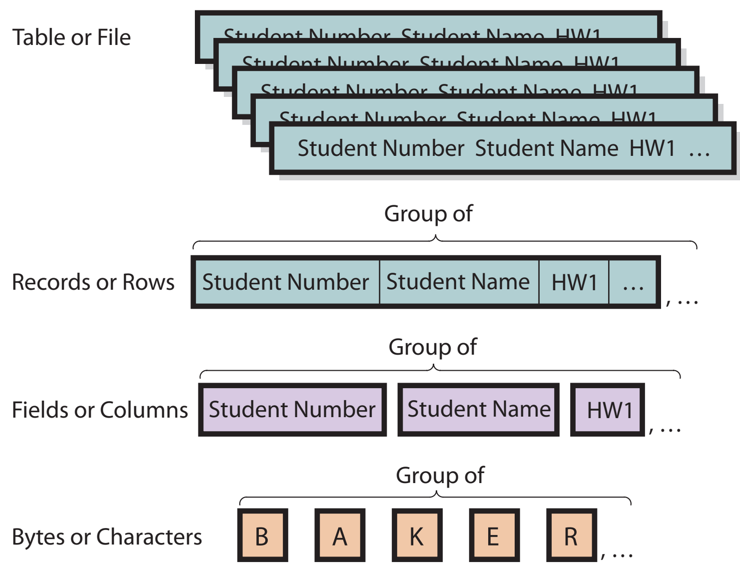
Columns, also called *fields*

Rows, also called *records*

Student Number	Student Name	HW1	HW2	MidTerm
1325	*BAKER*, ANDREA	88	100	78
1644	LAU, SWEE	75	90	90
2881	NELSON, STUART	100	90	98
3007	FISCHER, MAYAN	95	100	74
3559	TAM, JEFFREY		100	88
4867	VERBERRA, ADAM	70	90	92
5265	VALDEZ, MARIE	80	90	85
8009	ROGERS, SHELLY	95	100	98

Characters, also called *bytes*

Figure 5-4
Hierarchy of Data Elements



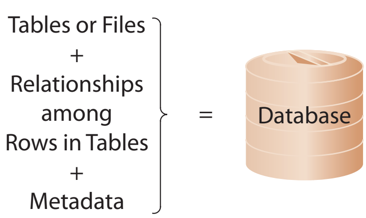


Figure 5-5
Components of a Database

sample data from the three tables *Email*, *Student*, and *Office_Visit*. Note the column named *Student Number* in the *Email* table. That column indicates the row in the *Student* table to which a row of *the Email* table is connected. In the first row of the *Email* table, the *Student Number* value is 1325. This indicates that this particular email was received from the student whose *Student Number* is 1325. If you examine the *Student* table, you will see that the row for Andrea Baker contains this value. Thus, the first row of the *Email* table is related to Andrea Baker.

Now consider the last row of the *Office_Visit* table at the bottom of the figure. The value of *Student Number* in that row is 4867. This value indicates that the last row in *Office_Visit* belongs to Adam Verberra.

From these examples, you can see that values in one table relate rows of that table to rows in a second table. Several special terms are used to express these ideas. A **key** is a column or group of columns that identifies a unique row in a table. *Student Number* is the key of the *Student* table. Given a value of *Student Number*,

Email Table

EmailNum	Date	Message	Student Number
1	2/1/2007	For homework 1, do you want us to provide notes on our references?	1325
2	3/15/2007	My group consists of Swee Lau and Stuart Nelson.	1325
3	3/15/2007	Could you please assign me to a group?	1644

Student Table

Student Number	Student Name	HW1	HW2	MidTerm
1325	BAKER, ANDREA	88	100	78
1644	LAU, SWEE	75	90	90
2881	NELSON, STUART	100	90	98
3007	FISCHER, MAYAN	95	100	74
3559	TAM, JEFFREY		100	88
4867	VERBERRA, ADAM	70	90	92
5265	VALDEZ, MARIE	80	90	85
8009	ROGERS, SHELLY	95	100	98

Office_Visit Table

VisitID	Date	Notes	Student Number
2	2/13/2007	Andrea had questions about using IS for raising barriers to entry.	1325
3	2/17/2007	Jeffrey is considering an IS major. Wanted to talk about career opportunities.	3559
4	2/17/2007	Will miss class Friday due to job conflict.	4867

Figure 5-6
Example of Relationships among Rows

you can determine one and only one row in *the Student table*. Only one student has the number 1325, for example.

Every table must have a key. The key of the *Email* table is *EmailNum*, and the key of the *Office_Visit* table is *VisitID*. Sometimes more than one column is needed to form a unique identifier. In a table called *City*, for example, the key would consist of the combination of columns (*City, Province*) because a given city name can appear in more than one province.

Student Number is not the key of the *Email* or the *Office_Visit* tables. We know that about *the Email table* because there are two rows in *Email* that have the *Student Number* value 1325. The value 1325 does not identify a unique row; therefore, *Student Number* is not the key of the *Email* table.

Nor is *Student Number* a key of the *Office_Visit* table, although you cannot tell that from the data in Figure 5-6. If you think about it, however, there is nothing to prevent a student from visiting a professor more than once. If that were to happen, there would be two rows in the *Office_Visit* table with the same value of *Student Number*. It just happens that no student has visited twice in the limited data in Figure 5-6.

Columns that fulfill a role like that of *Student Number* in the *Email* and *Office_Visit* tables are called **foreign keys**. This term is used because such columns are keys, but they are keys of a different (foreign) table from the one in which they reside.

Before we go on, note that databases that carry their data in the form of tables and that represent relationships using foreign keys are called **relational databases**. (The term *relational* is used because another, more formal name for a table is **relation**.) In the past, databases existed that were not relational in format, but such databases

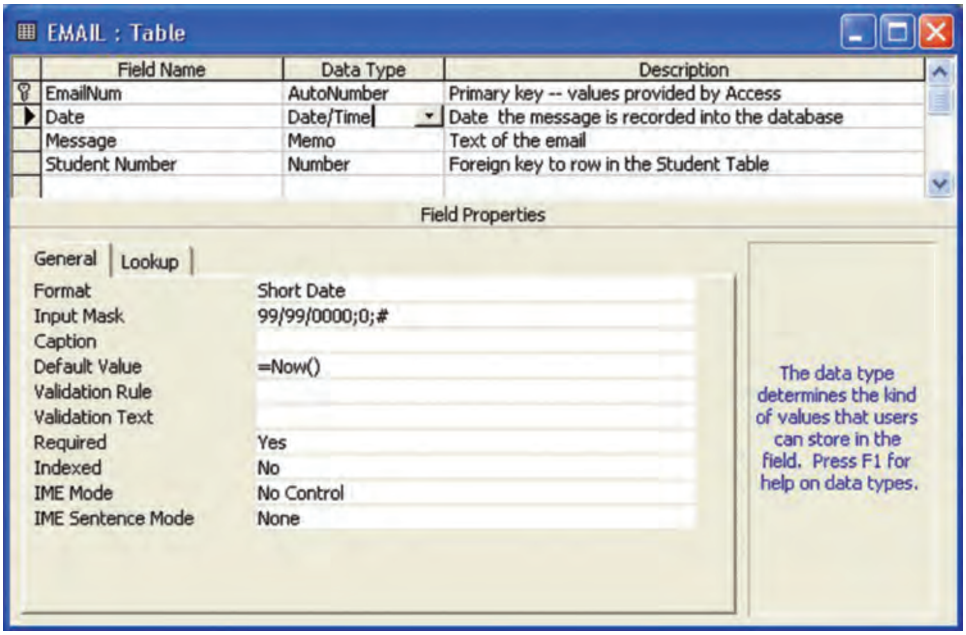


Figure 5-7
Example of Metadata (in Access)

Source: Microsoft Access

have nearly disappeared. Chances are you will never encounter one, and we will not consider them further.[2]

Metadata

Recall the definition of *database*—a self-describing collection of integrated records. The records are integrated because, as you have just learned, relationships among rows are represented in the database. But what does *self-describing* mean?

It means that a database contains, within itself, a description of its contents. Think of a library. A library is a self-describing collection of books and other materials. It is self-describing because the library contains a catalogue that describes its contents. The same idea also holds true for a database. They are self-describing because they contain not only data but also data about the data in the database.

Metadata are data that describe data. Figure 5-7 shows metadata for the *Email* table. The format of metadata depends on the software product that is processing the database. Figure 5-7 shows the metadata as they appear in Microsoft Access. Each row of the top part of this form describes a column of the *Email* table. The columns of these descriptions are *Field Name, Data Type*, and *Description*. *Field Name* contains the name of the column, *Data Type* shows the type of data the column may hold, and *Description* contains notes that explain the source or use of the column. As you can see, there is one row of metadata for each of the four columns of the *Email* table: *EmailNum, Date, Message*, and *Student Number.*

The bottom part of this form provides more metadata, which Access calls *Field Properties*, for each column. In Figure 5-7, the focus is on the *Date* column (note the filled-in right-face pointer next to its name, such as the one shown here ▶). Because the focus is on *Date* in the top pane, the details in the bottom pane pertain to the *Date* column. The *Field Properties* describe formats, a default value for Access to supply when a new row is created, and the constraint that a value is required for this column. It is not important for you to remember these details. Instead, just

[2] Another type of database, the object-relational database, is rarely used in commercial applications. Search the web if you are interested in learning more about object-relational databases. In this book, we will consider only relational databases.

understand that metadata are data about data and that such metadata are always a part of a database.

The presence of metadata makes databases much more useful than spreadsheets or data in other lists. Because of metadata, no one needs to guess, remember, or even record what is in the database. To find out what a database contains, we just look at the metadata inside the database. Metadata make databases easy to use—for both authorized and unauthorized purposes, as described in the exercise "Nobody Said I Shouldn't" at the end of this chapter on pages 154–155.

Q5 What Is a DBMS, and What Does It Do?

A database, all by itself, is not very useful. The tables in Figure 5-6 have all the data the professor wants, but the format is unwieldy. The professor wants to see the data in a form like that in Figure 5-2 and also as a formatted report. Pure database data are correct but in raw form, they are not pertinent or useful.

Figure 5-8 shows the components of a **database application system**. Such applications make database data more accessible and useful. Users employ a *database application* that consists of forms (such as the form in Figure 5-2), formatted reports, queries, and application programs. Each of these, in turn, calls on the DBMS to process the database tables. We will first describe DBMSs and then discuss database application components.

The Database Management System

A **database management system (DBMS)** is a program used to create, process, and administer a database. As is the case with operating systems, almost no organization develops its own DBMS. Instead, companies license DBMS products from vendors, such as IBM, Microsoft, and Oracle. Popular DBMS products are **DB2** from IBM, **Access** and **SQL Server** from Microsoft, and **Oracle** from Oracle Corporation. Another popular DBMS is **MySQL**, an open-source DBMS product that is free for most applications. Other DBMS products are available, but the five listed above account for the vast majority of databases on the market today.

Note that a DBMS and a database are two different things, even though many in the trade press, and even some books, confuse the two. A DBMS is a software program; a database is a collection of tables, relationships, and metadata. The two concepts are very different.

Creating the Database and Its Structures

Database developers use the DBMS to create tables, relationships, and other structures in the database. The form in Figure 5-7 can be used to define a new table or to modify an existing one. To create a new table, the developer just fills out a new form, such as the one in Figure 5-7.

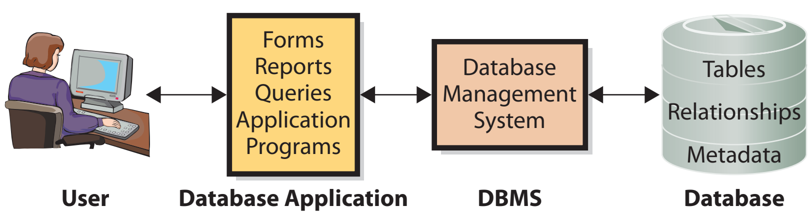
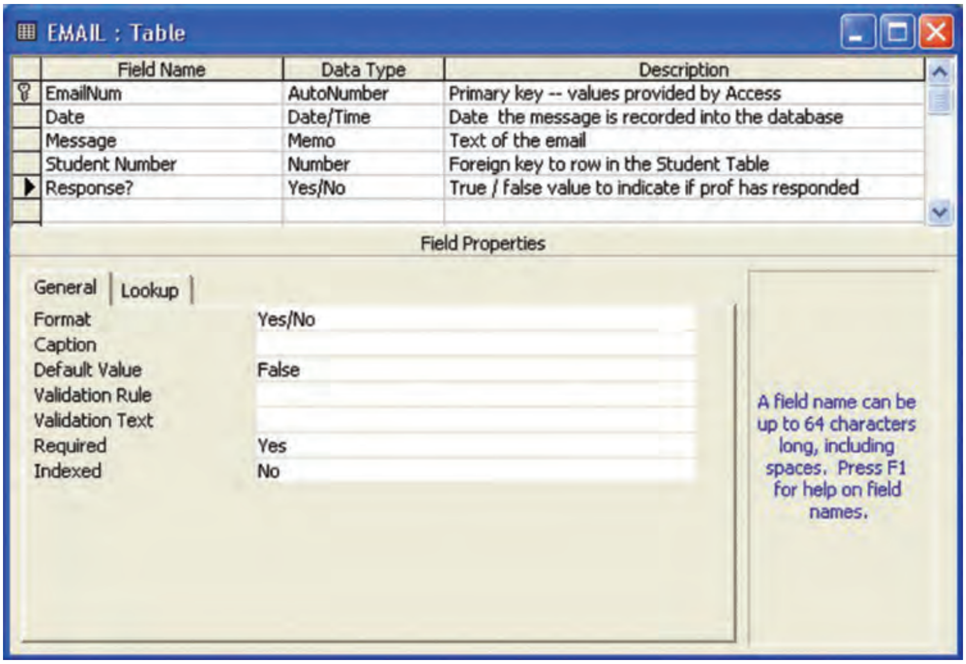


Figure 5-8
Components of a Database Application System

Figure 5-9

Adding a New Column to a Table (in Access)

Source: Microsoft Access 2010



To modify an existing table—for example, to add a new column—the developer opens the metadata form for that table and adds a new row of metadata. For example, in Figure 5-9, the developer has added a new column called *Response?* This new column is created by adding the label "Response?" under the Field Name column. This new column has the Data Type *Yes/No*, which means that the column can contain only one of the values—yes or no. A professor will use this column to indicate whether he or she has responded to the student's email. Other database structures are defined in similar ways.

Processing the Database

The second function of the DBMS is to process the database. Applications use the DBMS for four operations: *read*, *insert*, *modify*, or *delete* data. The applications call upon the DBMS in different ways. For example, when the user enters new data or changes data on a form, a computer program processes the data provided on the form and then calls the DBMS to make the necessary database changes. At other times, an application program can call directly on the DBMS to make the change. No matter which way the database is called, there is only one language that relational databases use when communicating data from a database: **Structured Query Language (SQL)**, which is an international standard language for processing a database. A query can be thought of as a question. SQL (pronounced "see-quell") can then be thought of as a formal way of putting a question to a database. The answer to that query will be the data that is specified. All five of the DBMS products mentioned earlier accept and process SQL statements. As an example, the following SQL statement inserts a new row into the *Student* table:

```
INSERT INTO Student
 ([Student Number], [Student Name], HW1, HW2, MidTerm)
 VALUES
 (1000, 'Franklin, Benjamin', 90, 95, 100)
```

Such statements are issued behind the scenes by programs that process forms. Alternatively, they can also be issued directly to the DBMS by an application program.

You do not need to understand or remember SQL language syntax. Instead, just be aware that SQL is an international standard for processing a database. As well, SQL can be used to create databases and database structures. You will learn more about SQL if you take a database management course.

Administering the Database

A third DBMS function is to provide tools to assist in the administration of the database. Database administration involves a wide variety of activities. For example, the DBMS can be used to set up a security system involving user accounts, passwords, permissions, and limits for processing the database. To provide database security, a user must sign on using a valid user account before he or she can process the database.

Permissions can be limited in very specific ways. In the *Student* database example, it is possible to limit a particular user to reading only *Student Name* from the *Student* table. A different user could be given permission to read the entire *Student* table, but limited to update only the *HW1, HW2*, and *MidTerm* columns. Other users can be given still other permissions.

In addition to security, DBMS administrative functions include backing up database data, adding structures to improve the performance of database applications, removing data that are no longer wanted or needed, and similar tasks. One of these tasks involves setting up a system for dealing with database growth, as discussed in "MIS in Use," on page 144.

Q6 What Is a Database Application?

A **database application** is a collection of forms, reports, queries, and application programs that process a database. A database may have one or more applications, and each application may have one or more users. Figure 5-10 shows three applications; the top two have multiple users. These applications have different purposes, features, and functions, but they all process the same inventory data stored in a common database.

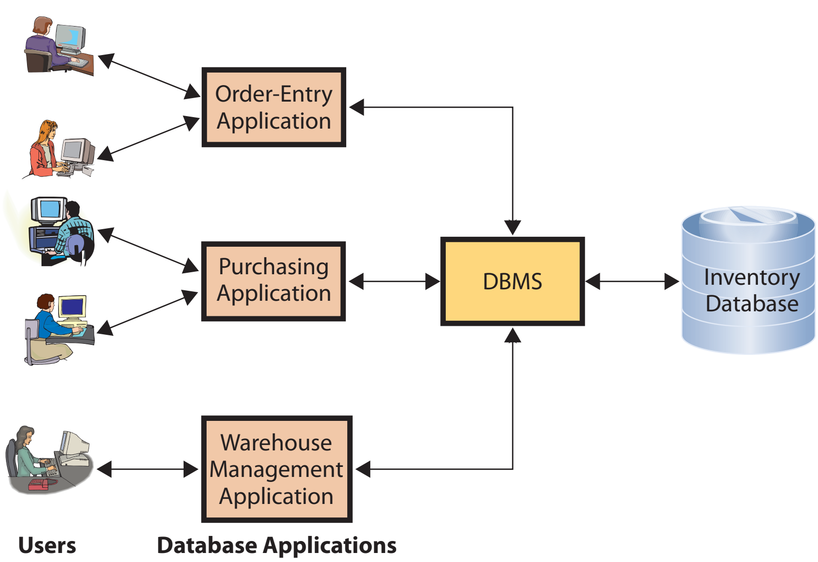


Figure 5-10
Use of Multiple Database Applications

MIS in Use

The Many Names of One Customer

Founded in 1945, Vancouver-based Vancity is Canada's largest credit union, with more than $16 billion in assets. By a combination of organic (natural) and inorganic (acquisition) growth, Vancity now has 57 branches in Metro Vancouver, the Fraser Valley, Victoria, and Squamish and 501 000 individual and business customers.

The majority of Vancity's member customers did not have just a single product or service but, rather, a variety of products and services that could include savings and chequing accounts, loans, credit cards, and mutual funds and other investment products. Indeed, further complicating the relationship was that customers not only had multiple products/services but also had multiple instances of individual products/services. That is, a customer could have two savings accounts, multiple credit cards, and a number of mutual funds in a variety of ownerships (such as registered retirement savings programs [RRSPs], registered education savings plans [RESPs], and nonregistered investment plans), and these could be held or have been set up at different branches.

This diversity of products and services—although attractive to both Vancity and its member customers—created a major data-quality headache for Tony Fernandes. As the former vice-president of technology strategy, one of his responsibilities was the overall quality of information. His challenge was to ensure that the data in the customer information file (CIF), the database that held all customer data, were accurate and that the CIF identified customers uniquely and completely. As he put it, "My job is to manage similarities and differences. We need to know if the Jon Doe who lives on Victoria Street and has a savings account is the same Jonathan Doe who has a business account and a residence on Boundary Road."

The challenge was significant. In many cases, names were not unique and were complicated by short forms or by people having a variety of legal, given, and familiar names.

Vancity attempted to resolve many of these problems as customers activated each new product or service, but it was not always feasible. Something as relatively simple as spelling an address could result in duplicate entries that had to be reconciled. For example, the address "35 Westforest Trail" could also be entered as "35 Westforest Tr." At the lowest level, these types of entries caused inefficiencies, such as sending duplicate information. More troubling to Tony, of course, were problems of incomplete customer information or more complicated issues, such as misidentification of financial records.

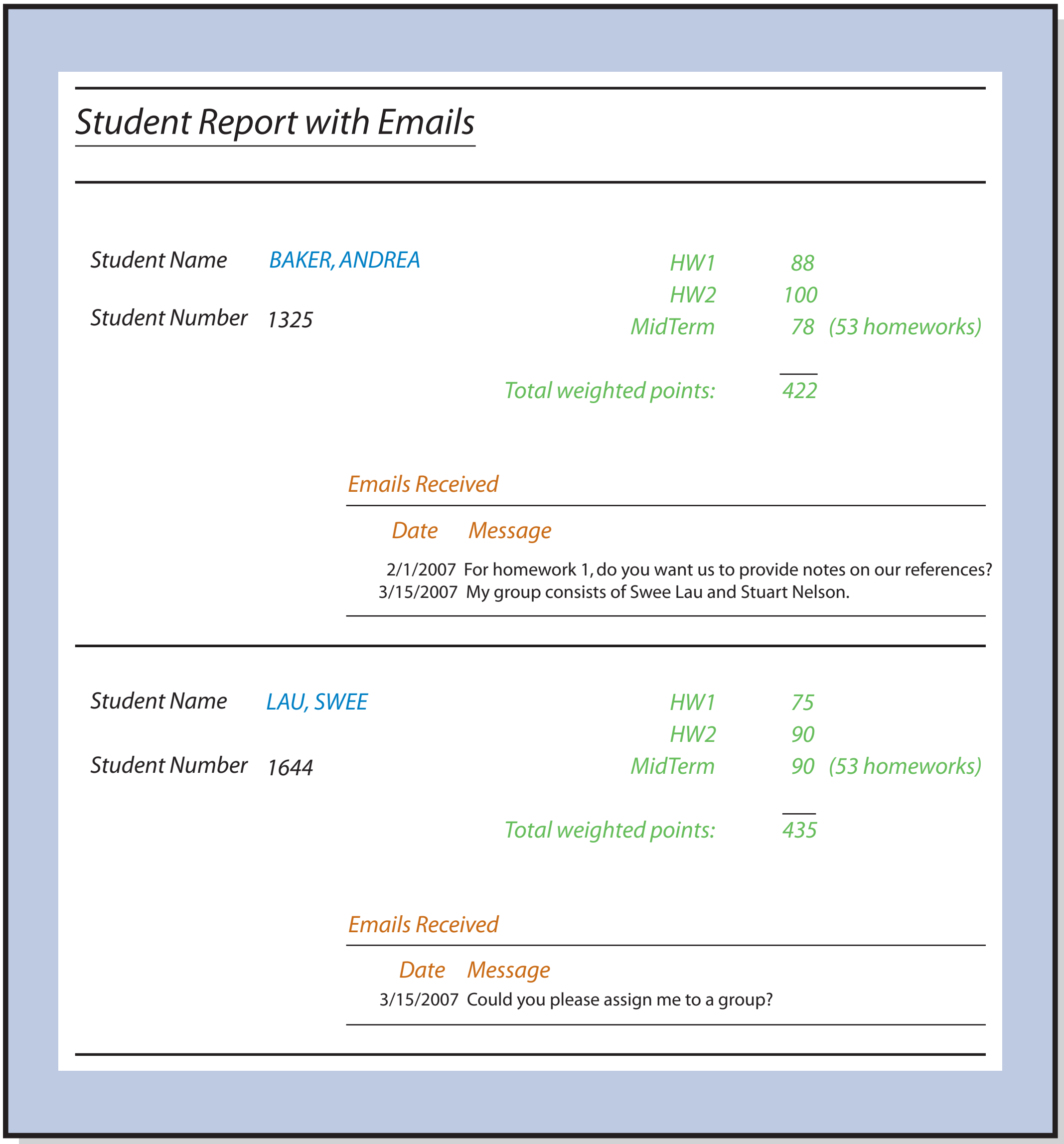
Questions

1. **How serious a problem is duplicate information to the financial services industry? Is it more serious for some industries than others? (*Hint:* How much of an issue is it for the health industry?)**
2. **Are there any other costs to Vancity when duplicate information is sent to customers? (*Hint:* What impression would you have if you received duplicate marketing information from various organizations?)**
3. **What are the various challenges in cleaning and grooming data? (*Hint:* Are there reasons why customers may have separate or changing information?)**
4. **Would the problem be solved by identifying customers numerically? How would customers perceive this? Are there legal issues?**

Forms, Reports, and Queries

Figure 5-2 shows a typical database application data entry **form**, and Figure 5-11 shows a typical **report**. Data entry forms are used to read, insert, modify, and delete data. Reports show data in a structured context.

Some reports, like the one in Figure 5-11, also compute values as they present the data. An example is the computation of *Total weighted points* in Figure 5-11.



Student Report with Emails

Student Name	*BAKER, ANDREA*	*HW1*	*88*	
		HW2	*100*	
Student Number	*1325*	*MidTerm*	*78*	*(53 homeworks)*
		Total weighted points:	*422*	

Emails Received

Date	*Message*
2/1/2007	For homework 1, do you want us to provide notes on our references?
3/15/2007	My group consists of Swee Lau and Stuart Nelson.

Student Name	*LAU, SWEE*	*HW1*	*75*	
		HW2	*90*	
Student Number	*1644*	*MidTerm*	*90*	*(53 homeworks)*
		Total weighted points:	*435*	

Emails Received

Date	*Message*
3/15/2007	Could you please assign me to a group?

Figure 5-11
Example of a Student Report

Recall from Chapter 2 that one of the definitions of information is "data presented in a meaningful context." The structure of this report creates information because it shows the student data in a context that will be meaningful to the professor.

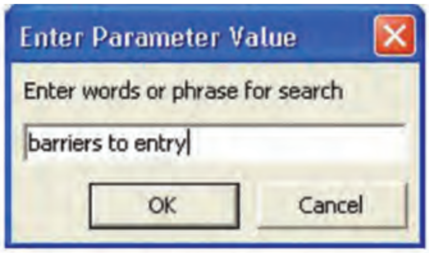
DBMS programs provide comprehensive and robust features for querying database data. For example, suppose the professor who uses the *Student* database remembers that one of the students referred to the topic *barriers to entry* in an office visit, but he or she cannot remember which student or when. If there are hundreds of students and visits recorded in the database, it will take some effort and time for the professor to search through all office visit records to find that event. The DBMS, however, can find any such record quickly. Figure 5-12(a) shows a **query** form in which the professor types in the keyword for which he or she is looking. Figure 5-12(b) shows the results of the query.

Database Application Programs

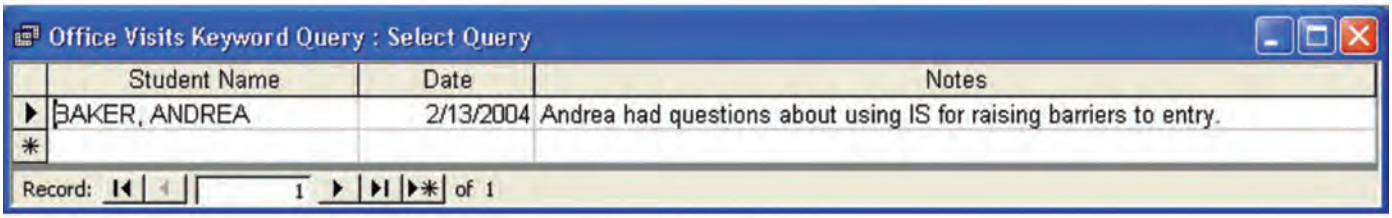
Forms, reports, and queries work well for standard functions. However, most applications have unique requirements that a simple form, report, or query cannot meet.

Figure 5-12
Example of a Query

Source: Microsoft Access 2010



a. Form used to enter phrase for search



b. Results of query operation

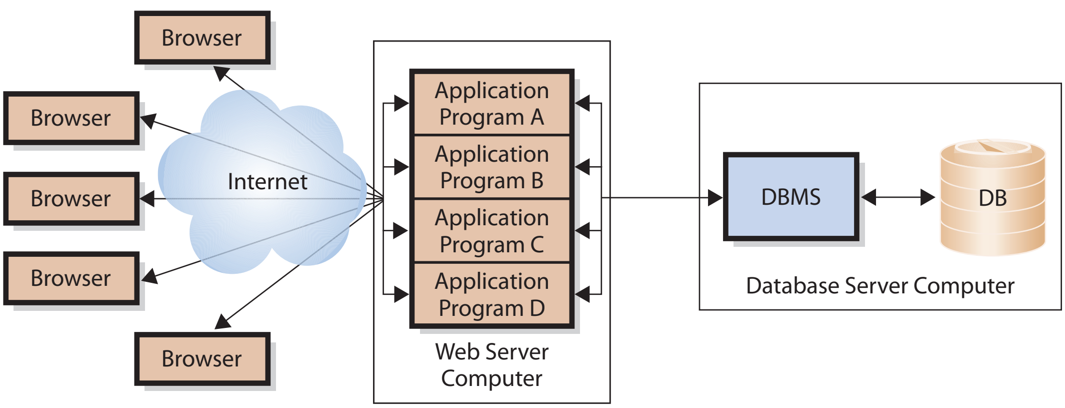
For example, in the order entry application in Figure 5-10, what should be done if only a portion of a customer's request can be met? If someone wants 10 widgets and there are only 3 in stock, should a backorder for 7 more be generated automatically? Or should some other action be taken?

Application programs process logic that is specific to a given business need. In the *Student* database, an example application is one that assigns grades at the end of the term. If the professor grades on a curve, the application reads the breakpoints for each grade from a form and then processes each row in the *Student* table, allocating a grade based on the breakpoints and the total number of points earned.

Another important use of application programs is to enable database processing over the internet. For this use, the application program serves as an intermediary between the web server and the database. The application program responds to events, such as when a user presses a submit button; it also reads, inserts, modifies, and deletes database data.

Figure 5-13 shows four different database application programs running on a web server computer. Users with browsers connect to the web server via the internet. The web server directs user requests to the appropriate application program. Each program then processes the database, as necessary.

Figure 5-13
Four Application Programs on a Web Server Computer



Multiuser Processing

Figures 5-10 and 5-13 show multiple users processing the database. Such **multiuser processing** is common, but it does pose unique problems that you, as a future manager, need to be aware of. To understand the nature of those problems, consider the following scenario:

Two users, Andrea and Jeffrey, are using the order entry application in Figure 5-10. Andrea is on the phone with her customer, who wants to purchase 5 widgets. At the same time, Jeffrey is talking with his customer, who wants to purchase 3 widgets. Andrea reads the database to determine how many widgets are in inventory. (She unknowingly invokes the order entry application when she types in her data entry form.) The DBMS returns a row showing 10 widgets in inventory.

Meanwhile, just after Andrea accesses the database, Jeffrey's customer says she wants widgets, and so he also reads the database (via the order entry application program) to determine how many widgets are in inventory. The DBMS returns the same row to him, indicating that 10 widgets are available.

Andrea's customer now says that he will take 5 widgets, and Andrea records this fact in her form. The application rewrites the widget row back to the database, indicating that there are 5 widgets in inventory.

Meanwhile, Jeffrey's customer says that he will take 3 widgets. Jeffrey records this fact in his form, and the application rewrites the widget row back to the database. However, Jeffrey's application knows nothing about Andrea's work and subtracts 3 from the original count of 10, thus storing an incorrect count of 7 widgets in inventory. Clearly, there is a problem. We began with 10 widgets, Andrea took 5 and Jeffrey took 3, but the database says there are 7 widgets in inventory. It should show 2, not 7.

This problem, known as the **lost-update problem**, exemplifies one of the special characteristics of multiuser database processing. To prevent this problem, some type of locking must be used to coordinate the activities of users who are unaware of each other. Locking brings its own complexity and problems that must be addressed, but these are beyond the scope of this chapter and text.

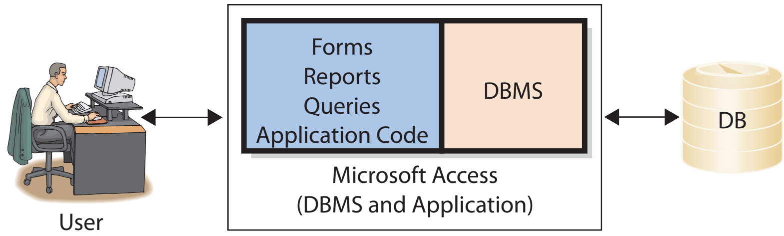
The purpose of this example is to illustrate that making a system usable by more than one person requires a lot more than simply enabling computer connections. The logic of the underlying application processing needs to be adjusted as well.

Be aware of possible data conflicts when you manage business activities that involve multiuser processing. If you find inaccurate results that seem not to have a cause, you may be experiencing multiuser data conflicts. Contact your MIS department for assistance.

Q7 What Is the Difference Between an Enterprise DBMS and a Personal DBMS?

DBMS products fall into two broad categories. **Enterprise DBMS** products process large organizational and workgroup databases. These products support many (perhaps thousands of) users and many different database applications. They also support 24/7 operations and can manage databases that span dozens of different magnetic disks with thousands of gigabytes or more of data. IBM's DB2, Microsoft's SQL Server, and Oracle's Oracle are examples of enterprise DBMS products.

Figure 5-14
Personal Database System



Personal DBMS products are designed for smaller, simpler database applications. Such products are used for personal or small workgroup applications that involve fewer than 100 users, and normally fewer than 15. In fact, the great bulk of databases in this category have only a single user. The professor's *Student* database is an example of a database that is processed by a personal DBMS product.

In the past, there were many personal DBMS products—Paradox, dBase, R:BASE, and FoxPro. Microsoft phased out these products when it developed Access and included it in the Microsoft Office suite. Today, the only remaining personal DBMS is Microsoft Access.

To avoid one point of confusion for you in the future, note that the separation of application programs and the DBMS shown in Figure 5-10 is true only for enterprise DBMS products. Microsoft Access includes features and functions for application processing along with the DBMS itself. For example, Access has a form generator and a report generator. Thus, as shown in Figure 5-14, Access is both a DBMS and an application development product.

Active Review

Use this Active Review to verify that you have understood the material in the chapter. You can read the entire chapter and then perform the tasks in this review, or you can read the material for just one question and perform the tasks for that question before moving on to the next one.

Q1 What is content?

Describe what is meant by the term *content*. How has content changed with computers and access to the internet?

Q2 How can content be organized?

What is a web content management system? How have content management systems evolved over time?

Q3 What is the purpose of a database?

Describe the purpose of a database. Explain when you should use a spreadsheet and when you should use a database.

Q4 What does a database contain?

Explain the hierarchy of data from bytes to tables. Show how a database stores the relationships among rows. Define *key* and *foreign key*. Define *metadata*, and explain how metadata makes databases more useful.

Q5 What is a DBMS, and what does it do?

Describe a database application system. Define *DBMS*. Name three prominent DBMS products. Describe the difference between a database and a DBMS. Explain the three major functions of a DBMS. What is SQL used for?

Q6 What is a database application?

Name and describe the components of a database application. Describe the circumstances that require a special logic for database applications. Describe the lost-update problem. Explain, in general terms, how this problem is prevented.

Q7 What is the difference between an enterprise DBMS and a personal DBMS?

Explain the function of an enterprise DBMS and describe its characteristics. Explain the function of a personal DBMS and describe its characteristics. Name the only surviving personal DBMS. Explain the differences between Figure 5-10 and Figure 5-14.

MyMISLab MyMISLab is an online learning and testing environment that features the perfect study tools to help you master the concepts covered in this chapter. Log in to MyMISLab to test your knowledge of key chapter concepts and explore additional practice tools, including videos, flashcards, annotated text figures, and more!

Key Terms and Concepts

Access 141
Byte 137
Columns 137
Content management systems (CMSs) 135
Database 137
Database application 143
Database application system 141
Database management system (DBMS) 141
DB2 141
Enterprise DBMS 147
Fields 137
File 137
Foreign keys 139
Form 144
Intellectual property 134
Key 138
Lost-update problem 147
Metadata 140
Multiuser processing 147
MySQL 141
Oracle 141
Personal DBMS 148
Query 145
Records 137
Relation 139
Relational databases 139
Report 144
Rows 137
SQL Server 141
Structured Query Language (SQL) 142
Table 137

Using Your Knowledge

1. Suppose you are a marketing assistant for a consumer electronics company and are in charge of setting up your company's booth at trade shows. Weeks before the shows, you meet with the marketing managers and determine what equipment they want to display. Then you identify each of the components that need to be shipped and schedule a shipper to deliver them to the trade-show site. You then supervise convention personnel as they set up the booths and equipment. Once the show is over, you supervise the packing of the booth and all equipment as well as schedule its shipment back to your home office. Once the equipment arrives, you check it in to your warehouse to ensure that all pieces of the booth and all equipment are returned. If there are problems because of shipping damage or loss, you handle those problems. Your job is important; at a typical show, you are responsible for more than $250 000 worth of equipment.
   a. You will need to track data about booth components, equipment, shippers, and shipments. List typical fields for each type of data.
   b. Could you use a spreadsheet to keep track of this data? What would be the advantages and disadvantages of doing so?
   c. Using your answer to question (a), give an example of two relationships that you need to track. Show the keys and foreign keys for each.
   d. Which of the following components of a database application are you likely to need: data entry forms, reports, queries, or application programs? Explain one use for each component you will need.
   e. Will your application be single-user or multiuser? Will you need a personal DBMS or an enterprise DBMS? If you need a personal DBMS, which product will you use?

2. Samantha Green owns and operates Twigs Tree Trimming Service. Recall from Chapter 3 that Samantha has a degree from a forestry program and recently opened her business in Winnipeg. Her business consists of many one-time operations (e.g., removing a tree or stump), as well as recurring services (e.g., trimming customers' trees every year or two). When business is slow, Samantha calls former clients to remind them of her services and of the need to trim their trees on a regular basis.
    a. Name and describe the tables of data that Samantha will need in order to run her business. Indicate possible fields for each table.
    b. Could Samantha use a spreadsheet to keep track of these data? What would be the advantages and disadvantages of doing so?
    c. Using your answer to question (a), give an example of two relationships that Samantha needs to track. Show the keys and foreign keys for each.
    d. Which of the following components of a database application is Samantha likely to need: data entry forms, reports, queries, or application programs? Explain one use for each component she needs.
    e. Will this application be single-user or multiuser? Will she need a personal DBMS or an enterprise DBMS? If she needs a personal DBMS, which product will she use?

3. FiredUp Inc. is a small business owned by Curt and Julie Robards. Based in Brisbane, Australia, FiredUp manufactures and sells FiredNow, a lightweight camping stove. Recall from Chapter 3 that Curt used his previous experience as an aerospace engineer to invent a burning nozzle that enables the stove to stay lit in very high winds. Using her industrial-design training, Julie designed the stove so that it is small, lightweight, easy to set up, and very stable. Curt and Julie sell the stove directly to their customers over the internet and via the phone. The warranty on the stove covers five years of cost-free repair for stoves that are used for recreational purposes.

    FiredUp wants to track every stove and the customer who purchased it. They want to know which customers own which stoves, in case they need to notify customers of safety problems or need to order a stove recall. Curt and Julie also want to keep track of any repairs they have performed.
    a. Name and describe tables of data that FiredUp will need. Indicate possible fields for each table.
    b. Could FiredUp use a spreadsheet to keep track of data? What would be the advantages and disadvantages of doing so?
    c. Using your answer to (a), give an example of two relationships FiredUp needs to track. Show the keys and foreign keys for each.
    d. Which of the following components of a database application is FiredUp likely to need: data entry forms, reports, queries, application programs? Explain one use for each needed component.
    e. Will this application be single-user or multiuser? Will FiredUp need a personal DBMS or an enterprise DBMS? If they need a personal DBMS, which product will it use? If they need an enterprise DBMS, which product can they obtain licence-free?

Collaborative Exercises

Collaborate with a group of students on the following exercises.

Figure 5-15 shows a spreadsheet that is used to track the assignment of sheet music to a choir—it could be a church choir, or school or community choir. The type

	A	B	C	D	E
1	Last Name	First Name	Email	Phone	Part
2	Ashley	Jane	JA@somewhere.com	703.555.1234	Soprano
3	Davidson	Kaye	KD@somewhere.com	703.555.2236	Soprano
4	Ching	Kam Hoong	KHC@overhere.com	703.555.2236	Soprano
5	Menstell	Lori Lee	LLM@somewhere.com	703.555.1237	Soprano
6	Corning	Sandra	SC2@overhere.com	703.555.1234	Soprano
7		B-minor mass	J.S. Bach	Soprano Copy 7	
8		Requiem	Mozart	Soprano Copy 17	
9		9th Symphony Chorus	Beethoven	Soprano Copy 9	
10	Wei	Guang	GW1@somewhere.com	703.555.9936	Soprano
11	Dixon	Eleanor	ED@thisplace.com	703.555.12379	Soprano
12		B-minor mass	J.S. Bach	Soprano Copy 11	
13	Duong	Linda	LD2@overhere.com	703.555.8736	Soprano
14		B-minor mass	J.S. Bach	Soprano Copy 7	
15		Requiem	J.S. Bach	Soprano Copy 19	
16	Lunden	Haley	HL@somewhere.com	703.555.0836	Soprano
17	Utran	Diem Thi	DTU@somewhere.com	703.555.1089	Soprano

Figure 5-15

Spreadsheet Used for Assignment of Sheet Music

Source: Microsoft Excel

of choir does not matter because the problem is a universal one. Sheet music is expensive, choir members need to be able to take sheet music away for practise at home, and not all of the music gets back to the inventory. (Sheet music can be purchased or rented, but either way, lost music is an expense.)

Look closely at these data, and you will see some data integrity problems—or at least some possible data integrity problems. For one, do Sandra Corning and Linda Duong really have the same copy of music checked out? Second, did Mozart and J. S. Bach both write a Requiem, or in row 15 should J. S. Bach actually be Mozart? Also, there is a problem with Eleanor Dixon's phone number and several phone numbers are the same, which seems suspicious.

Additionally, this spreadsheet is confusing and hard to use. The column labelled *First Name* includes both people names and the names of choruses. *Email* has both email addresses and composer names, and *Phone* has both phone numbers and copy identifiers. Furthermore, to record a checkout of music the user must first add a new row and then re-enter the name of the work, the composer's name, and the copy to be checked out. Finally, consider what happens when the user wants to find all copies of a particular work: The user will have to examine the rows in each of four spreadsheets for the four voice parts.

In fact, a spreadsheet is ill-suited for this application. A database would be a far better tool, and such situations are obvious candidates for innovation.

1. Analyze the spreadsheet shown in Figure 5-15 and list all of the problems that occur when trying to track the assignment of sheet music using this spreadsheet.

2. The following two tables could be used to store the data in Figure 5-15 in a database:

   ChoirMember (*LastName, FirstName, Email, Phone, Part*)

   MusicalWork *(NameOfWork, Composer, Part, CopyNumber)*

   Note: This notation means there are two tables, one named *ChoirMember* and a second named *MusicalWork*. The *ChoirMember* table has five columns: *LastName, FirstName, Email, Phone*, and *Part*; *MusicalWork* has four columns: *NameOfWork, Composer, Part, CopyNumber.*

   a. Redraw the data in Figure 5-15 into this two-table format.
   b. Select primary keys for the *ChoirMember* and *MusicalWork* tables.

c. The two tables are not integrated; they do not show who has checked out which music. Add foreign key columns to one of the tables to integrate the data.
d. This two-table design does not eliminate the potential for data integrity problems that occur in the spreadsheet. Explain why not.

3. A three-table database design for the data in the spreadsheet in Figure 5-15 is as follows:

   ChoirMember (*LastName, FirstName, Email, Phone, Part*)

   MusicalWork (*NameOfWork*)

   CheckOut (*LastName, FirstName, NameOfWork, Part, CopyNumber, DateOut, DateIn*)

   a. Redraw the data in Figure 5-15 into this three-table format.
   b. Identify which columns are primary keys for each of these tables.
   c. The foreign keys are already in place; identify which columns are foreign keys and which relationships they represent.
   d. Does this design eliminate the potential for data integrity problems that occur in the spreadsheet? Why or why not?

4. Assume that you manage the choir and you foresee two possibilities:

   - Keep the spreadsheet, but create procedures to reduce the likelihood of data integrity problems.
   - Create an Access database and database application for the three-table design.

   Describe the advantages and disadvantages of each of these possibilities. Recommend one of these two possibilities and justify your recommendation.

Case Study 5

Behind the Race

For hundreds of thousands of athletes, competing in a marathon or triathlon is the culmination of years of dedicated training and endurance. Although largely invisible, a similar regimen can exist for the marathon's organizers, who must manage planning, donations, fundraising, creating custom websites, marketing, volunteer management, online registration, newsletters, and the publication of results. Although not directly related to the event itself, these activities can be individually and collectively overwhelming.

The Active Network was originally formed by a small community of endurance race directors, league administrators, and enthusiastic volunteers. They realized that each time one of the hundreds of autonomous athletic organizations planned an event, they were in essence reinventing the wheel. So, the Active Network (www.active.com) was formed to harness the power of online technology and marketing solutions.

Planning an event has now become vastly simplified. The Active Network provides tailored solutions for better experience and exposure for participants. Rather than one-off or ad hoc solutions cobbled together by part-time administrators—who, in many cases, were doing things for the first time—all organizations, regardless of their size, now have access to state-of-the-art systems.

The Active Network was a solution that had legs. By focusing on three areas—improving the site for the benefit of their technology customers, enhancing the athlete's overall experience, and providing heightened exposure for their clients—the Active Network now provides online technology and marketing solutions to thousands of business owners and organizers in more than 80 sports and recreational markets that serve millions of participants worldwide.

Building on its vision to enable and encourage every individual to learn about, share, register, and ultimately participate in any activity, Active.com has become the largest searchable worldwide directory of sports and recreational activities. By providing access to such tools as community message boards and specialized moderators and allowing members to post photographs and videos, Active.com has become a destination site for athletes and a trusted community among its members.

Questions

1. What problem does Active.com solve for event organizers?
2. Does Active.com have any other advantages other than economies of scale?
3. Are there are any network effects for Active.com? (*Hint:* What are the benefits to having a large number of events in one place?)
4. What kind of information does Active.com have about its members? How would this be useful and to whom? What is the value of this information?

Nobody Said I Shouldn't

"My name is Kelly, and I do systems support for our group. I configure the new computers, set up the network, make sure the servers are operating, and so forth. I also do all the database backups. I've always liked computers. After high school, I worked odd jobs to make some money, then I got an associate degree in information technology from our local community college.

"Anyway, as I said, I make backup copies of our databases. One weekend, I didn't have much going on, so I copied one of the database backups to a CD and took it home. I had taken a class on database processing as part of my associate degree, and we used an SQL Server (our database management system) in my class. In fact, I suppose that's part of the reason I got the job. Anyway, it was easy to restore the database on my computer at home, and I did.

"Of course, as they'll tell you in your database class, one of the big advantages of database processing is that databases have metadata, or data that describe the content of the database. So, although I didn't know what tables were in our database, I did know how to access the SQL server metadata. I just queried a table called sysTables to learn the names of our tables. From there it was easy to find out what columns each table had.

"I found tables with data about orders, customers, salespeople, and so forth, and just to amuse myself and to see how much of the query language SQL I could remember, I started playing around with the data. I was curious to know which order-entry clerk was the best, so I started querying each clerk's order data, the total number of orders, total order amounts, things like that. It was easy to do and fun.

"I know one of the order-entry clerks, Jason, pretty well, so I started looking at the data for his orders. I was just curious, and it was very simple SQL. I was just playing around with the data when I noticed something odd. All his biggest orders were with one company, Valley Appliances; even stranger, every one of its orders had a huge discount. I thought, *Well, maybe that's typical*. Out of curiosity, I started looking at data for the other clerks, and very few of them had an order with Valley Appliances. But when they did, Valley didn't get a big discount. Then I looked at the rest of Jason's orders, and none of them had much in the way of discounts, either.

"The next Friday, a bunch of us went out for beer after work. I happened to see Jason, so I asked him about Valley Appliances and made a joke about the discounts. He asked me what I meant, and then I told him that I'd been looking at the data for fun and that I saw this odd pattern. He laughed, said he 'just did his job,' and then changed the subject.

"Well, to make a long story short, when I got to work on Monday morning, my office was cleaned out. There was nothing there except a note telling me to go see my boss. The bottom line was, I was fired. The company also threatened that if I didn't return all of its data, I'd be in court for the next five years. .. things like that. I was so mad I didn't even tell them about Jason. Now my problem is that I'm out of a job, and I can't exactly use my last company for a reference."

Discussion Questions

1. Where did Kelly go wrong?
2. Do you think it was illegal, unethical, or neither for Kelly to take the database home and query the data?
3. Does the company share culpability with Kelly?
4. What do you think Kelly should have done upon discovering the odd pattern in Jason's orders?
5. What should the company have done before firing Kelly?
6. "Metadata make databases easy to use—for both authorized and unauthorized purposes." Explain what organizations should do in light of this fact.

wavebreakmedia/Shutterstock

Chapter Extension 5a

Chapter 5 provides the background for this extension.

Database Design

In this chapter extension, you will learn about data modelling and how data models are transformed into database designs. You'll also learn the important role that business professionals have in the development of a database application system.

Study Questions

Q1 Who will volunteer?

Q2 How are database application systems developed?

Q3 What are the components of the entity-relationship data model?

Q4 How is a data model transformed into a database design?

Q5 What is the users' role?

Q6 Who will volunteer? (Continued)

MyMISLab

Visit MyMISLab for simulations, tutorials, and end-of-chapter problems.

Q1 Who Will Volunteer?

Suppose you are the manager of fund-raising for a local public television station. Twice a year you conduct fund drives during which the station runs commercials that ask viewers to donate. These drives are important; they provide nearly 40 percent of the station's operating budget.

One of your job functions is to find volunteers to staff the phones during these drives. You need 10 volunteers per night for six nights, or 60 people, twice per year. The volunteers' job is exhausting, and normally a volunteer will work only one night during a drive.

Finding volunteers for each drive is a perpetual headache. Two months before a drive begins, you and your staff start calling potential volunteers. You first call volunteers from prior drives, using a roster that your administrative assistant prepares for each drive. Some volunteers have been helping for years; you'd like to know that information before you call them so that you can tell them how much you appreciate their continuing support. Unfortunately, the roster does not have that data.

Additionally, some volunteers are more effective than others. Some have a particular knack for increasing the callers' donations. Although those data are available, the information is not in a format that you can use when calling for volunteers. You think you could better staff the fund-raising drives if you had that missing information.

You know that you can use a computer database to keep better track of prior volunteers' service and performance, but you are not sure how to proceed. By the end of this chapter extension, when we return to this fund-raising situation, you will know what to do.

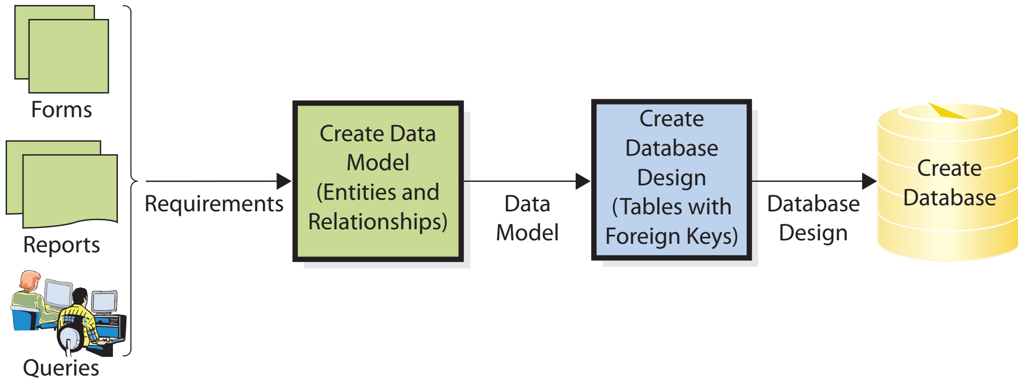
Q2 How Are Database Application Systems Developed?

You learned in Chapter 5 that a database application system consists of a database, a DBMS, and one or more database applications. A database application, in turn, consists of forms, reports, queries, and possibly application programs. The question then becomes: How are such systems developed? And, even more important to you, what is the users' role? We will address these questions in this chapter extension.

Figure AE5a-1 summarizes the database application system development process. First, the developers interview users and develop the requirements for the new system. During this process, the developers analyze existing forms, reports, queries,

Figure AE5a-1

Database Development Process



and other user activities. They also determine the need for new forms, reports, and queries. The requirements for the database are then summarized in something called a **data model**, which is a logical representation of the structure of the data. The data model contains a description of both the data and the relationships among the data. It is akin to a blueprint. Just as building architects create a blueprint before they start construction, so, too, database developers create a data model before they start designing the database.

Once the users have validated and approved the data model, it is transformed into a database design. After that, the design is implemented in a database, and that database is then filled with user data.

You will learn much more about systems development in Chapter 10 and its related extensions. We discuss data modelling here because users have a crucial role in the success of any database development: They must validate and approve the data model. Only the users know what should be in the database.

Consider, for example, a database of students that an adviser uses for his or her advisees. What should be in it? Students? Classes? Records of emails from students? Records of meetings with students? Majors? Student organizations? Even when we know what themes should be in the database, we must ask, how detailed should the records be? Should the database include campus addresses? Home addresses? Billing addresses?

In fact, there are many possibilities, and the database developers do not and cannot know what to include. They do know, however, that a database must include all the data necessary for the users to perform their jobs. Ideally, it contains that amount of data and no more. So during database development, the developers must rely on the users to tell them what they need in the database. They will rely on the users to check the data model and to verify it for correctness, completeness, and appropriate level of detail. That verification will be your job. We begin with a discussion of the entity-relationship data model—the most common tool to use to construct data models.

Q3 What Are the Components of the Entity-Relationship Data Model?

The most popular technique for creating a data model is the **entity-relationship (E-R) data model**. With it, developers describe the content of a database by defining the things (*entities*) that will be stored in the database and the *relationships* among those entities. A second, less popular tool for data modelling is the **Unified Modelling Language (UML)**. We will not describe that tool here. However, if you learn how to interpret E-R models, with a bit of study you will be able to understand UML models as well.

Entities

An **entity** is something that the users want to track. Examples of entities are *Order, Customer, Salesperson*, and *Item*. Some entities represent a physical object, such as *Item* or *Salesperson*; others represent a logical construct or transaction, such as *Order* or *Contract.* For reasons beyond this discussion, entity names are always singular. We use *Order*, not *Orders*; *Salesperson*, not *Salespersons.*

Entities have **attributes** that describe characteristics of the entity. Example attributes of *Order* are *OrderNumber, OrderDate, SubTotal, Tax, Total*, and so forth. Example attributes of *Salesperson* are *SalespersonName, Email, Phone*, and so forth.

Entities have an **identifier**, which is an attribute (or group of attributes) whose value is associated with one and only one entity instance. For example, *OrderNumber* is an identifier of *Order* because only one *Order* instance has a given value of *OrderNumber.* For the same reason, *CustomerNumber* is an identifier of *Customer.* If each member of the sales staff has a unique name, then *SalespersonName* is an identifier of *Salesperson.*

Before we continue, consider that last sentence. Is the salesperson's name unique among the sales staff? Both now and in the future? Who decides the answer to such a question? Only the users know whether this is true; the database developers cannot know. This example underlines why it is important for you to be able to interpret data models because only users like yourself will know for sure.

Figure AE5a-2 shows examples of entities for the Student database. Each entity is shown in a rectangle. The name of the entity is just above the rectangle, and the identifier is shown in a section at the top of the entity. Entity attributes are shown in the remainder of the rectangle. In Figure AE5a-2, the *Adviser* entity has an identifier called *AdviserName* and the attributes *Phone, CampusAddress*, and *EmailAddress.*

Observe that the entities *Email* and *Office_Visit* do not have an identifier. Unlike *Student* or *Adviser*, the users do not have an attribute that identifies a particular email. In fact, *Email* and *Office_Visit* are identified, in part, by their relationship to Student. For now, we need not worry about that. The data model needs only to show how users view their world. When it comes to database design, the designer will deal with the missing identifiers by adding columns, possibly using hidden identifiers, to implement the users' view. You can learn about the modelling and representation of such entities if you enroll in a database class.

Relationships

Entities have **relationships** to each other. An *Order*, for example, has a relationship to a *Customer* entity and also to a *Salesperson* entity. In the Student database, a *Student* has a relationship to an *Adviser*, and an *Adviser* has a relationship to a *Department.*

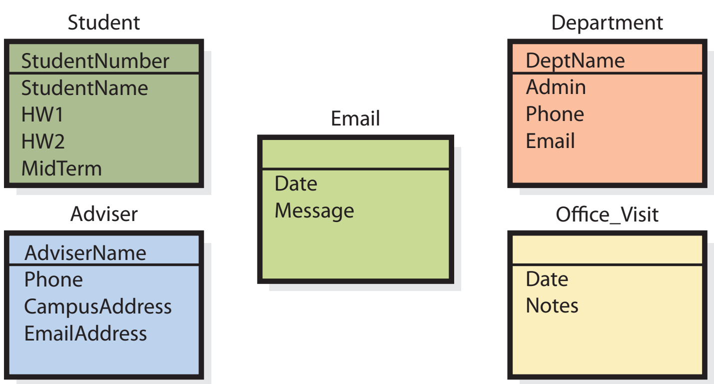


Figure AE5a-2
Student Data Model Entities

Figure AE5a-3

Example of *Department, Adviser*, and *Student* Entities and Relationships

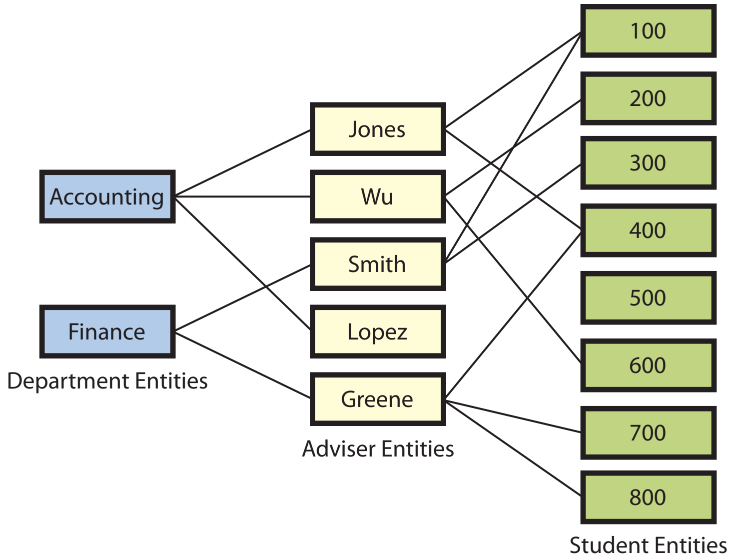


Figure AE5a-3 shows sample *Department, Adviser*, and *Student* entities and their relationships. For simplicity, this figure shows just the identifier of the entities and not the other attributes. For this sample data, *Accounting* has three professors, Jones, Wu, and Lopez, and *Finance* has two professors, Smith and Greene.

The relationship between *Advisers* and *Students* is a bit more complicated because, in this example, an adviser is allowed to advise many students and a student is allowed to have many advisers. Perhaps this happens because students can have multiple majors. In any case, note that Professor Jones advises students 100 and 400 and that student 100 is advised by both Professors Jones and Smith.

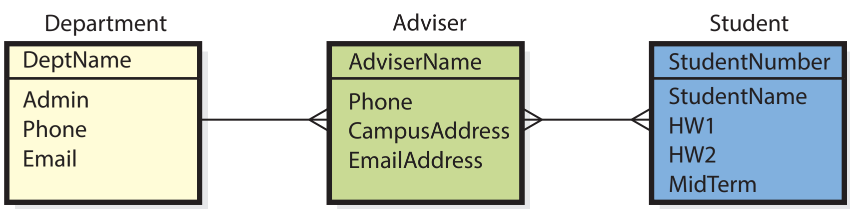
Diagrams like the one in Figure AE5a-3 are too cumbersome for use in database design discussions. Instead, database designers use diagrams called **entity-relationship (E-R) diagrams**. Figure AE5a-4 shows an E-R diagram for the data in Figure AE5a-3. In this figure, all of the entities of one type are represented by a single rectangle. Thus, there are rectangles for the *Department, Adviser*, and *Student* entities. Attributes are shown as before in Figure AE5a-2.

Additionally, a line is used to represent a relationship between two entities. Notice the line between *Department* and *Adviser*, for example. The forked lines on the right side of that line signify that a department may have more than one adviser. The little lines, which are referred to as a **crow's foot**, are shorthand for the multiple lines between *Department* and *Adviser* in Figure AE5a-3. Relationships like this one are called **one-to-many (1:N) relationships** because one department can have many advisers.

Now examine the line between *Adviser* and *Student*. Here a crow's foot appears at each end of the line. This notation signifies that an adviser can be related to many students and that a student can be related to many advisers, which is the situation in Figure AE5a-3. Relationships like this one are called **many-to-many (N:M)**

Figure AE5a-4

Example Relationships—Version 1



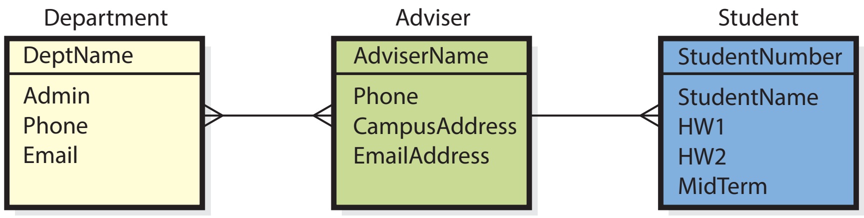


Figure AE5a-5
Example Relationships—Version 2

relationships because one adviser can have many students and one student can have many advisers.

Students sometimes find the notation N:M confusing. Interpret the *N* and *M* to mean that a variable number, greater than one, is allowed on each side of the relationship. Such a relationship is not written *N:N* because that notation would imply that there are the same number of entities on each side of the relationship, which is not necessarily true. *N:M* means that more than one entity is allowed on each side of the relationship and that the number of entities on each side can be different.

Figure AE5a-4 is an example of an entity-relationship diagram. Unfortunately, there are several different styles of entity-relationship diagrams. This one is called, not surprisingly, a **crow's-foot diagram** version. You may learn other versions if you take a database management class.

Figure AE5a-5 shows the same entities with different assumptions. Here advisers may advise in more than one department, but a student may have only one adviser, representing a policy that students may not have multiple majors.

Which, if either of these versions—Figure AE5a-4 or Figure AE5a-5—is correct? Only the users know. These alternatives illustrate the kinds of questions you will need to answer when a database designer asks you to check a data model for correctness.

The crow's-foot notation shows the maximum number of entities that can be involved in a relationship. Accordingly, they are called the relationship's **maximum cardinality**. Common examples of maximum cardinality are 1:N, N:M, and 1:1 (not shown).

Another important question is, "What is the minimum number of entities required in the relationship?" Must an adviser have a student to advise, and must a student have an adviser? Constraints on minimum requirements are called **minimum cardinalities**.

Figure AE5a-6 presents a third version of this E-R diagram that shows both maximum and minimum cardinalities. The vertical bar on a line means that at least one entity of that type is required. The small oval means that the entity is optional; the relationship need not have an entity of that type.

Thus, in Figure AE5a-6, a department is not required to have a relationship to any adviser, but an adviser is required to belong to a department. Similarly, an adviser is not required to have a relationship to a student, but a student is required to have a relationship to an adviser. Note, also, that the maximum cardinalities in Figure AE5a-6 have been changed so that both are 1:N.

Is the model in Figure AE5a-6 a good one? It depends on the rules of the university. Again, only the users know for sure.

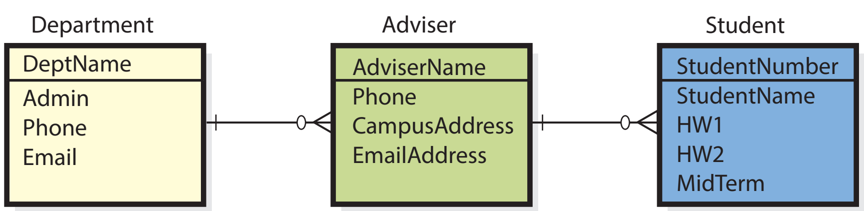


Figure AE5a-6
Example of Relationships Showing Both Maximum and Minimum Cardinalities

Q4 How Is a Data Model Transformed into a Database Design?

Database design is the process of converting a data model into tables, relationships, and data constraints. The database design team transforms entities into tables and expresses relationships by defining foreign keys. Database design is a complicated subject; as with data modelling, it occupies weeks in a database management class. In this section, however, we will introduce two important database design concepts: normalization and the representation of two kinds of relationships. The first concept is a foundation of database design, and the second will help you understand key considerations made during design.

Normalization

Normalization is the process of converting poorly structured tables into two or more well-structured tables. A table is such a simple construct that you may wonder how one could possibly be poorly structured. In truth, there are many ways that tables can be malformed—so many, in fact, that researchers have published hundreds of papers on this topic alone.

Consider the *Employee* table in Figure AE5a-7. It lists employee names, hire dates, email addresses, and the name and number of the department in which the employee works. This table seems innocent enough. But consider what happens when the Accounting department changes its name to Accounting and Finance. Because department names are duplicated in this table, every row that has a value of "Accounting" must be changed to "Accounting and Finance."

Data Integrity Problems

Suppose the Accounting name change is correctly made in two rows, but not in the third. The result is shown in Figure AE5a-7b. This table has what is called a **data integrity problem**: Two rows indicate that the name of Department 100 is Accounting and Finance, and another row indicates that the name of Department 100 is Accounting.

Figure AE5a-7
A Poorly Designed *Employee* Table

Employee

Name	HireDate	Email	DeptNo	DeptName
Jones	Feb 1, 2010	Jones@ourcompany.com	100	Accounting
Smith	Dec 3, 2007	Smith@ourcompany.com	200	Marketing
Chau	March 7, 2007	Chau@ourcompany.com	100	Accounting
Greene	July 17, 2010	Greene@ourcompany.com	100	Accounting

a. Table Before Update

Employee

Name	HireDate	Email	DeptNo	DeptName
Jones	Feb 1, 2010	Jones@ourcompany.com	100	Accounting and Finance
Smith	Dec 3, 2007	Smith@ourcompany.com	200	Marketing
Chau	March 7, 2007	Chau@ourcompany.com	100	Accounting and Finance
Greene	July 17, 2010	Greene@ourcompany.com	100	Accounting

b. Table with Incomplete Update

This problem is easy to spot in this small table. But consider a table in a large database that has more than 300 000 rows. Once a table that large develops serious data integrity problems, months of labour will be required to remove them.

Data integrity problems are serious. A table that has data integrity problems will produce incorrect and inconsistent information. Users will lose confidence in the information, and the system will develop a poor reputation. Information systems with poor reputations become heavy burdens to the organizations that use them.

Normalizing for Data Integrity

The data integrity problem can occur only if data are duplicated. Because of this, one easy way to eliminate the problem is to eliminate the duplicated data. We can do this by transforming the table design in Figure AE5a-7a into two tables, as shown in Figure AE5a-8. Here the name of the department is stored just once; therefore, no data inconsistencies can occur.

Of course, to produce an employee report that includes the department name, the two tables in Figure AE5a-8 will need to be joined back together. Because such joining of tables is common, DBMS products have been programmed to perform it efficiently, but it still requires work. From this example, you can see a trade-off in database design: Normalized tables eliminate data duplication, but they can be slower to process. Dealing with such trade-offs is an important consideration in database design.

The general goal of normalization is to construct tables such that every table has a *single* topic or theme. In good writing, every paragraph should have a single theme. This is true of databases as well; every table should have a single theme. The problem with the table design in Figure AE5a-7 is that it has two independent themes: employees and departments. The way to correct the problem is to split the table into two tables, each with its own theme. In this case, we create an *Employee* table and a *Department* table, as shown in Figure AE5a-8.

As mentioned, there are dozens of ways that tables can be poorly formed. Database practitioners classify tables into various **normal forms** according to the kinds of problems they have. Transforming a table into a normal form to remove duplicated data and other problems is called *normalizing* the table.[1] Thus, when you hear a database designer say, "Those tables are not normalized," she does not mean that the tables have irregular, not-normal data. Instead, she means that the tables have a format that could cause data integrity problems.

Employee

Name	HireDate	Email	DeptNo
Jones	Feb 1, 2010	Jones@ourcompany.com	100
Smith	Dec 3, 2011	Smith@ourcompany.com	200
Chau	March 7, 2007	Chau@ourcompany.com	100
Greene	July 17, 2010	Greene@ourcompany.com	100

Department

DeptNo	DeptName
100	Accounting
200	Marketing
300	Information Systems

Figure AE5a-8
Two Normalized Tables

[1] See David Kroenke and David Auer, *Database Concepts*, 6th ed. (Upper Saddle River, NJ: Pearson Education, 2013) for more information.

Summary of Normalization

As a future user of databases, you do not need to know the details of normalization. Instead, understand the general principle that every normalized (well-formed) table has one and only one theme. Further, tables that are not normalized are subject to data integrity problems.

Be aware, too, that normalization is just one criterion for evaluating database designs. Because normalized designs can be slower to process, database designers sometimes choose to accept non-normalized tables. The best design depends on the users' requirements.

Representing Relationships

Figure AE5a-9 shows the steps involved in transforming a data model into a relational database design. First, the database designer creates a table for each entity. The identifier of the entity becomes the key of the table. Each attribute of the entity becomes a column of the table. Next, the resulting tables are normalized so that each table has a single theme. Once that has been done, the next step is to represent the relationship among those tables.

For example, consider the E-R diagram in Figure AE5a-10a. The *Adviser* entity has a 1:N relationship to the *Student* entity. To create the database design, we construct a table for *Adviser* and a second table for *Student*, as shown in Figure AE5a-10b. The key of the *Adviser* table is *AdviserName*, and the key of the *Student* table is *StudentNumber*.

Further, the *EmailAddress* attribute of the *Adviser* entity becomes the *EmailAddress* column of the *Adviser* table, and the *StudentName* and *MidTerm* attributes of the *Student* entity become the *StudentName* and *MidTerm* columns of the *Student* table.

The next task is to represent the relationship. Because we are using the relational model, we know that we must add a foreign key to one of the two tables. The possibilities are: (1) place the foreign key *StudentNumber* in the *Adviser* table or (2) place the foreign key *AdviserName* in the *Student* table.

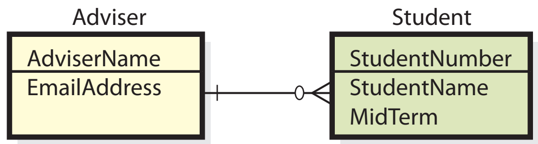
The correct choice is to place *AdviserName* in the *Student* table, as shown in Figure AE5a-10c. To determine a student's adviser, we just look into the *AdviserName* column of that student's row. To determine the adviser's students, we search the *AdviserName* column in the *Student* table to determine which rows have that adviser's name. If a student changes advisers, we simply change the value in the *AdviserName* column. Changing *Jackson* to *Jones* in the first row, for example, will assign student 100 to Professor Jones.

For this data model, placing *StudentNumber* in *Adviser* would be incorrect. If we were to do that, we could assign only one student to an adviser. There is no place to assign a second adviser.

This strategy for placing foreign keys will not work for N:M relationships, however. Consider the data model in Figure AE5a-11a (page 166); here there is an N:M relationship between advisers and students. An adviser may have many students,

Figure AE5a-9
Transforming a Data Model into a Database Design

- Represent each entity with a table
  - Entity identifier becomes table key
  - Entity attributes become table columns
- Normalize tables as necessary
- Represent relationships
  - Use foreign keys
  - Add additional tables for N:M relationships



a. 1: N Relationship Between Adviser and Student Entities

Adviser Table—Key is AdviserName

AdviserName	EmailAddress
Jones	Jones@myuniv.edu
Choi	Choi@myuniv.edu
Jackson	Jackson@myuniv.edu

Student Table—Key is StudentNumber

StudentNumber	StudentName	MidTerm
100	Lisa	90
200	Jennie	85
300	Jason	82
400	Terry	95

b. Creating a Table for Each Entity

Adviser Table—Key is AdviserName

AdviserName	EmailAddress
Jones	Jones@myuniv.edu
Choi	Choi@myuniv.edu
Jackson	Jackson@myuniv.edu

Foreign Key Column Represents Relationship

Student—Key is StudentNumber

StudentNumber	StudentName	MidTerm	AdviserName
100	Lisa	90	Jackson
200	Jennie	85	Jackson
300	Jason	82	Choi
400	Terry	95	Jackson

c. Using the AdviserName Foreign Key to Represent the 1:N Relationship

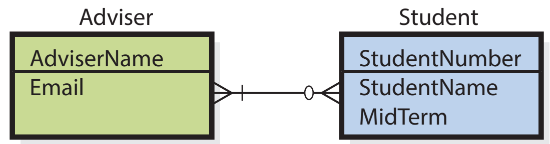
Figure AE5a-10
Representing a 1:N Relationship

and a student may have multiple advisers (for multiple majors). The strategy we used for the 1:N data model will not work here. To see why, examine Figure AE5a-11b. If student 100 has more than one adviser, there is no place to record second or subsequent advisers.

It turns out that to represent an N:M relationship, we need to create a third table, as shown in Figure AE5a-11c. The third table has two columns, *AdviserName* and *StudentNumber*. Each row of the table means that the given adviser advises the student with the given number.

As you can imagine, there is a great deal more to database design than we have presented here. Still, this section should give you an idea of the tasks that need to be accomplished to create a database. You should also realize that the database design is a direct consequence of decisions made in the data model. If the data model is wrong, the database design will be wrong as well.

Figure AE5a-11
Representing an N:M Relationship



a. N:M Relationship Between Adviser and Student

Adviser—Key is AdviserName

AdviserName	Email
Jones	Jones@myuniv.edu
Choi	Choi@myuniv.edu
Jackson	Jackson@myuniv.edu

No room to place second or third AdviserName

Student—Key is StudentNumber

StudentNumber	StudentName	MidTerm	AdviserName
100	Lisa	90	Jackson
200	Jennie	85	Jackson
300	Jason	82	Choi
400	Terry	95	Jackson

b. Incorrect Representation of N:M Relationship

Adviser—Key is AdviserName

AdviserName	Email
Jones	Jones@myuniv.edu
Choi	Choi@myuniv.edu
Jackson	Jackson@myuniv.edu

Student—Key is StudentNumber

StudentNumber	StudentName	MidTerm
100	Lisa	90
200	Jennie	85
300	Jason	82
400	Terry	95

Adviser_Student_Intersection

AdviserName	StudentNumber
Jackson	100
Jackson	200
Choi	300
Jackson	400
Choi	100
Jones	100

Student 100 has three advisers.

c. Adviser_Student_Intersection Table Represents the N:M Relationship

Q5 What Is the Users' Role?

As stated, a database is a model of how the users view their business world. This means that the users are the final judges of what data the database should contain and how the records in that database should be related to one another.

The easiest time to change the database structure is during the data modelling stage. Changing a relationship from 1:N to N:M in a data model is simply a matter of changing the 1:N notation to N:M. However, once the database has been constructed, and loaded with data and application forms, reports, queries, and application programs have been created, changing a 1:N relationship to N:M means weeks of work.

You can glean some idea of why this might be true by contrasting Figure AE5a-10c with Figure AE5a-11c. Suppose that instead of having just a few rows, each table has thousands of rows; in that case, transforming the database from one format to the other involves considerable work. Even worse, however, is that application components will need to be changed as well. For example, if students have at most one adviser, then a single text box can be used to enter *AdviserName.* If students can have multiple advisers, then a multiple-row table will need to be used to enter *AdviserName*, and a program will need to be written to store the values of *AdviserName* into the *Adviser_Student_Intersection* table. There are dozens of other consequences as well, consequences that will translate into wasted labour and wasted expense.

The conclusion from this discussion is that user review of a data model is crucial. When a database is developed for your use, you must carefully review the data model. If you do not understand any aspect of it, you should ask for clarification until you do. The data model must accurately reflect your view of the business. If it does not, the database will be designed incorrectly, and the applications will be difficult to use, if not worthless. Do not proceed unless the data model is accurate.

As a corollary, when asked to review a data model, take that review seriously. Devote the time necessary to perform a thorough review. Any mistakes you miss will come back to haunt you, and by then the cost of correction may be very high with regard to both time and expense. This brief introduction to data modelling shows why databases can be more difficult to develop than spreadsheets.

Q6 Who Will Volunteer? (Continued)

Knowing what you know now, if you were the manager of fund-raising at the TV station, you would hire a consultant and expect the consultant to interview all of the key users. From those interviews, the consultant would then construct a data model.

You now know that the structure of the database must reflect the way the users think about their activities. If the consultant did not take the time to interview you and your staff or did not construct a data model and ask you to review it, you would know that you are not receiving good service and would take corrective action.

Suppose you found a consultant who interviewed your staff for several hours and then constructed the data model shown in Figure AE5a-12. This data model has an

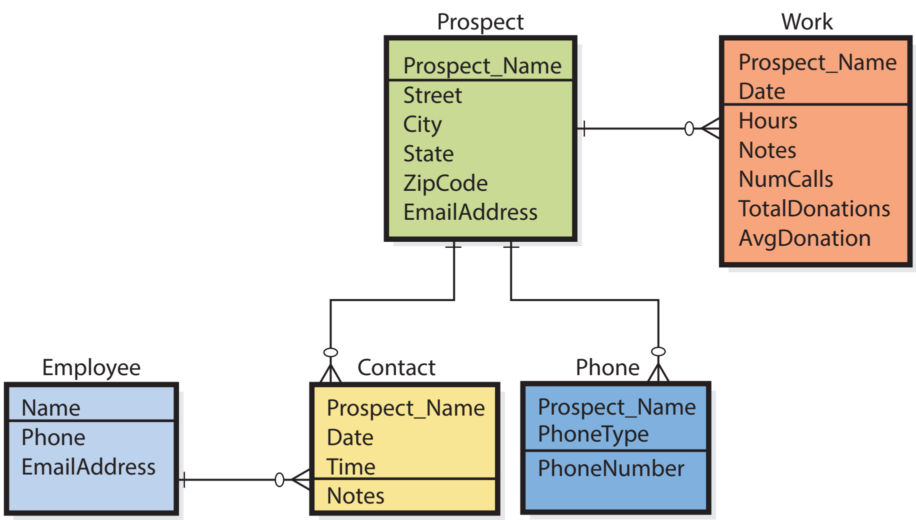


Figure AE5a-12
Data Model for Volunteer Database

Figure AE5a-13

First Table Design for Volunteer Database

Prospect (Name, Street, City, Province, Postal Code, EmailAddress)
Phone (*Name*, PhoneType, PhoneNumber)
Contact (*Name*, Date, Time, Notes, *EmployeeName*)
Work (*Name*, Date, Notes, NumCalls, TotalDonations)
Employee (EmployeeName, Phone, EmailAddress)

Note:
Underline means table key.
Italics means foreign key.
Underline and italics mean both table and foreign key.

entity for *Prospect*, an entity for *Employee*, and three additional entities for *Contact, Phone*, and *Work*. The *Contact* entity records contacts that you or other employees have made with the prospective volunteer. This record is necessary so that you know what has been said to whom. The *Phone* entity is used to record multiple phone numbers for each prospective volunteer, and the *Work* entity records work that the prospect has performed for the station.

After you reviewed and approved this data model, the consultant constructed the database design shown in Figure AE5a-13. In this design, table keys are underlined, foreign keys are shown in italics, and columns that are both table and foreign keys are underlined and italicized. Observe that the *Name* column is the table key of *Prospect*, and it is both part of the table key and a foreign key in *Phone, Contact*, and *Work.*

The consultant did not like having the *Name* column used as a key or as part of a key in so many tables. Based on her interviews, she suspected that prospect names are fluid—and that sometimes the same prospect name is recorded in different ways (e.g., sometimes with a middle initial and sometimes without). If that were to happen, phone, contact, and work data could be misallocated to prospect names. Accordingly, the consultant added a new column, *ProspectID*, to the prospect table and created the design shown in Figure AE5a-14. Values of this ID will have no meaning to the users, but the ID will be used to ensure that each prospect obtains a unique record in the Volunteer database. Because this ID has no meaning to the users, the consultant will hide it on forms and reports that users see.

There is one difference between the data model and the table designs. In the data model, the *Work* entity has an attribute, *AvgDonation*, but there is no corresponding *AvgDonation* column in the *Work* table. The consultant decided that there was no need to store this value in the database because it could readily be computed on forms and reports using the values in the *NumCalls* and *TotalDonations* columns.

Figure AE5a-14

Second Table Design for Volunteer Database

Prospect (*ProspectID*, Name, Street, City, Province, Postal Code, EmailAddress)
Phone (*ProspectID*, PhoneType, PhoneNumber)
Contact (*ProspectID*, Date, Time, Notes, *EmployeeName*)
Work (*ProspectID*, Date, Notes, NumCalls, TotalDonations)
Employee (EmployeeName, Phone, EmailAddress)

Note:
Underline means table key.
Italics means foreign key.
Underline and italics mean both table and foreign key.

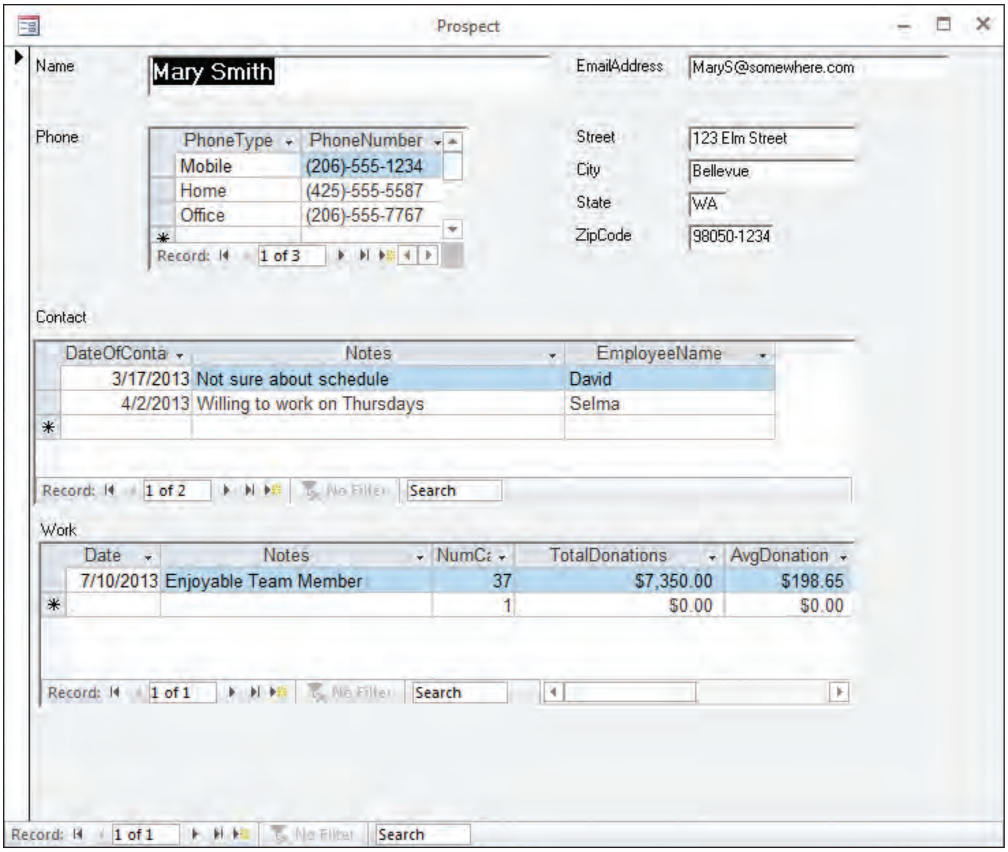


Figure AE5a-15

Volunteer Prospect Data Entry Form

Source: Microsoft Access 2013.

Once the tables had been designed, the consultant created a Microsoft Access 2013 database. She defined the tables in Access, created relationships among the tables, and constructed forms and reports. Figure AE5a-15 shows the primary data entry form used for the Volunteer database. The top portion of the form has contact data, including multiple phone numbers. It is important to know the type of the phone number so that you and your staff know if you're calling someone at work or another setting. The middle and bottom sections of this form have contact and prior work data. Observe that *AvgDonation* has been computed from the *NumCalls* and *TotalDonations* columns.

You were quite pleased with this database application, and you're certain that it helped you to improve the volunteer staffing at the station. Of course, over time, you thought of several new requirements, and you already have changes in mind for next year.

Active Review

Use this Active Review to verify that you understand the ideas and concepts that answer this chapter extension's study questions.

Q1 Who will volunteer?

Summarize the problem that the fund-raising manager must solve. Explain how a database can help solve this problem. Describe the missing information. In your own words, what data must be available to construct the missing information?

Q2 How are database application systems developed?

Name and briefly describe the components of a database application system. Explain the difference between a database application system and a database application program. Using Figure AE5a-1 as a guide, describe the major steps in the process of developing a database application system. Explain what role is crucial for users and why that role is so important.

Q3 What are the components of the entity-relationship data model?

Define the terms *entity, attributes*, and *relationship*. Give an example of two entities (other than those in this book) that have a 1:N relationship. Give an example of two entities that have an N:M relationship. Explain the difference between maximum and minimum cardinality. Show two entities having a 1:N relationship in which one is required and one is optional.

Q4 How is a data model transformed into a database design?

Give an example of a data integrity problem. Describe, in general terms, the process of normalization. Explain how normalizing data prevents data integrity problems. Explain the disadvantage of normalized data. Using your examples from Q3, show how 1:N relationships are expressed in relational database designs. Show how N:M relationships are expressed in relational database designs.

Q5 What is the users' role?

Describe the major role for users in the development of a database application system. Explain what is required to change a 1:N relationship to an N:M relationship during the data modelling stage. Explain what is required to make that same change after the database application system has been constructed. Describe how this knowledge impacts your behaviour when a database application system is being constructed for your use.

Q6 Who will volunteer? (Continued)

Examine Figure AE5a-12. Describe the maximum and minimum cardinality for each relationship. Justify these cardinalities. Change the relationship between *Prospect* and *Phone* to N:M, and explain what this means. Change the relationship between *Prospect* and *Work* to 1:1, and explain what this means. Explain how each relationship is represented in the design in Figure AE5a-14. Show examples of both primary keys and foreign keys in this figure. In *Contact*, determine whether *EmployeeName* is part of a primary key or part of a foreign key.

Explain what problem the consultant foresaw in the use of the *Name* attribute. Explain how that problem was avoided. The consultant added an attribute to the data model that was not part of the users' world. Explain why that attribute will not add unnecessary complication to the users' work experiences.

MyMISLab MyMISLab is an online learning and testing environment that features the perfect study tools to help you master the concepts covered in this chapter. Log in to MyMISLab to test your knowledge of key chapter concepts and explore additional practice tools, including videos, flashcards, annotated text figures, and more!

Key Terms and Concepts

Using Your Knowledge

AE5a-1. Explain how you could use a spreadsheet to solve the volunteer problem at the television station. What data would you place in each column and row of your spreadsheet? Name each column and row of your spreadsheet. What advantages does a database have over a spreadsheet for this problem? Compare and contrast your spreadsheet solution to the database solution shown in the design in Figure AE5a-14 and the data entry form in Figure AE5a-15.

AE5a-2. Suppose you are asked to build a database application for a sports league. Assume that your application is to keep track of teams and equipment that is checked out to teams. Explain the steps that need to be taken to develop this application. Specify entities and their relationships. Build an E-R diagram. Ensure your diagram shows both minimum and maximum cardinalities. Transform your E-R diagram into a relational design.

AE5a-3. Suppose you are asked to build a database application for a bicycle rental shop. Assume your database is to track customers, bicycles, and rentals. Explain the steps that need to be taken to develop this application. Specify entities and their relationships. Build an E-R diagram. Ensure your diagram shows both minimum and maximum cardinalities. Transform your E-R diagram into a relational design.

MyMISLab

Go to MyMISLab for auto-graded writing questions as well as the following assisted-graded writing questions:

AE5a-4. Assume you work at the television station and are asked to evaluate the data model in Figure AE5a-12. Suppose that you want to differentiate between prospects who have worked in the past and those who have never worked, but who are prospects for future work. Say that one of the data modelers tells you, "No problem. We'll know that because any *Prospect* entity that has no relationship to a *Work* entity is a prospect who has never worked." Restate the data modeler's response in your own words. Does this seem like a satisfactory solution? What if you want to keep *Prospect* data that pertains only to prospects who have worked? (No such attributes are shown in *Prospect* in Figure AE5a-12, but say there is an attribute such as *YearFirstVolunteered* or some other attribute that pertains to prospects who have worked in the past.) Show an alternative E-R diagram that would differentiate between prospects who have worked in the past and those who have not. Compare and contrast your alternative to the one shown in Figure AE5a-12.

AE5a-5. Suppose you manage a department that is developing a database application. The IT professionals who are developing the system ask you to identify two employees to evaluate data models. What criteria would you use in selecting those employees? What instructions would you give them? Suppose one of the employees says to you, "I go to those meetings, but I just don't understand what they're talking about." How would you respond? Suppose that you go to one of those meetings and don't understand what they're talking about. What would you do? Describe a role for a prototype in this situation. How would you justify the request for a prototype?

AE5a-6. MyMISLab Only – comprehensive writing assignment for this chapter.

Chapter Extension 5b

Chapter 5 provides the background for this extension.

Using Microsoft Access 2013

In this chapter extension, you will learn fundamental techniques for creating a database and forms, queries, and reports with Microsoft Access.

Study Questions

Q1 How do you create tables?

Q2 How do you create relationships?

Q3 How do you create a data entry form?

Q4 How do you create queries using the query design tool?

Q5 How do you create a report?

MyMISLab

Visit MyMISLab for simulations, tutorials, and end-of-chapter problems.

Q1 How Do You Create Tables?

Before using Access or any other DBMS, you should have created a data model from the users' requirements, and you must transform that data model into a database design. For the purpose of this chapter extension, we will use a portion of the database design created in Chapter Extension 5a. Specifically, we will create a database with the following two tables:

PROSPECT (ProspectID, Name, Street, City, Province, Postal Code, Email Address)

and

WORK (*ProspectID*, Date, Hour, NumCalls, TotalDonations)

As in Chapter Extension 5a, an underlined attribute is the primary key and an italicized attribute is a foreign key. Thus, ProspectID is the primary key of PROSPECT, and the combination (ProspectID, Date, Hour) is the primary key of WORK. *ProspectID* is also a foreign key in WORK; hence it is shown both underlined and in italics. The data model and database design in Chapter Extension 5a specified that the key of WORK is the combination (*ProspectID*, Date). Upon review, the users stated that prospects will sometimes work more than one time during the day. For scheduling and other purposes, the users want to record both the date and the hour that someone worked. Accordingly, the database designer added the Hour attribute and made it part of the key of WORK.

The assumption in this design is that each row of WORK represents an hour's work. If a prospect works for consecutive hours, say from 7 to 9 p.m., then he or she would have two rows, one with an Hour value of 1900 and a second with an Hour value of 2000. Figure AE5b-1 further documents the attributes of the design. Sample data for this table are shown in Figure AE5b-2 on page 175.

Note the ambiguity in the name *PROSPECT.* Before someone has become a volunteer, he is a prospect, and the term is fine. However, once that person has actually done work, he is no longer merely a prospect. This ambiguity occurs because the database is used both for finding volunteers and for recording their experiences once they have joined. We could rename PROSPECT as VOLUNTEER, but then we'd still have a problem. The person is not a volunteer until he has actually agreed to become one. So, for now, just assume that a PROSPECT who has one or more WORK records is no longer a prospect but has become a volunteer.

Starting Access

Figure AE5b-3 shows the opening screen for Microsoft Access 2013. (If you use another version of Access, your screen will appear differently, but the essentials will be the same.) To create a new database, select Blank desktop database in the templates

Table	Attribute (Column)	Remarks	Data Type	Example Value
PROSPECT	ProspectID	An identifying number provided by Access when a row is created. The value has no meaning to the user.	AutoNumber	55
PROSPECT	Name	A prospect's name.	Text (50)	Emily Jones
PROSPECT	Street	Prospect's contact street address.	Text (50)	123 West Elm
PROSPECT	City	Prospect's contact city.	Text (40)	Miami
PROSPECT	State	Prospect's contact state.	Text (2)	FL
PROSPECT	Zip	Prospect's contact zip code.	Text (10)	30210-4567 or 30210
PROSPECT	EmailAddress	Prospect's contact email address.	Text (65)	ExamplePerson@somewhere.com
WORK	ProspectID	Foreign key to PROSPECT. Value provided when relationship is created.	Number (Long Integer)	55
WORK	Work Date	The date of work.	Date	9/15/2014
WORK	Hour	The hour at which work is started.	Number (Integer)	0800 or 1900 (7 PM)
WORK	NumCalls	The number of calls taken.	Number (Integer)	25
WORK	TotalDonations	The total of donations generated.	Currency	$10 575
WORK	AvgDonations	The average donation.	Currency	To be computed in queries and reports

Figure AE5b-1
Attributes of the Database

displayed in the centre of the screen, as shown in Figure AE5b-4. Then type the name of your new database under *File Name* (here we use *Volunteer*). Access will suggest a directory; change it if you want to use another one, and then click *Create*. You will see the screen shown in Figure AE5b-5 on page 176.

Creating Tables

Access opens the new database by creating a default table named Table. We want to modify the design of this table, so in the upper left-hand corner, where you see a pencil and a right angle square, click *View* and select *Design View*. Access will ask you to name your table. Enter *PROSPECT* and click *OK*. Your screen will appear as in Figure AE5b-6 on page 177.

The screen shown in Figure AE5b-6 has three parts. The left-hand pane lists all of the tables in your database. At this point, you should see only the PROSPECT table in this list. We will use the upper part of the right-hand pane to enter the name of each attribute (which Access calls *Fields*) and its *Data Type*. We can optionally enter a *Description* of that field. The Description is used for documentation; as you will see, Access displays any text you enter as help text on forms. In the bottom part of the screen, we set the properties of each field (or attribute, using our term). To start

Example of PROSPECT Data

Prospect ID	Name	Street	City	State	Zip	EmailAddress
1	Carson Wu	123 Elm	Los Angeles	CA	98007	Carson@somewhere.com
2	Emily Jackson	2234 17th	Pasadena	CA	97005	JacksonE@elsewhere.com
3	Peter Lopez	331 Moses Drive	Fullerton	CA	97330	PeterL@ourcompany.com
4	Lynda Dennison	54 Strand	Manhattan Beach	CA	97881	Lynda@somewhere.com
5	Carter Fillmore III	Restricted	Brentwood	CA	98220	Carter@BigBucks.com
6	CJ Greene	77 Sunset Strip	Hollywood	CA	97330	CJ@HollywoodProducers.com
7	Jolisa Jackson	2234 17th	Pasadena	CA	97005	JacksonJ@elsewhere.com

Example of WORK Data

ProspectID	Work Date	Hour	NumCalls	TotalDonations
3	9/15/2014	1600	17	8755
3	9/15/2014	1700	28	11578
5	9/15/2014	1700	25	15588
5	9/20/2014	1800	37	29887
5	9/10/2015	1700	30	21440
5	9/10/2015	1800	39	37050
6	9/15/2014	1700	33	21445
6	9/16/2014	1700	27	17558
6	9/10/2015	1700	31	22550
6	9/10/2015	1800	37	36700

Figure AE5b-2
Sample Data

designing the table, replace the *Field Name* ID with *ProspectID.* Access has already set its type to *AutoNumber*, so you can leave that alone.

To create the rest of the table, enter the *Field Names* and *Data Types* according to our design.[1] Figure AE5b-7 shows how to set the length of a Short Text Data Type. In this figure, the user has set City to *Text* and then has moved down into the bottom part of this form and entered 40 under *Field Size*. You will do the same thing to set the length of all of the Short Text Field Names. The complete table is shown in Figure AE5b-8.

[1] When you enter the Name field, Access will give you an error message. Ignore the message and click OK. The fact that you are using a reserved word for this example will not be a problem. If you want to be safe, you could enter *PName* or *ProspectName* (rather than *Name*) and avoid this issue. Many people, including me, believe that Access is poorly designed in this respect. You ought to be able to enter any value for Field Name the way you want. Access should stay out of your way; you shouldn't have to stay out of its way!

Figure AE5b-3

Opening Screen for Microsoft Access 2013

Source: Microsoft Access 2013.

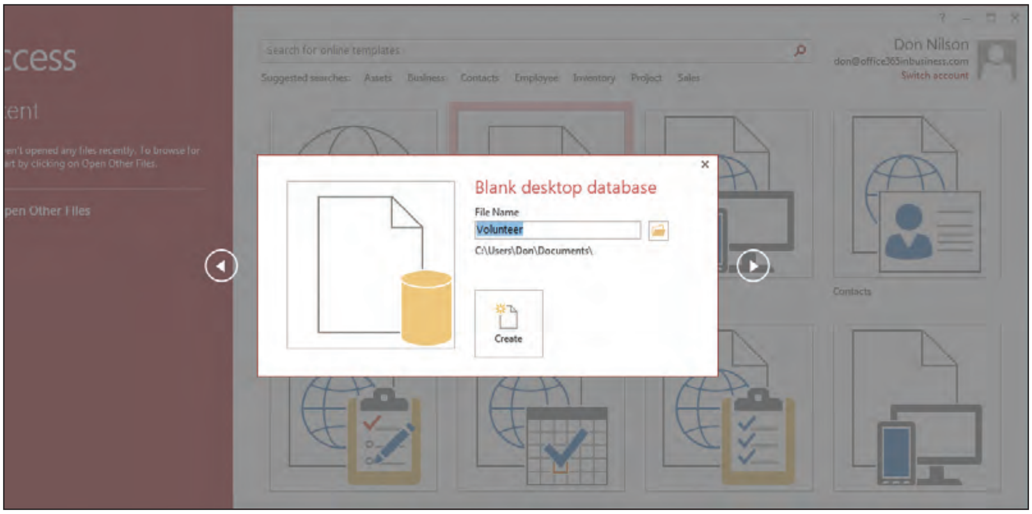


Figure AE5b-4

Naming a Desktop Database

Source: Microsoft Access 2013.

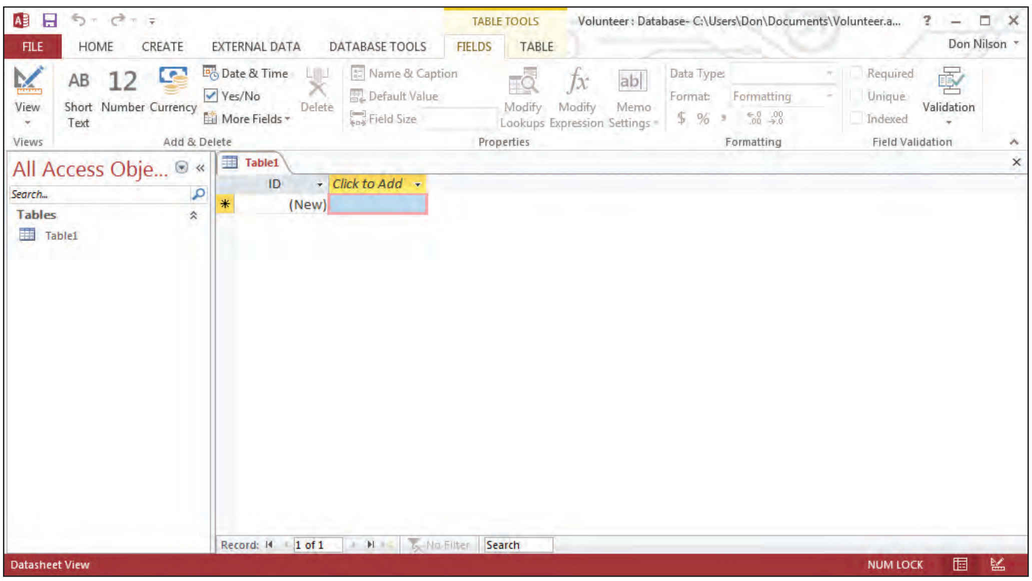


Figure AE5b-5

Access Opens with an Initial Table Definition

Source: Microsoft Access 2013.

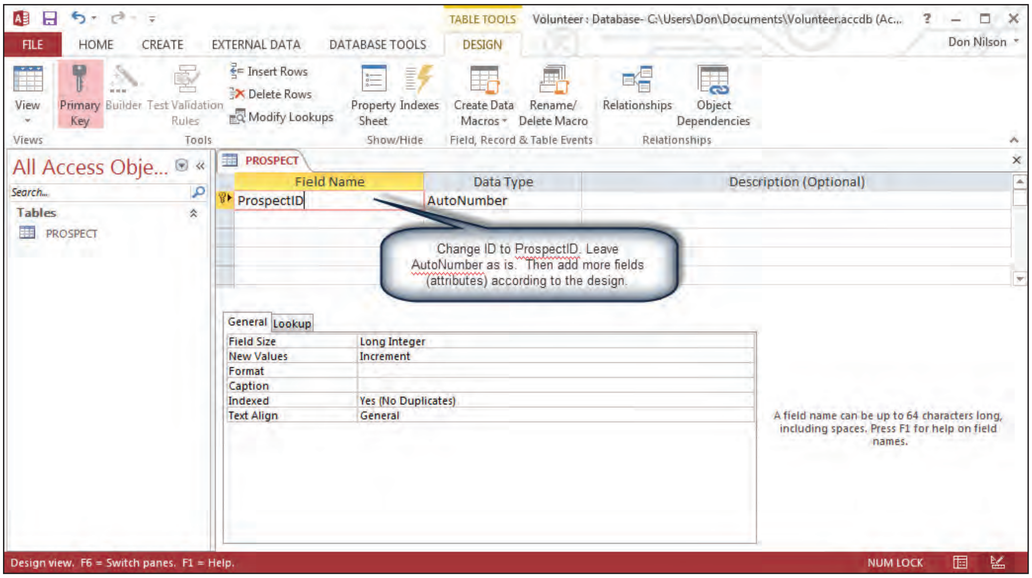


Figure AE5b-6

Creating Tables in Access, Step 1

Source: Microsoft Access 2013.

ProspectID is the primary key of this table, and the little key icon next to the ProspectID *Field Name* means Access has already made it so. If we wanted to make another field the primary key, we would highlight that field and then click the *Primary Key* icon in the left-hand portion of the *DESIGN* ribbon.

Follow similar steps to create the WORK table. The only difference is that you will need to create a key of the three columns (ProspectID, WorkDate, Hour). To create that key, highlight all three rows by dragging over the three squares to the left of the names of ProspectID, WorkDate, and Hour. Then click the *Key* icon in the *DESIGN* ribbon. Also, change the *Required Field Property* for each of these columns to *Yes.* The finished WORK table is shown in Figure AE5b-9. This figure also shows that the user selected *Number* for the *Data Type* of *NumCalls* and then set its *Field Size* (lower pane) to *Integer.* This same technique was used to set the *Data Type* of ProspectID (in WORK) to *Number* (*Field Size* of *Long Integer*) and that of *Hour* to *Number* (*Field Size* of *Integer*).

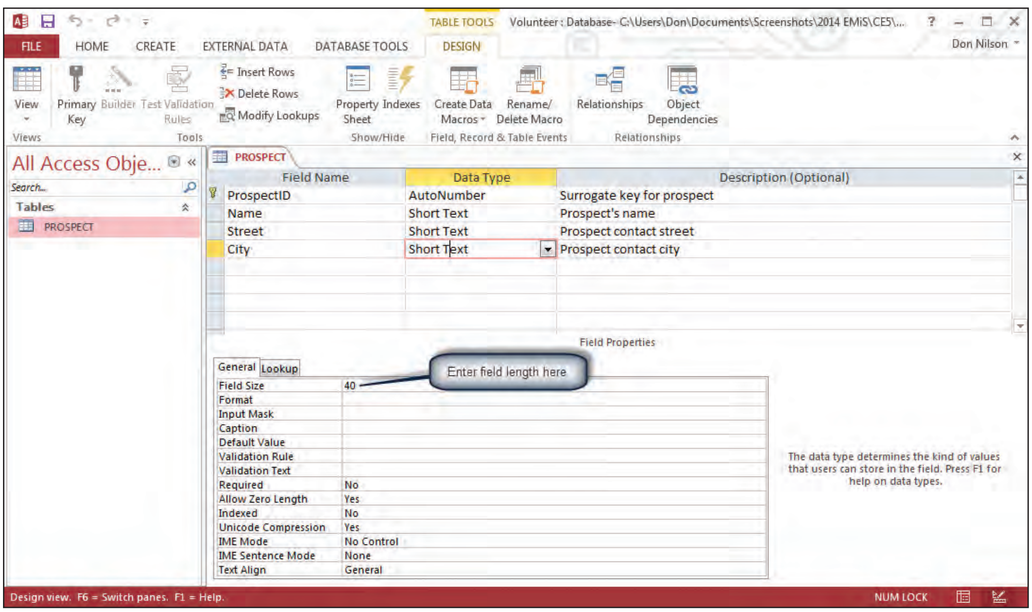


Figure AE5b-7

Creating Tables in Access, Step 2

Source: Microsoft Access 2013.

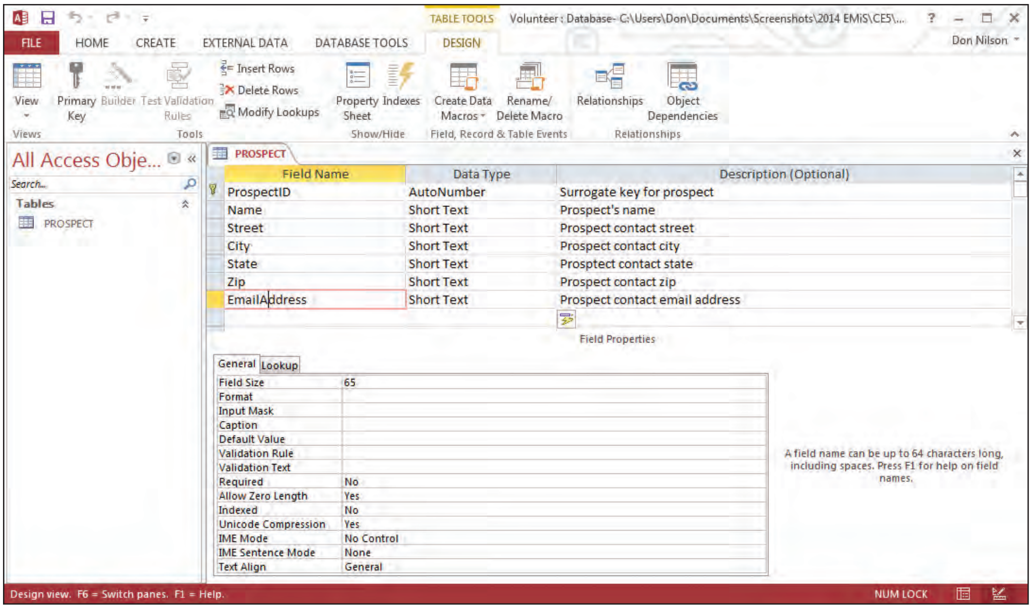


Figure AE5b-8

Complete Sample PROSPECT Table

Source: Microsoft Access 2013.

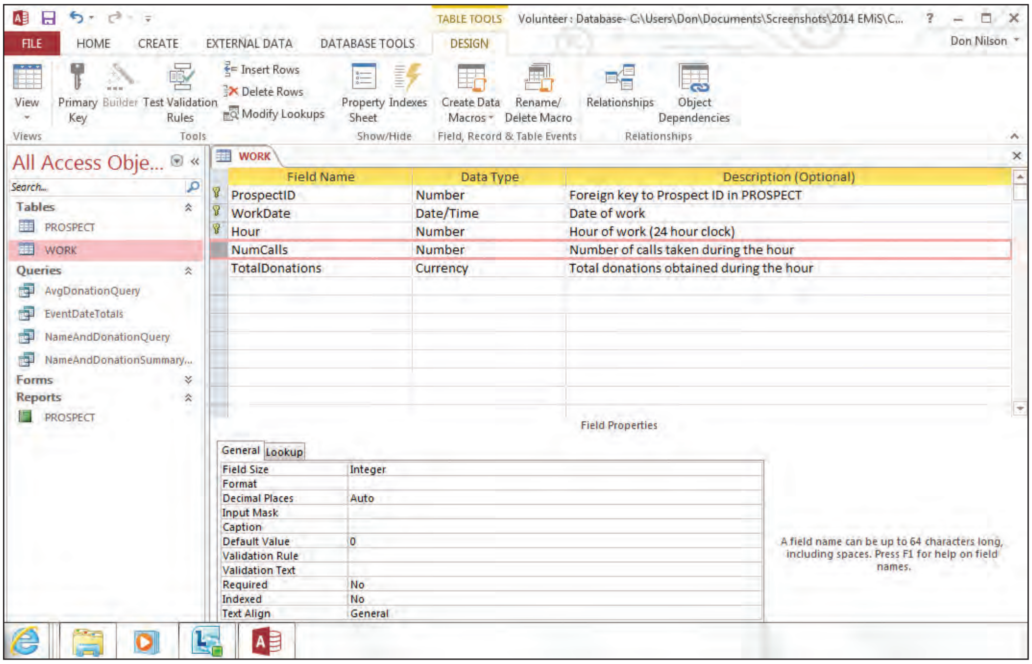


Figure AE5b-9

Finished WORK Table

Source: Microsoft Access 2013.

At this point, close both tables and save your work. You have created your first database!

Q2 How Do You Create Relationships?

After you have created the tables, the next step is to define relationships. To do so, click the *DATABASE TOOLS* tab in the ribbon and then click the *Relationships* icon near the left-hand side of that ribbon. The *Relationships* window will open and the *Show Table* dialog box will be displayed, as shown in Figure AE5b-10. Double-click both table names and both tables will be added to the *Relationships* window. Close the *Show Table* dialog box.

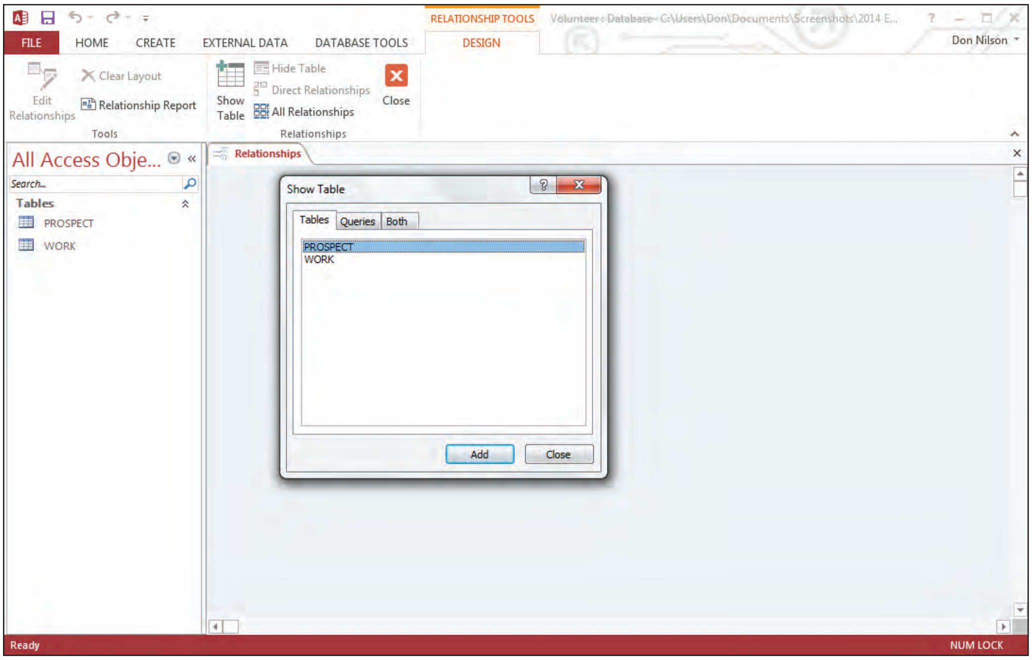


Figure AE5b-10
The Show Table Dialog Box in Access

Source: Microsoft Access 2013.

To create the relationship between these two tables, click on the attribute *ProspectID* in PROSPECT and drag that attribute on top of the *ProspectID* in WORK. (It is important to drag *ProspectID* from PROSPECT to WORK and not the reverse.) When you do this, the screen shown in Figure AE5b-11 will appear.

In the dialog box, click *Enforce Referential Integrity*, click *Cascade Update Related Fields*, and then click *Cascade Delete Related Records.* The specifics of these actions are beyond the scope of our discussion. Just understand that clicking these options will cause Access to make sure that ProspectID values in WORK also exist in PROSPECT. The completed relationship is shown in Figure AE5b-12. The notation *1*... ∞ at the end of the relationship line means that one row of PROSPECT can be related to an unlimited number (*N*) of rows in WORK. Close the *Relationships* window and save the changes when requested to do so. You now have a database with two tables and a relationship.

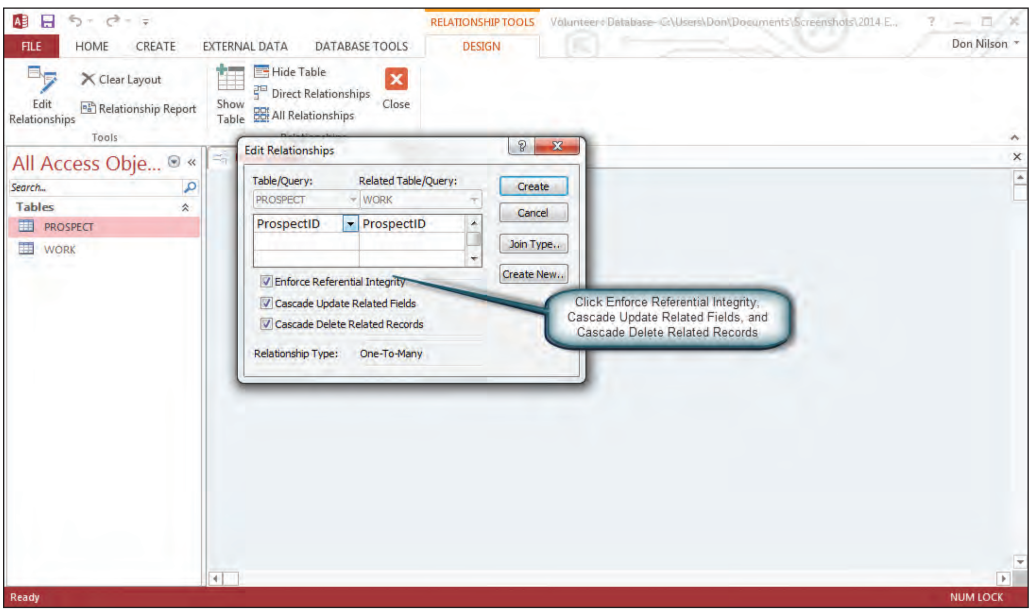


Figure AE5b-11
Creating a Relationship Between Two Tables

Source: Microsoft Access 2013.

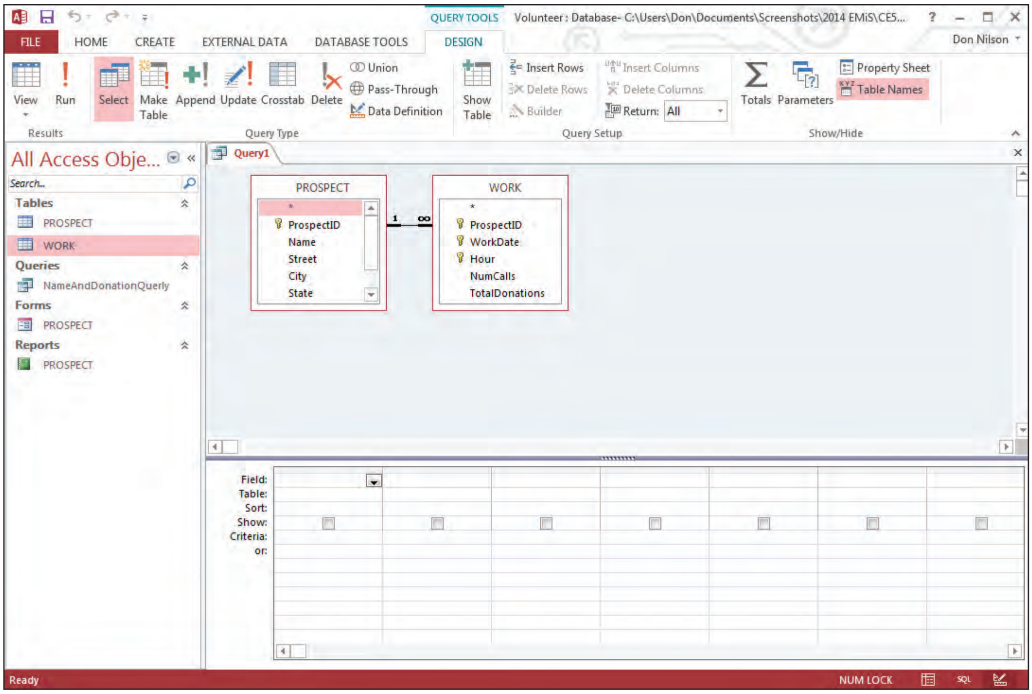


Figure AE5b-12

Completed Relationship Between PROSPECT and WORK Tables

Source: Microsoft Access 2013.

The next step is to enter data. To enter data, double-click the table name in the *left hand* pane. The table will appear, and you can enter values into each cell. You cannot and need not enter values for the *ProspectID* field. Access will create those values for you.

Enter the data in Figure AE5b-2 for both PROSPECT and WORK, and you will see a display like that in Figures AE5b-13a and AE5b-13b. Examine the lower left-hand corner of Figure AE5b-13b. The text *Foreign key to ProspectID in PROSPECT* is the Description that you provided when you defined the ProspectID column when the WORK table was created. (You can see this in the ProspectID column in Figure AE5b-9.) Access displays this text because the focus is on the ProspectID column in the active table window (WORK). Move your cursor from field to field and watch this text change.

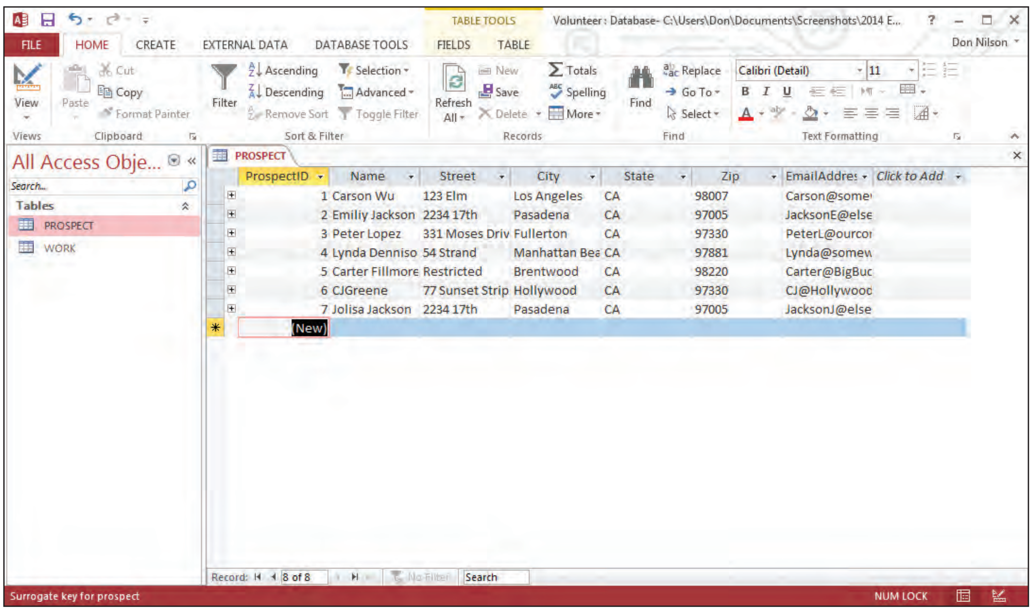


Figure AE5b-13a

Tables with Data Entered for PROSPECT

Source: Microsoft Access 2013.

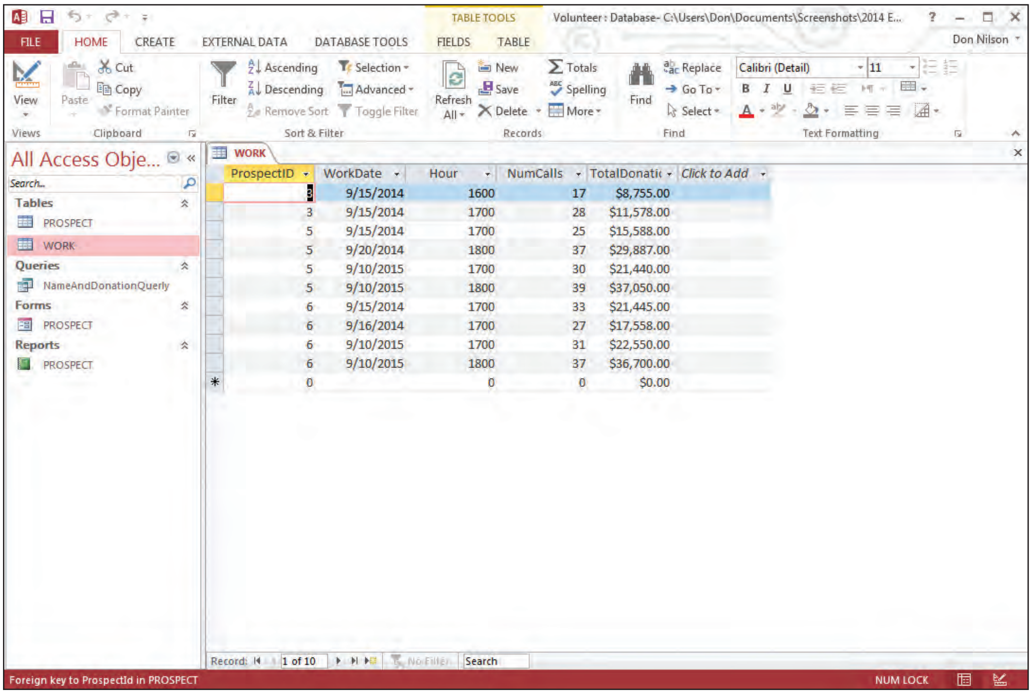


Figure AE5b-13b

Tables with Data Entered for WORK

Source: Microsoft Access 2013.

Q3 How Do You Create a Data Entry Form?

Access provides several alternatives for creating a data entry form. The first is to use the default table display, as you did when you entered the data shown in Figure AE5b-13. In the PROSPECT table, notice the plus sign on the left. If you click those plus signs, you will see the PROSPECT rows with their related WORK rows, as shown in Figure AE5b-14. This display, although convenient, is limited in its

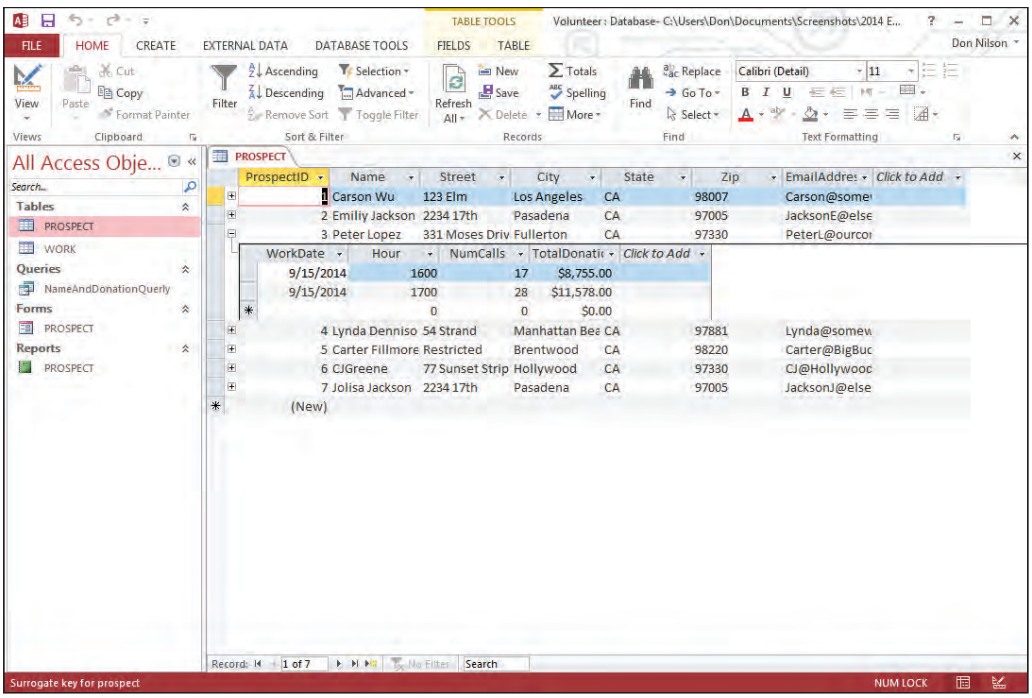


Figure AE5b-14

Default Table Display

Source: Microsoft Access 2013.

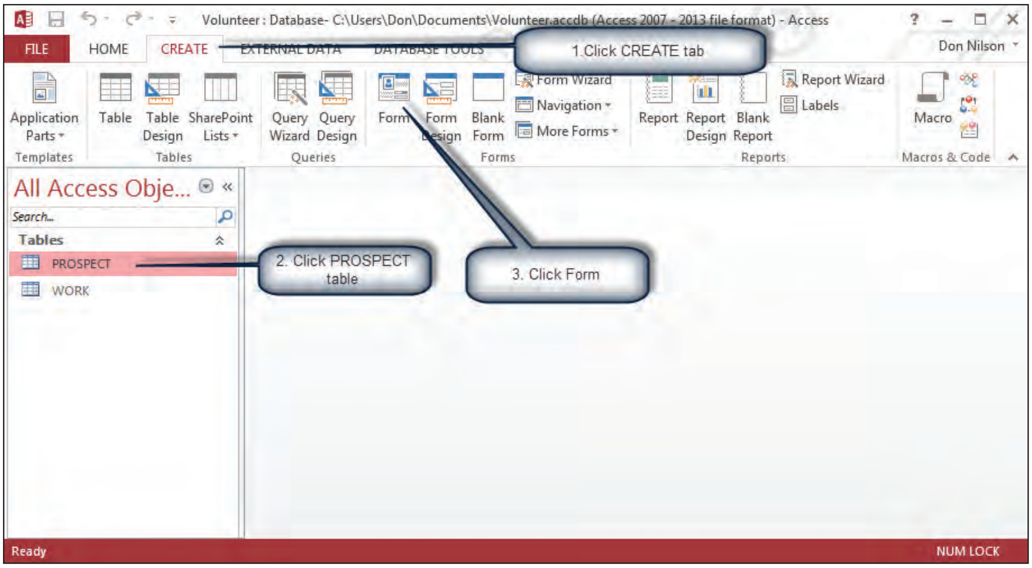


Figure AE5b-15

Starting the Form Generator

Source: Microsoft Access 2013.

capability. It also does not provide a very pleasing user interface. For more generality and better design, you can use the Access form generator.

Access can generate a data entry form that is more pleasing to view and easier to use than that in Figure AE5b-14. The process is shown in Figure AE5b-15. First, click the *CREATE* tab to open the *CREATE* ribbon. Next, click the PROSPECT table (this causes Access to create a form for PROSPECT). Finally, click *Form.* Access uses metadata about the tables and their relationship to create the data entry form in Figure AE5b-16.

You can use this form to modify data; just type over any data that you wish to change. You can also add data. To add work data, just click in the last row of the work grid; in this case that would be the first row of this grid. To delete a record, click the *HOME* tab, and then in the *Records* section click the down arrow next to *Delete* and select *Delete Record.* This action will delete the prospect data and all related work data (not shown in Figure AE5b-16).

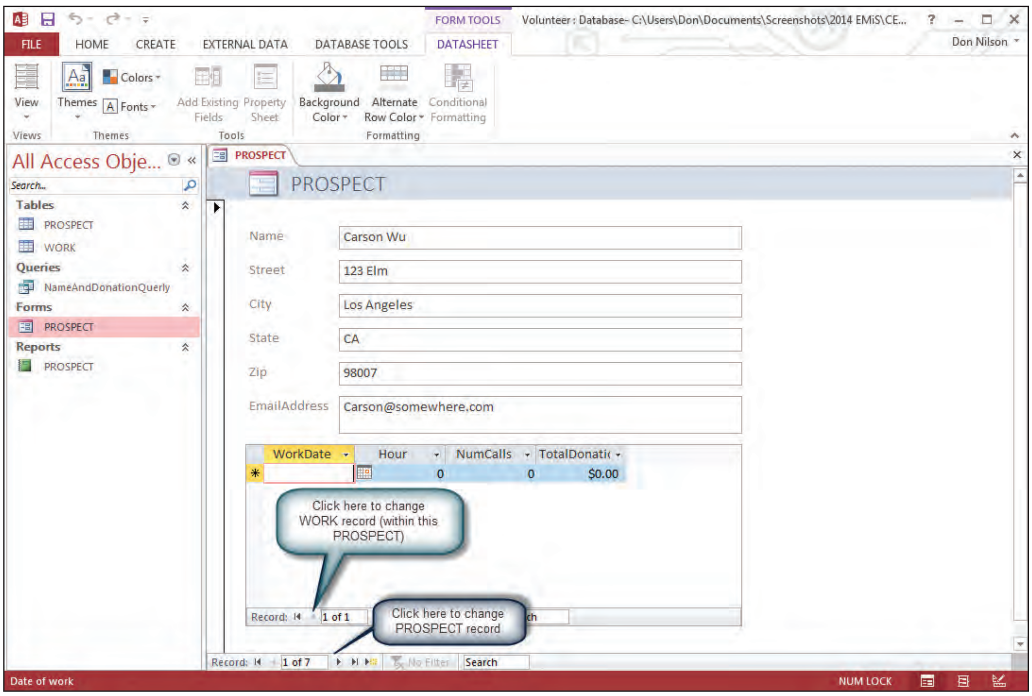


Figure AE5b-16

Resulting Data Entry Form

Source: Microsoft Access 2013.

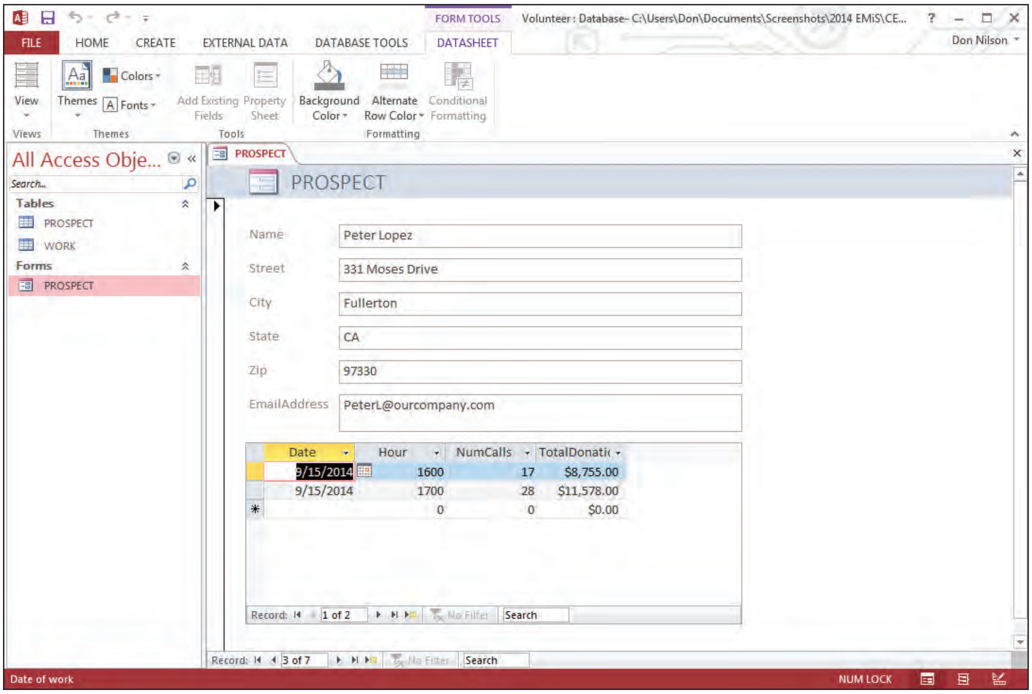


Figure AE5b-17
Reformatted Data Entry Form

Source: Microsoft Access 2013.

This form is fine, but we can make it better. For one, ProspectID is a surrogate key and has no meaning to the user. Access uses that key to keep track of each PROSPECT row. Because it has no meaning to the user (in fact, the user cannot change or otherwise modify its value), we should remove it from the form. Also, we might like to reduce the size of the fields as well as reduce the size of the work area and centre it on the form. Figure AE5b-17 shows the form after these changes. It is smaller and cleaner, and it will be easier to use.

The data about a prospect is shown in the top portion of this form, and data about that person's work sessions is shown in the bottom portion. The user of this form has clicked the arrow at the bottom of the form to bring up the third Prospect record, the one for Peter Lopez. Notice that he has two work sessions. If you click the arrow in the next-to-last row of this form, you will change the focus of the work record. To make the changes shown, see the steps illustrated in Figure AE5b-18. First, right-click

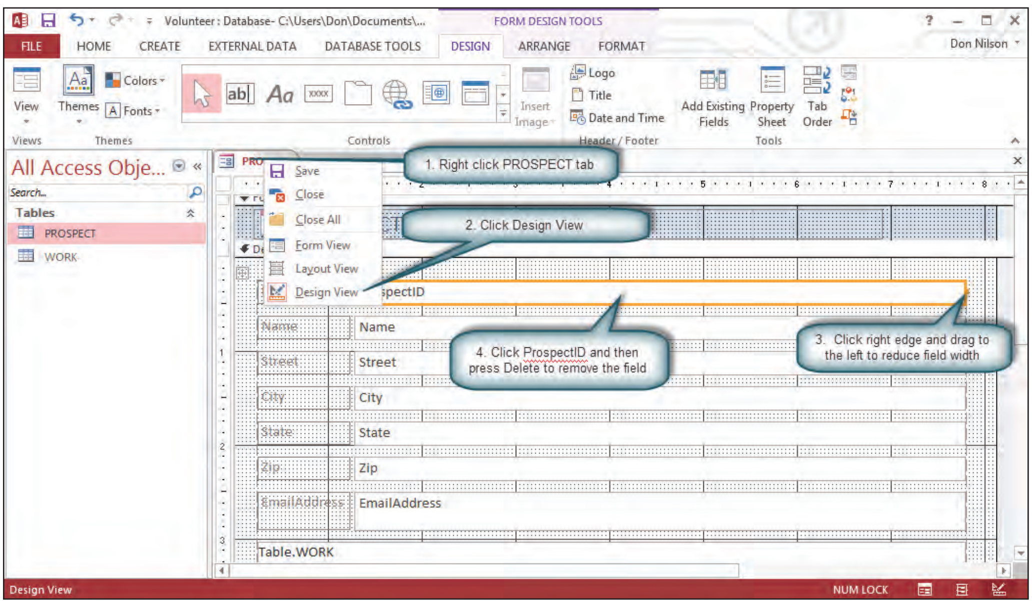


Figure AE5b-18
Process for Reformatting Data Entry Form

Source: Microsoft Access 2013.

the PROSPECT tab and then select *Design View.* The form will open in Design mode; click the right edge of the rightmost rectangle and, holding your mouse down, drag to the left. Access will reduce the width of each of these fields as well as the table.

Finally, click *ProspectID*, as shown in step 4. Press the *Delete* key, and the ProspectID field will be removed from the form. Click *View/Form View*, and your form should look like that in Figure AE5b-17. You can go back to *Design View* to make more adjustments, if necessary.

To save your form, either close it and Access will give you the chance to save it or click *FILE* and select *Save.* Save with an informative file name, such as PROSPECT Data Entry Form.

There are many options for customizing Access forms. You can learn about them if you take a database processing class after you complete this MIS class.

Q4 How Do You Create Queries Using the Query Design Tool?

Like all relational DBMS products, Access can process the SQL query language. Learning that language, however, is beyond the scope of this textbook. However, Access does provide a graphical interface that we can use to create and process queries, and that graphical interface will generate SQL statements for us, behind the scenes.

To begin, first clean up your screen by closing the PROSPECT Data Entry Form. Click the *CREATE* tab in the ribbon, and in the *Queries* section click the *Query Design* button. You should see the display shown in Figure AE5b-19. Double-click the names of both the PROSPECT and WORK tables, and close the *Show Table* window. Access will have placed both tables into the query design form, as shown in Figure AE5b-20. Notice that Access remembers the relationship between the two tables (shown by the line connecting ProspectID in PROSPECT to the same attribute in WORK).

To create a query, drag columns out of the PROSPECT and WORK tables into the grid in the lower part of the query definition form. In Figure AE5b-21, the *Name, EmailAddress, NumCalls*, and *TotalDonations* columns have been placed into that

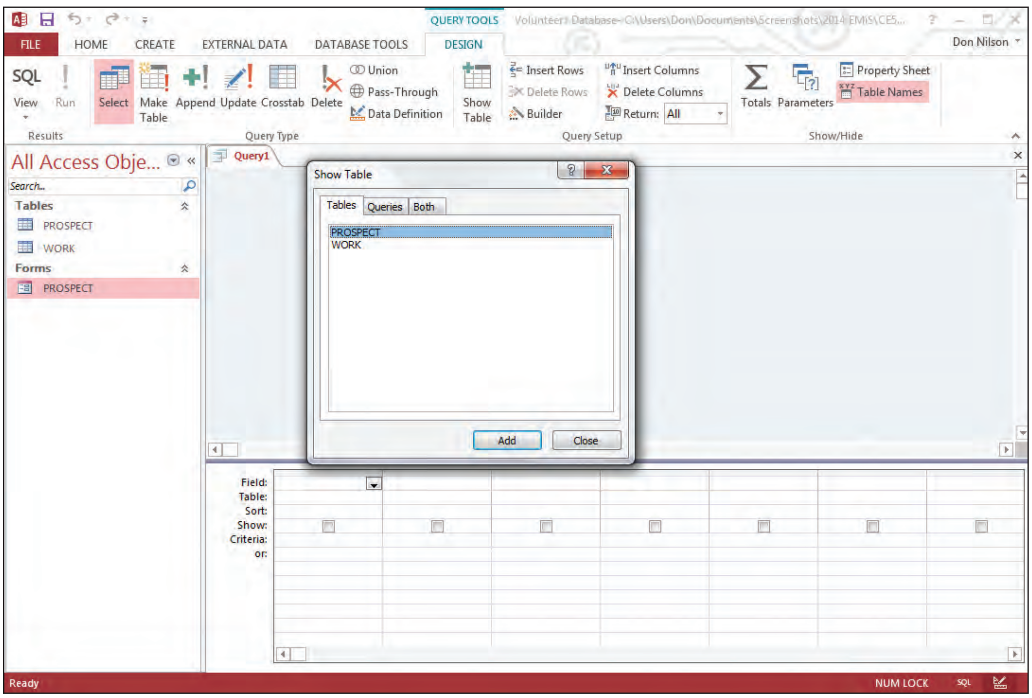


Figure AE5b-19

Creating a Query, Step 1

Source: Microsoft Access 2013.

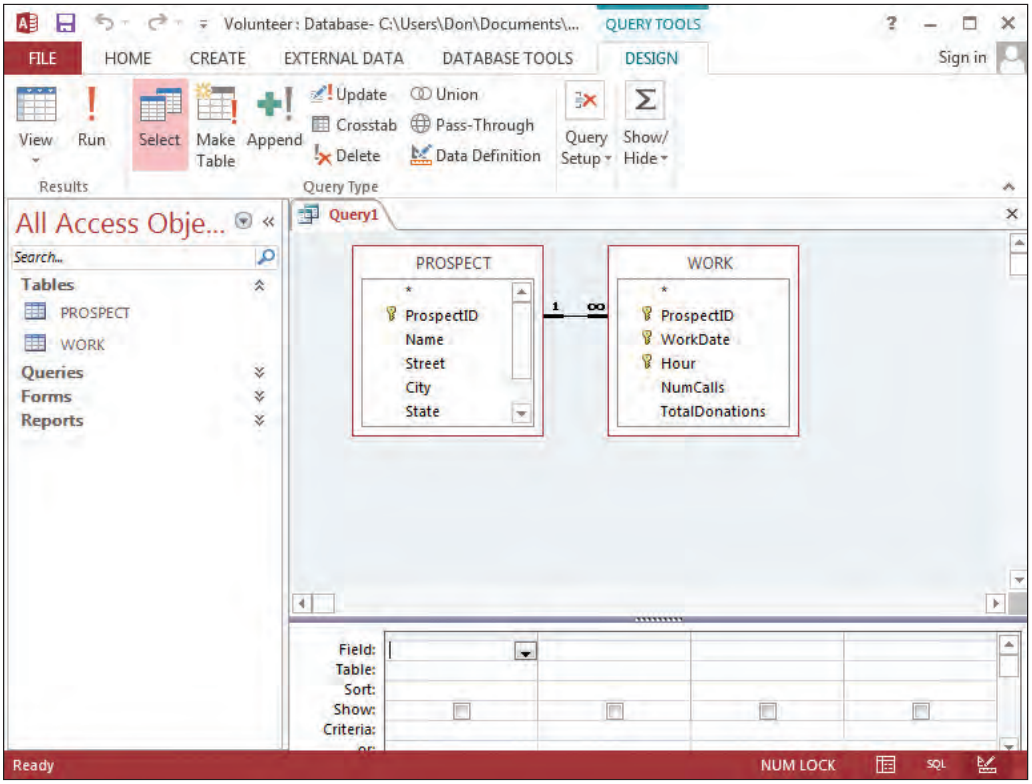


Figure AE5b-20
Creating a Query, Step 2

Source: Microsoft Access 2013.

grid. Note, too, that the *Ascending* keyword has been selected for the *Name* column. That selection tells Access to present the data in alphabetical order by name.

If you now click the red exclamation point labeled *Run* in the *Results* section of the ribbon, the result shown in Figure AE5b-22 will appear. Notice that only PROSPECT rows that have at least one WORK row are shown. By default, for queries of two or

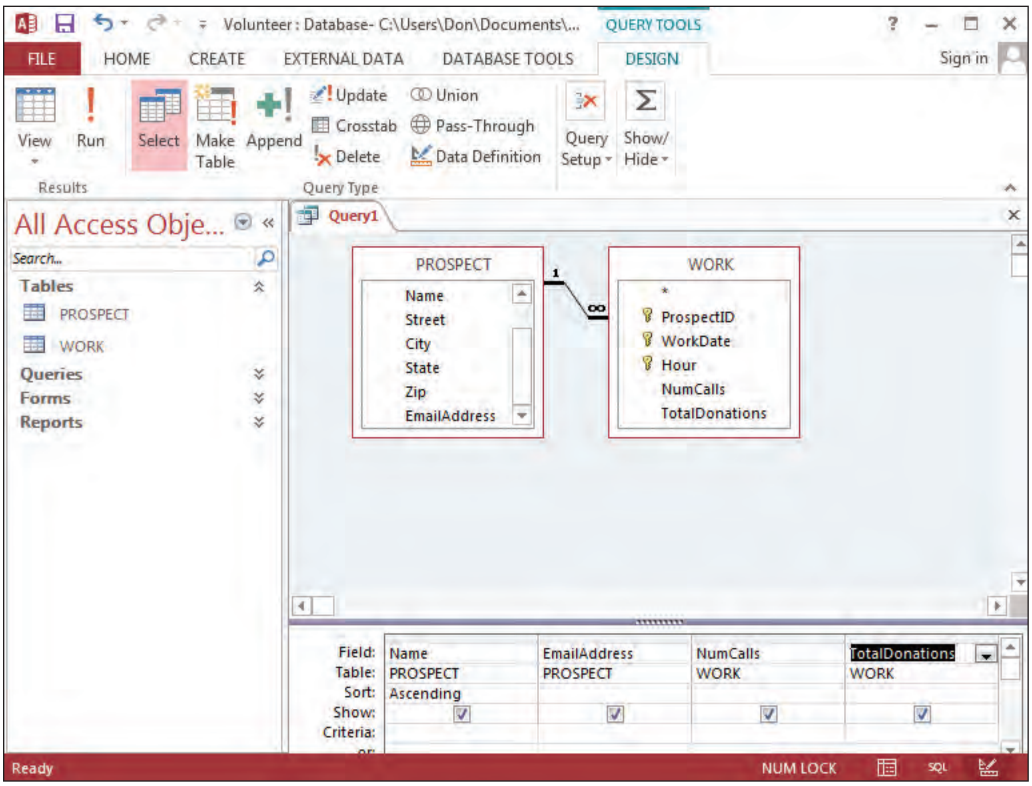


Figure AE5b-21
Creating a Query, Step 3

Source: Microsoft Access 2013.

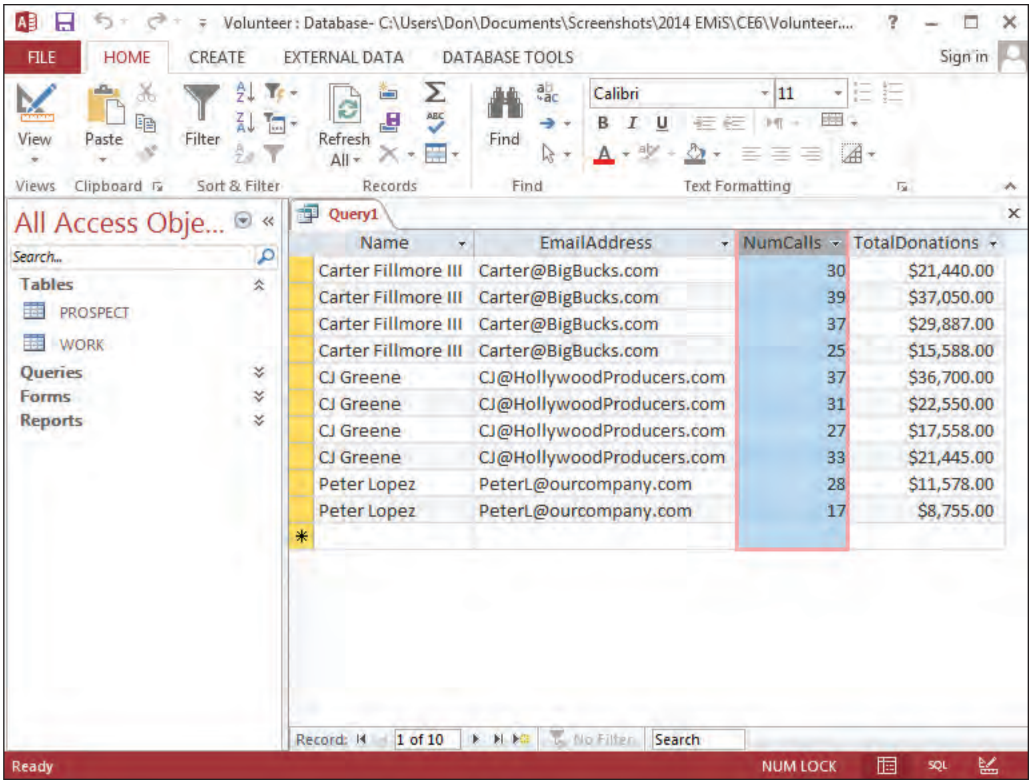


Figure AE5b-22

Results of *TotalDonations* Query

Source: Microsoft Access 2013.

more tables Access (and SQL) shows only those rows that have value matches in both tables. Save the query under the name *NameAndDonationQuery*.

Queries have many useful purposes. For example, suppose we want to see the average dollar value of donations generated per hour of work. This query, which is just slightly beyond the scope of this chapter extension, can readily be created using either the Access graphical tool or SQL. The results of such a query are shown in Figure AE5b-23. This query processes the *NameAndDonationQuery* query just

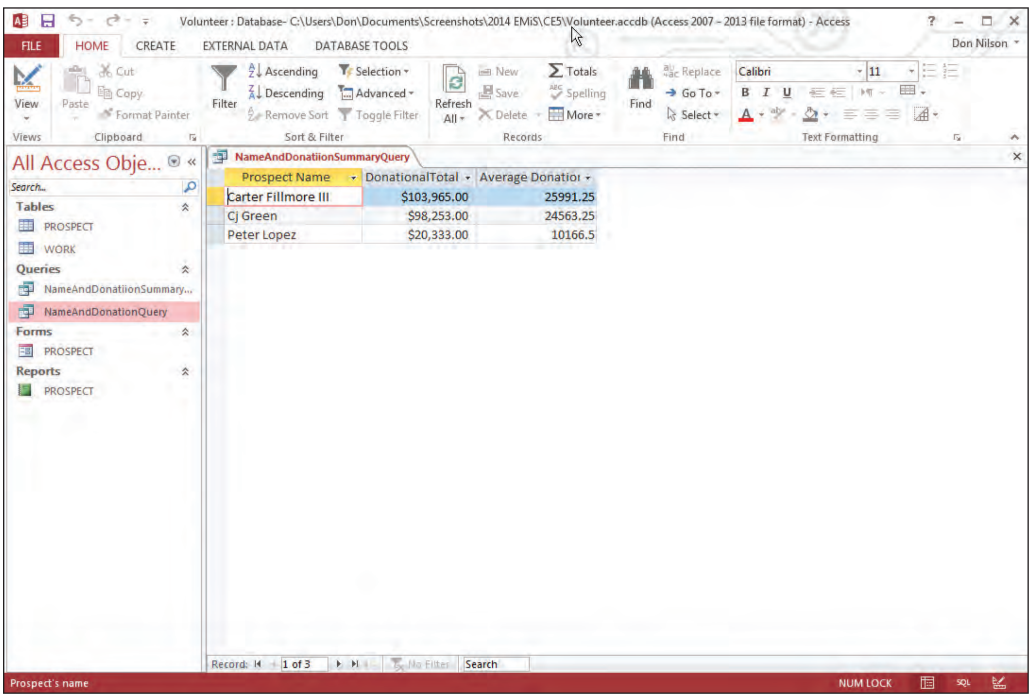


Figure AE5b-23

Result of More Advanced Query

Source: Microsoft Access 2013.

created. Again, if you take a database class, you will learn how to create queries like this and others of even greater complexity (and utility).

Q5 How Do You Create a Report?

You can create a report using a process similar to that for forms, but the report won't include the WORK data. To create a report with data from two or more tables, we must use the Report Wizard. Click the *CREATE* tab, and then in the Reports section click *Report Wizard*.

Now, click *Table: PROSPECT* in the *Table/Queries* combo box, highlight *Name* in the *Available Fields* list, and click the single chevron (>) to add *Name* to the report. You will see the display shown in Figure AE5b-24.

Using a similar process, add *EmailAddress*. Then select *Table: WORK* in the *Table/Queries* combo box and add *WorkDate, Hour, NumCalls*, and *TotalDonations*. Click *Finish*, and you will see the report shown in Figure AE5b-25. (By the way, we are skipping numerous options that Access provides in creating reports.)

We will consider just one of those options now. Suppose we want to show the total donations that a prospect has obtained, for all hours of his or her work. To do that, right-click the *PROSPECT* tab, and then click *Design View*. Your report will appear as shown in Figure AE5b-26. (If it does not appear like this, click *View, Design View* in the ribbon.)

In the ribbon, click *Group & Sort* in the *Grouping & Totals* section. In the bottom of the form, under Group, Sort, and Total click *More*, and then click the down arrow next to the phrase *with no totals*. Next, select *TotalDonations* from the *Total On* box, and then check *Show Grand Total* and *Show subtotal in group footer*, as illustrated in Figure AE5b-27.

Click the *Report* icon in the *View* section of the ribbon, and you will see the report shown in Figure AE5b-28. The only remaining problem is that the label *NumCalls* is

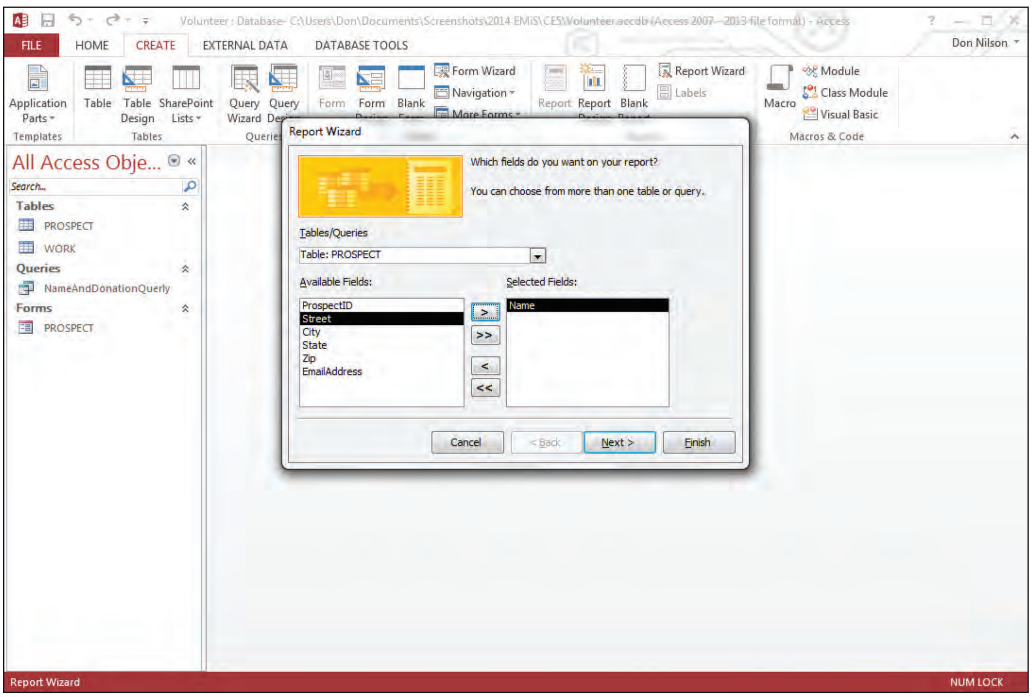


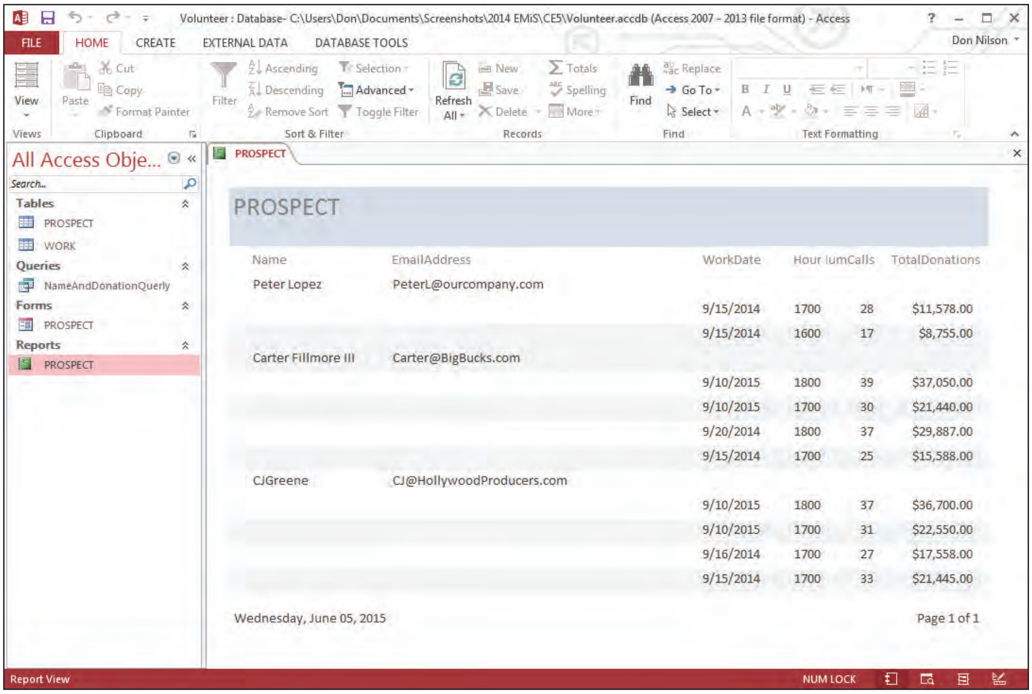
Figure AE5b-24
Selecting Data to Show in a Report

Source: Microsoft Access 2013.

Figure AE5b-25

Report on Donations, by Prospect

Source: Microsoft Access 2013.

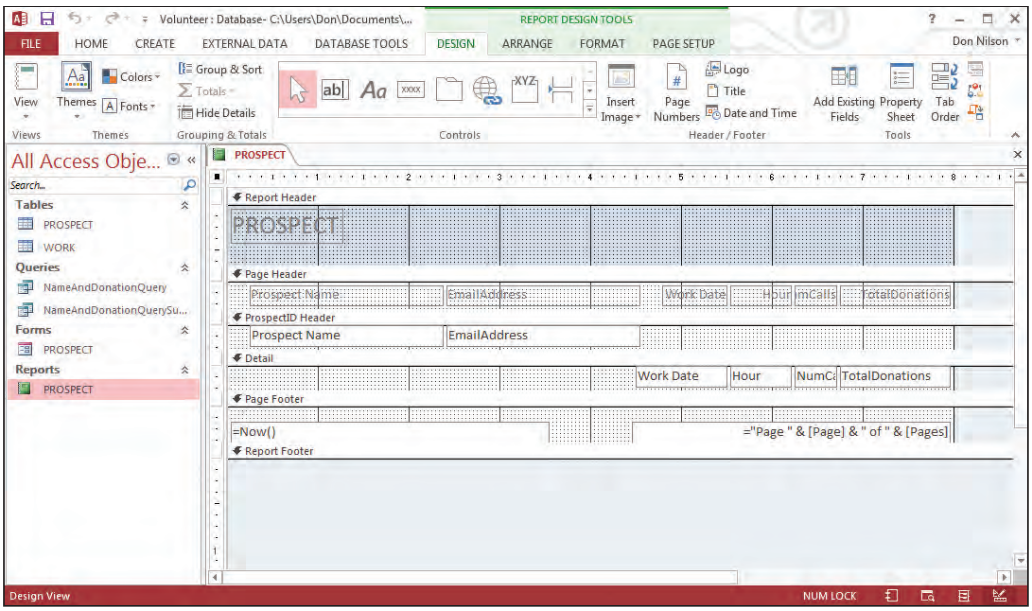


cut off. We need to expand the box that contains this value. To do so, select *Layout View* from *View* in the ribbon, click *Date*, and then slide it slightly to the left. Do the same with *Hour*. Then expand *NumCalls* until you can see all of the label, as shown in Figure AE5b-29. Click *Report View* in *View*, and your report should appear as shown in Figure AE5b-30.

Figure AE5b-26

Report Design View

Source: Microsoft Access 2013.



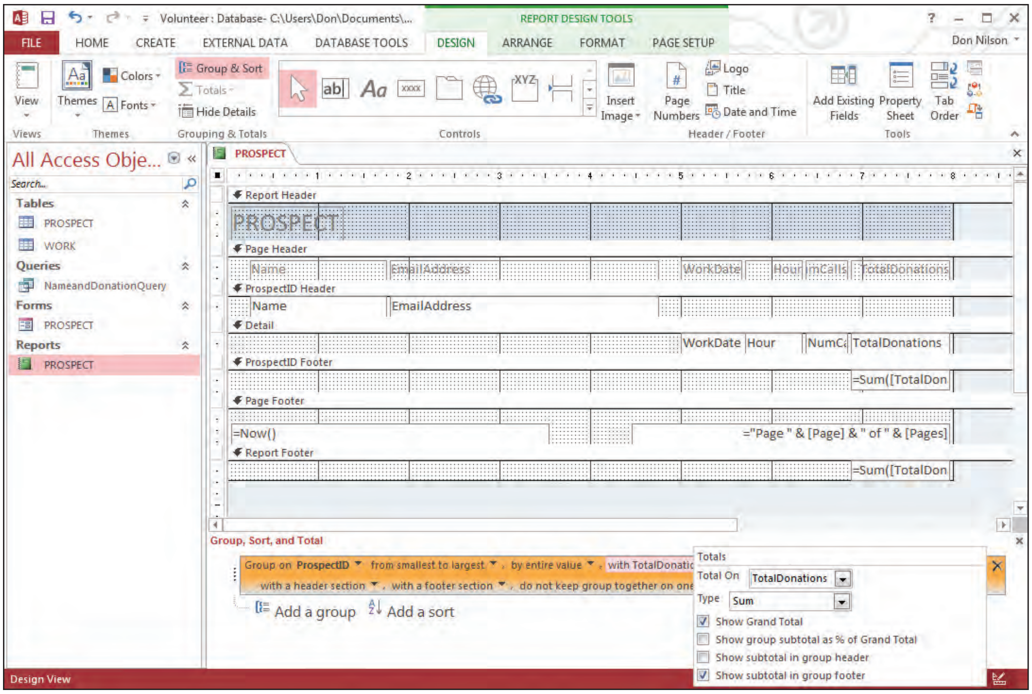


Figure AE5b-27

Creating a Sum of *TotalDonations* for Each Prospect

Source: Microsoft Access 2013.

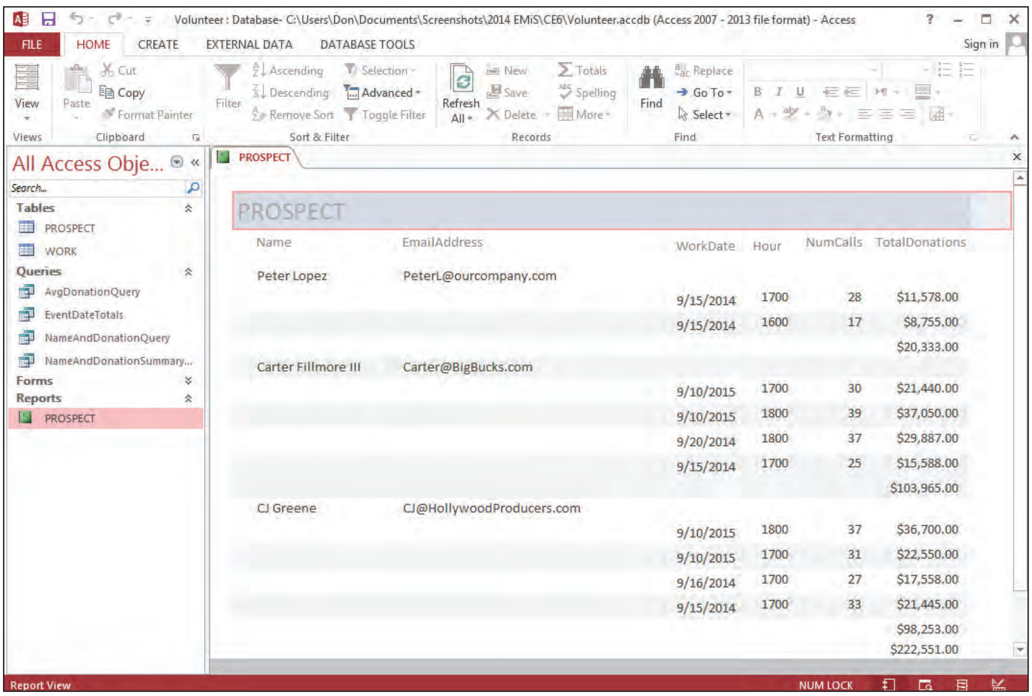


Figure AE5b-28

Report with Sum of *TotalDonations*

Source: Microsoft Access 2013.

Figure AE5b-29

Increasing the Size of the *NumCalls* Field

Source: Microsoft Access 2013.

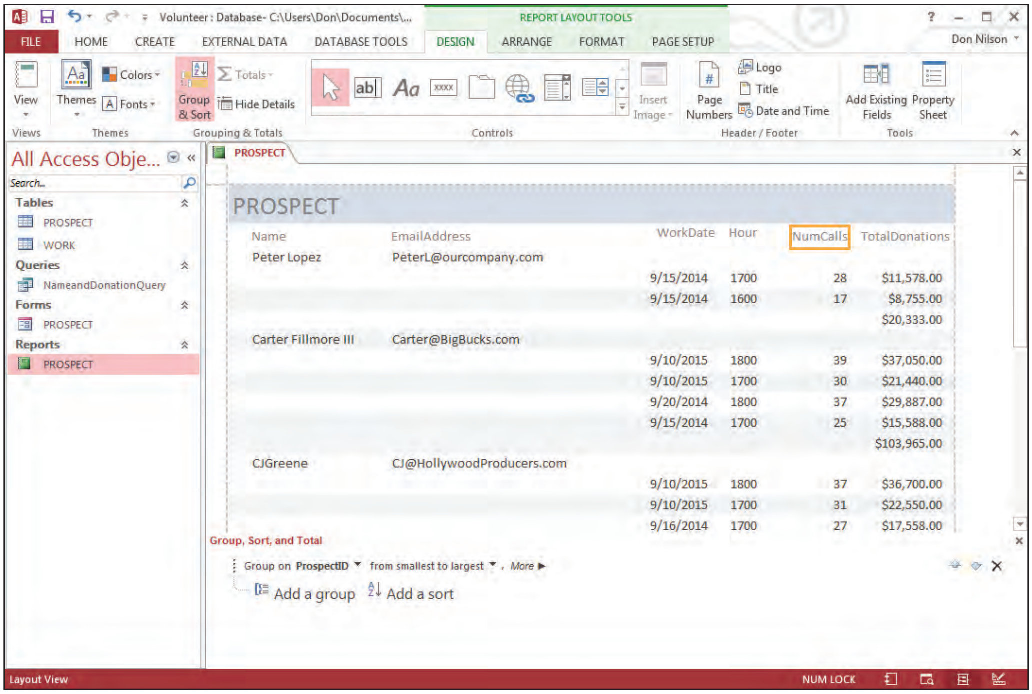
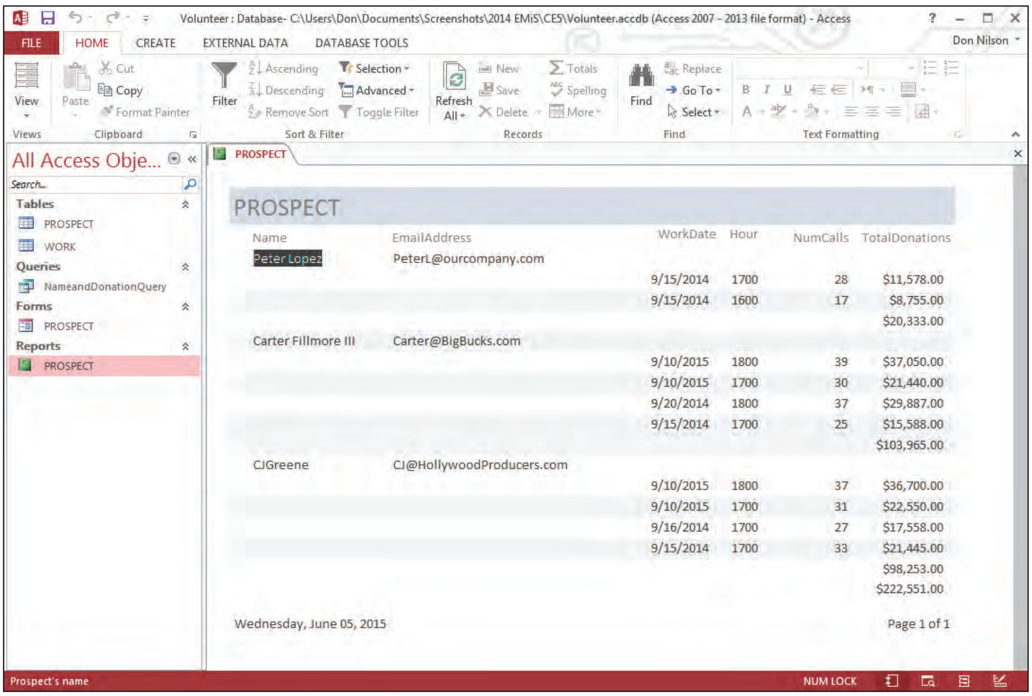


Figure AE5b-30

Final Version of Report

Source: Microsoft Access 2013.



Active Review

Use this Active Review to verify that you understand the ideas and concepts in this chapter extension's study questions.

For this Active Review, assume that you are creating a database application with the following two tables:

```
CUSTOMER (CustomerID, Name, Email)
CONTACT (CustomerID, Date, Subject)
```

Q1 How do you create tables?

Open Access and create a new database with a name of your choosing. Create the CUSTOMER and CONTACT tables. Assume the following data types:

Attribute (Field)	Data Type
CustomerID (in CUSTOMER)	AutoNumber
Name	Text (50)
Email	Text (75)
CustomerID (in CONTACT)	Number (long integer)
Date	Date
Subject	Text (200)

Add Description entries to the Field definitions that you think are appropriate.

Q2 How do you create relationships?

Open the *Relationships* window and create a relationship from CUSTOMER to CONTACT using the CustomerID attribute. Click all of the check boxes. Enter sample data. Add at least five rows to CUSTOMER and at least seven rows to CONTACT. Ensure that some CUSTOMER rows have no matching CONTACT rows.

Q3 How do you create a data entry form?

Open the default data entry form for the CUSTOMER table. Click the CUSTOMER rows to display the related CONTACT data. Now use the *Form* tool to create a data entry form. Navigate through that form to see that the CONTACT rows are correctly connected to the CUSTOMER rows. Adjust spacing as you deem appropriate while removing the CustomerID field from the CUSTOMER section.

Q4 How do you create queries using the query design tool?

Create a query that displays Name, Email, Date, and Subject. Sort the results of Name in alphabetical order.

Q5 How do you create a report?

Use the Report Wizard to create a report that has Name, Email, Date, and Subject. View that report. Add a group total for each CUSTOMER that counts the number of contacts for each customer. Follow the procedure shown, except instead of selecting Sum for Type choose *Count Records* instead.

Using Your Knowledge

AE5b-1. Answer question AE5a-2 at the end of Chapter Extension 5a (page 171). Use Access to implement your database design. Create the tables and add sample data. Create a data entry form that shows teams and the equipment they have checked out. Verify that the form correctly processes new checkouts, changes to checkouts, and equipment returns. Create a report that shows each team, the items they have checked out, and the number of items they have checked out. (Use *Count Records* as explained in Active Review Q5.)

AE5b-2. Answer question AE5a-3 at the end of Chapter Extension 5a (page 171). Create an Access database for the CUSTOMER and RENTAL tables only. Create the tables and add sample data. Create a data entry form that shows customers and all of their rentals (assume customers rent bicycles more than once). Verify that the form correctly processes new rentals, changes to rentals, and rental returns. Create a report that shows each customer, the rentals they have made, and the total rental fee for all of their rentals.

There are no Assisted-graded writing questions in this chapter extension.