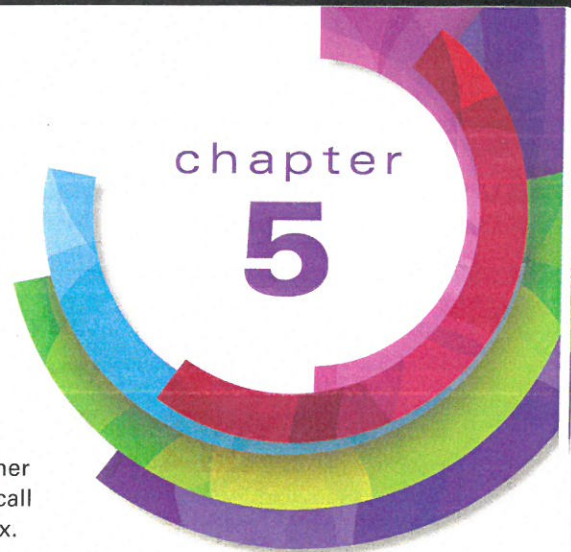


Database Processing



chapter 5

It's Friday night, and Camillia (Cam) Forset is on her way to an art show opening. She gets an urgent call from Jess Denkar, the head of security at PetroTex. PetroTex is a large oil refinery based in Texas and one of Falcon Security's biggest industrial clients. Jess is looking for information that could help him find out who stole almost \$75,000 worth of custom piping and copper wiring.

Cam made sure Jess had her personal cell phone number and told him to call her anytime—day or night. She knows it's important to show Jess that the money PetroTex spends on Falcon's services is worth it. She immediately calls Toshio Sato, director of IT Services, and tells him to come back into the office. She also sends a text message to CEO Mateo Thomas and CFO Joni Campbell.

"Have you found anything yet?" Joni asks. She quickly sets down her purse at a nearby workstation and begins hovering behind Toshio.

"Not yet—we're working as fast as we can," Cam replies tersely. Cam wants to focus on helping Toshio find the correct security footage, not discuss the importance of the PetroTex account to Falcon Security.

"Is there anything I can do to help?" Joni wonders.

"No, we're just trying to find the right footage. There's a lot of it to review," Cam sighs.

"What's the problem? Why do we have to look through so much footage?"

Toshio is tempted to tell Joni that "we" aren't looking through anything. He's the one doing the searching. But he holds his tongue. "Well, the problem is we have footage of dozens of different buildings at PetroTex, from several different drones, over about a 2-week period. Tracking down the footage of exactly when the equipment was stolen

means we have to search through hundreds of different video files. This could take all night," he answers matter-of-factly.

"There's got to be a faster way to do this. Can't we just search for the footage somehow?" Joni says.

"No," Toshio says calmly. "We don't have a way to track the data about the videos. The video files are sequentially numbered and stored in directories for a specific company. We can also see the date and time the video file was created, but there are several different drones. It's just..."

Cam interrupts Toshio to try to keep him from getting distracted. "Toshio and I have talked about creating a database to track all of our video files. We've just been busy trying to automate the data collection and storage process."

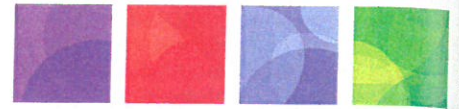
"Well, how long would it take to make it? How much would it cost?" Joni asks.

This
could happen
to you





STUDY QUESTIONS



- Q5-1** WHY DO YOU NEED TO KNOW ABOUT DATABASES?
- Q5-2** WHAT IS A DATABASE?
- Q5-3** WHAT IS A DATABASE MANAGEMENT SYSTEM (DBMS)?
- Q5-4** HOW DO DATABASE APPLICATIONS MAKE DATABASES MORE USEFUL?
- Q5-5** HOW CAN FALCON SECURITY BENEFIT FROM A DATABASE SYSTEM?
- Q5-6** WHAT ARE NONTRADITIONAL DBMS PRODUCTS?

How does the **knowledge** in this chapter help **you**?



Source: cheskyw/123RF

"We don't have a way to track the data about the videos."

"We're not sure. We don't even know which database management system we'd use. Toshio and I have both used Microsoft Access, but Toshio mentioned something called MongoDB that might be better for tracking video files."

Toshio pauses the video he's reviewing and starts pointing at a listing of video files in a directory. "Instead of searching through all of these manually, I could just specify characteristics about each video like company name, building name, date, time, and elevation. It would return the URLs for videos matching those characteristics. We'd know right where they are on our file server."

"Sounds good to me. Anything to keep us from spending our Friday nights looking through video footage. Let's do it," Joni quips.

Cam tries to get things back on track. "OK, great, we'll put it on our list. There's no doubt that a database will keep us from having to search through all this footage. For now, though, we should probably focus on finding the footage for Jess. We've got a lot to go through tonight."

CE

Optional Extensions for this chapter are • CE5: Database Design 467 • CE6: Using Microsoft Access 2013 481 • CE7: Using Excel and Access Together 501



Q5-1 WHY DO YOU NEED TO KNOW ABOUT DATABASES?

This chapter and its related chapter extensions introduce you to database technology. This knowledge is important to you as a future business professional. For one, databases are everywhere. Although you may not realize it, you access dozens, if not hundreds, of databases every day. Every time you make a cell phone call, log on to the Internet, or buy something online using a credit card, applications behind the scenes are processing numerous databases. Use Snapchat, Facebook, Twitter, or LinkedIn, and again applications are processing databases on your behalf. Google something, and yet again dozens of databases are processed to obtain the search results.

REASONS FOR LEARNING DATABASE TECHNOLOGY

As a user, you need know nothing about the underlying technology. From your perspective, “it just works,” to quote the late Steve Jobs. However, as a business professional in the 21st century, it’s a different story. You need the knowledge of this chapter and the related extensions for four principal reasons:

1. When you participate in the development of any new business initiative, you need to know if database technology can facilitate your project goals. If so, you need sufficient knowledge to assess whether building that database is akin to building a small shed or is closer to building a skyscraper. Joni, in the opening vignette of this chapter, needs to have some knowledge to assess how hard (and thus how expensive) building that new database will be.
2. Because databases are ubiquitous in commerce, billions of bytes of data are stored every day. You need to know how to turn that data into a format from which you can construct useful information. To that end, you might use one of many different graphical tools to query that data. Or, to become truly proficient, you might learn SQL, an international standard language for querying databases. Many business professionals have done just that. See the So What? feature of this chapter on page 161.
3. Business is dynamic, and information systems must adapt. Often, such adaptation means that the structure of the database needs to be changed. Sometimes, it means that entirely new databases must be created. As you will learn in this chapter, only the users, such as yourself, know what and how details should be stored. You may be asked to evaluate a data model like those described in Chapter Extension 5.
4. Finally, you might someday find yourself or your department in a material mess. Maybe you don’t know who has which equipment, or where certain tools are located, or what’s really in your supply closet. In that case, you might choose to build your own database. Unless you’re an IS professional, that database will be small and relatively simple, but it can still be very useful to you and your colleagues. Case Study 5 on page 179 illustrates one such example.

So let’s get started by discussing the purpose of a database.

WHAT IS THE PURPOSE OF A DATABASE?

The purpose of a database is to help people keep track of things. When most students learn that, they wonder why we need a special technology for such a simple task. Why not just use a list? If the list is long, put it into a spreadsheet.

In fact, 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.

Figure 5-1
A List of Student Grades
Presented in a Spreadsheet

Source: Access 2013, Microsoft Corporation

Student Name	Student Number	HW1	HW2	MidTerm	HW3	HW4	Final
BAKER, ANDREA	1325	88	100	78			
FISCHER, MAYAN	3007	95	100	74			
LAU, SWEE	1644	75	90	90			
NELSON, STUART	2881	100	90	98			
ROGERS, SHELLY	8009	95	100	98			
TAM, JEFFREY	3559	100	88	88			
VALDEZ, MARIE	5265	80	90	85			
VERBERRA, ADAM	4867	70	90	92			

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

Instead, the professor wants a form like that in Figure 5-2. With it, the professor can record student grades, emails, and office visits all in one place. A form like the one in Figure 5-2 is difficult, if not impossible, to produce from a spreadsheet. Such a form is easily produced, however, from a database.

The key distinction between Figures 5-1 and 5-2 is that the data in Figure 5-1 is about a single theme or concept. It is about student grades only. The data in Figure 5-2 has multiple themes;

Chapter Extension 5 explains how to apply this general rule using a design process called **normalization**.

Figure 5-2
Student Data Shown in a
Form from a Database

Source: Access 2013, Microsoft Corporation

STUDENT

Student Name: BAKER, ANDREA
 Student Number: 1325
 HW1: 88
 HW2: 100
 MidTerm: 78

EMAIL

Date	Message
2/1/2016	For homework 1, do you want us to provide notes on our ref
3/15/2016	My group consists of Swee Lau and Stuart Nelson.
* 3/4/2015	

Record: 1 of 2

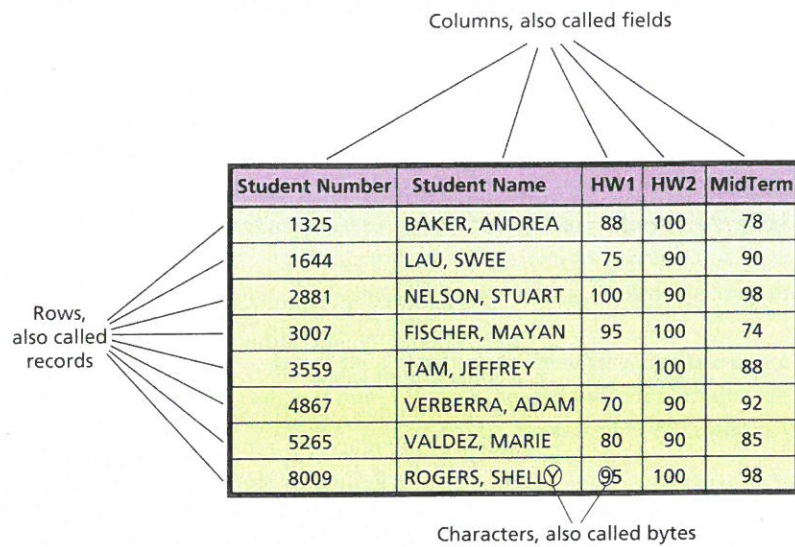
OFFICE VISITS

Date	Notes
2/13/2016	Andrea had questions about using IS for raising barriers to entry.
*	

Record: 1 of 1

Record: 1 of 8

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



it shows student grades, student emails, and student office visits. We can make a general rule from these examples: Lists of data involving a single theme can be stored in a spreadsheet; lists that involve data with multiple themes require a database.

Q5-2 WHAT IS A DATABASE?

A **database** is a self-describing collection of integrated records. To understand the terms in 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. In databases, 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.

Figure 5-4
Hierarchy of Data Elements

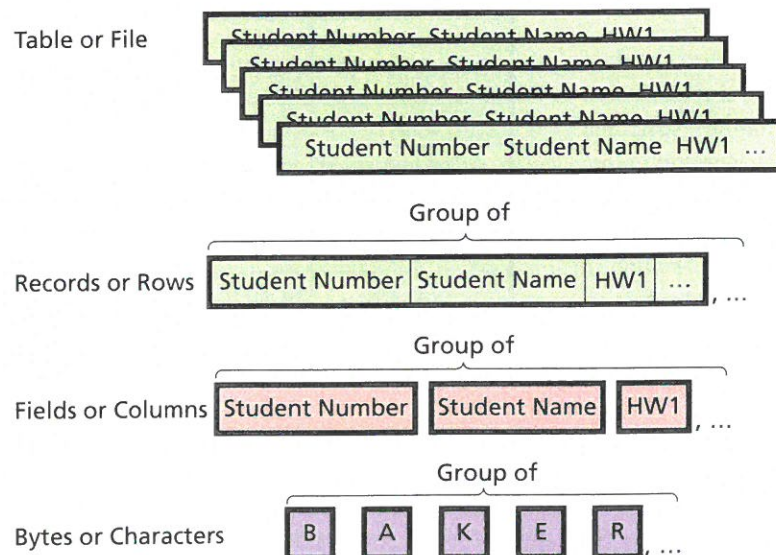
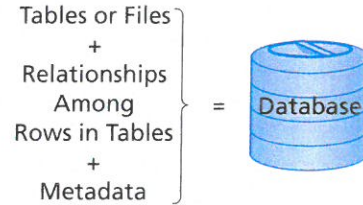


Figure 5-5
Components of a Database



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. 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 describes the structure of the database. By the way, the cylindrical symbol labeled “database” in Figure 5-5 represents a computer disk drive. It represents databases because databases are most frequently stored on disks.

Access is a common database. For more information on Access, see Chapter Extensions 6 and 7. Chapter Extension 7 covers Excel as well.

RELATIONSHIPS AMONG ROWS

Consider the terms on the left-hand 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 sample data from the three tables *Email*, *Student*, and *Office_Visit*. Notice the column named *Student Number* in the *Email* table. That column indicates the row in *Student* to which a row of *Email* is connected. In the first row of *Email*, 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,

Figure 5-6
Example of Relationships Among Rows

Email Table

EmailNum	Date	Message	Student Number
1	2/1/2016	For homework 1, do you want us to provide notes on our references?	1325
2	3/15/2016	My group consists of Swee Lau and Stuart Nelson.	1325
3	3/15/2016	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	2/13/2016	Andrea had questions about using IS for raising barriers to entry.	1325
3	2/17/2016	Jeffrey is considering an IS major. Wanted to talk about career opportunities.	3559
4	2/17/2016	Adam will miss class Friday due to job conflict.	4867

you will see that the row for Andrea Baker has 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 Figure 5-6. 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** (also called a **primary 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*, you can determine one and only one row in *Student*. 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*, *State*) because a given city name can appear in more than one state.

Student Number is not the key of the *Email* or the *Office_Visit* tables. We know that about *Email* 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* cannot be the key of *Email*.

Nor is *Student Number* a key of *Office_Visit*, 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 *Office_Visit* with the same value of *Student Number*. It just happens that no student has visited twice in the limited data in Figure 5-6.

In both *Email* and *Office_Visit*, *Student Number* is a key, but it is a key of a different table, namely *Student*. Hence, the 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 than the one in which they reside.

Before we go on, 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 like those we're discussing is **relation**.) You'll learn about another kind of database in Q5-5.

METADATA

Recall the definition of database: A database is a self-describing collection of integrated records. The records are integrated because, as you just learned, rows can be linked together by their key/foreign key relationships. Thus, 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 catalog that describes the library's contents. The same idea also pertains to a database. Databases 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 it appears 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 light rectangle drawn

Figure 5-7
Sample Metadata (in Access)

Source: Access 2013, Microsoft Corporation

Field Name	Data Type	Description (Optional)
EmailNum	AutoNumber	Primary key -- values provided by Access
Date	Date/Time	Date and time the message is recorded
Message	Long Text	Text of the email
Student Number	Number	Foreign key to row in the Student Table

Field Properties	
General	Lookup
Format	Short Date
Input Mask	99/99/0000;0;#
Caption	
Default Value	=Now()
Validation Rule	
Validation Text	
Required	Yes
Indexed	No
IME Mode	No Control
IME Sentence Mode	None
Text Align	General
Show Date Picker	For dates

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

around the *Date* row.) 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 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. 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.

Q5-3 WHAT IS A DATABASE MANAGEMENT SYSTEM (DBMS)?

A **database management system (DBMS)** is a program used to create, process, and administer a database. As with operating systems, almost no organization develops its own DBMS. Instead, companies license DBMS products from vendors such as IBM, Microsoft, Oracle, and others. Popular DBMS products are **DB2** from IBM, **Access** and **SQL Server** from Microsoft, and **Oracle Database** from the Oracle Corporation. Another popular DBMS is **MySQL**, an open source DBMS product that is license-free for most applications.¹ Other DBMS products are available, but these five process the great bulk of databases today.

Note that a DBMS and a database are two different things. For some reason, 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 are very different concepts.

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 the new table's metadata into the form.

To modify an existing table—say, 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 5-8 the developer has added a new column called *Response?*. This new column has the data type *Yes/No*, which means that the column can contain only one value—*Yes* or *No*. The professor will use this column to



Not What the Data Says...

Two product managers, Jeremy Will and Neil Town, are arguing about the effectiveness of a September product sales promotion in front of their boss, Sarah Murphy.

SARAH: "So, should we repeat that promotion in October?"

JEREMY: "No way. It was expensive, and there was no increase in sales."

NEIL: "I disagree. Well, wait, I agree there was no increase in sales from August, but if you look at sales history, we have a substantial increase over past Septembers."

JEREMY: "Where'd you get that data?"

NEIL: "Extract from the sales database. Anyway, on average, our sales are up 11 percent from past years. And, even better, most of that increase is from new customers."

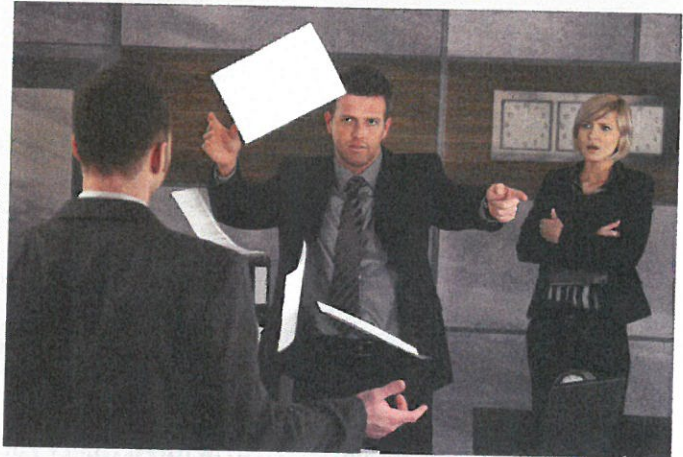
JEREMY: "I don't think so. I called four different sales reps, and they said they can't get any prospects to bite."

NEIL: "Not what the data says. I put it into Access and then did a series of queries. Nineteen percent of sales in September were from new customers."

SARAH: "Amazing. But can you relate that to the campaign?"

NEIL: "Yes, strongly. Turns out of the new customers' sales, almost two-thirds used a coupon."

SARAH: "Neil, put this into a report for me. I want to take it to the executive meeting tomorrow."



Source: StockLite/Shutterstock

- List specific sales data you need to provide answers as Neil does.
- In words, describe how that data needs to be processed in order to produce Neil's responses.
- All of the results that Neil provides are readily produced by Access or by any SQL-processing DBMS. With a basic understanding of SQL, you could write the queries in 5 minutes or less. However, this chapter will give you necessary background, but it won't teach you SQL queries. Consult your university's course catalog and find a course that would teach you the necessary skills. Name that course and explain why you will (or will not) take it.

QUESTIONS

- Do you want to be Jeremy or Neil? Justify your answer.
- What skills and abilities will you need to be Neil?

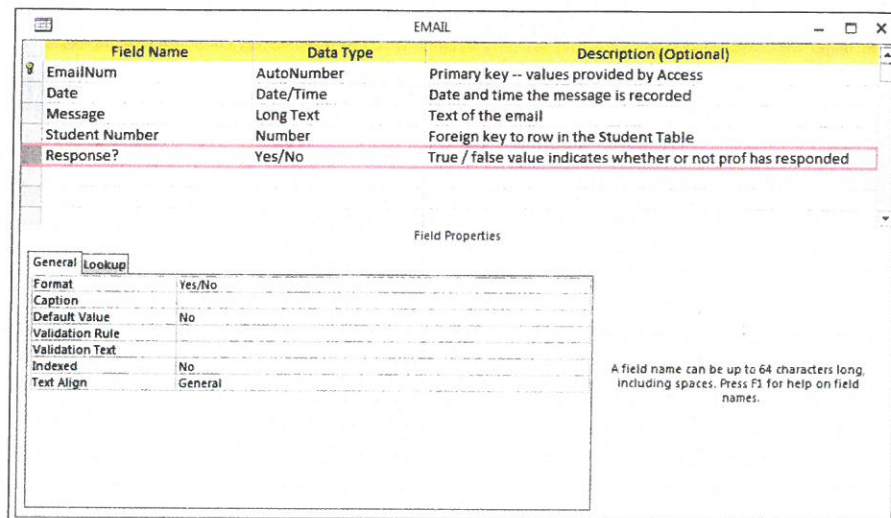
indicate whether he has responded to the student's email. A column can be removed by deleting its row in this table, though doing so will cause any existing data to be lost.

Processing the Database

The second function of the DBMS is to process the database. Such processing can be quite complex, but, fundamentally, the DBMS provides applications for four processing operations: to read, insert, modify, or delete data. These operations are requested in application calls upon the DBMS. From a form, when the user enters new or changed data, a computer

Figure 5-8
Adding a New Column to a
Table (in Access)

Source: Access 2013, Microsoft Corporation



SQL is essential for processing a database, but it can be misused by criminals to steal data. This kind of SQL injection attack is described in the Guide on pages 174–175.

program behind the form calls the DBMS to make the necessary database changes. From a Web application, a program on the client or on the server application program calls the DBMS to make the change.

Structured Query Language (SQL) is an international standard language for processing a database. All five of the DBMS products mentioned earlier accept and process SQL (pronounced “see-quell”) 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);
```

As stated, statements like this one are issued “behind the scenes” by programs that process forms. Alternatively, they can be issued directly to the DBMS by an application program.

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

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 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.

Figure 5-9
Summary
of Database
Administration
(DBA) Tasks

Category	Database Administration Task	Description
Development	Create and staff DBA function	Size of DBA group depends on size and complexity of database. Groups range from one part-time person to small group.
	Form steering committee	Consists of representatives of all user groups. Forum for community-wide discussions and decisions.
	Specify requirements	Ensure that all appropriate user input is considered.
	Validate data model	Check data model for accuracy and completeness.
	Evaluate application design	Verify that all necessary forms, reports, queries, and applications are developed. Validate design and usability of application components.
Operation	Manage processing rights and responsibilities	Determine processing rights/restrictions on each table and column.
	Manage security	Add and delete users and user groups as necessary; ensure that security system works.
	Track problems and manage resolution	Develop system to record and manage resolution of problems.
	Monitor database performance	Provide expertise/solutions for performance improvements.
	Manage DBMS	Evaluate new features and functions.
Backup and Recovery	Monitor backup procedures	Verify that database backup procedures are followed.
	Conduct training	Ensure that users and operations personnel know and understand recovery procedures.
	Manage recovery	Manage recovery process.
Adaptation	Set up request tracking system	Develop system to record and prioritize requests for change.
	Manage configuration change	Manage impact of database structure changes on applications and users.

For important databases, most organizations dedicate one or more employees to the role of database administration. Figure 5-9 summarizes the major responsibilities for this function. You will learn more about this topic if you take a database management course.

Q5-4 HOW DO DATABASE APPLICATIONS MAKE DATABASES MORE USEFUL?

A set of database tables, by itself, is not very useful; the tables in Figure 5-6 contain the data the professor wants, but the format is awkward at best. The data in database tables can be made more useful, or more available for the conception of information, when it is placed into forms like that in Figure 5-2 or other formats.

A **database application** is a collection of **forms, reports, queries**, and application programs² that serves as an intermediary between users and database data. Database applications reformat database table data to make it more informative and more easily updated. Application programs also have features that provide security, maintain data consistency, and handle special cases.

The specific purposes of the four elements of a database application are:

Forms	View data; insert new, update existing, and delete existing data.
Reports	Structured presentation of data using sorting, grouping, filtering, and other operations.
Queries	Search based on data values provided by the user.
Application programs	Provide security, data consistency, and special-purpose processing, e.g., handle out-of-stock situations.

Database applications came into prominence in the 1990s and were based on the technology that was available at that time. Many existing systems today are long-lived extensions to those applications; the ERP system SAP (discussed in Chapter 7) is a good example of this concept. You should expect to see these kinds of applications during the early years of your career.

Today, however, many database applications are based on newer technology that employs browsers, the Web, and related standards. These browser-based applications can do everything the older ones do, but they are more dynamic and better suited to today's world. To see why, consider each type.

TRADITIONAL FORMS, QUERIES, REPORTS, AND APPLICATIONS

In most cases, a traditional database is shared among many users. In that case, the application shown in Figure 5-10 resides on the users' computers and the DBMS and database reside on a server computer. A network, in most cases *not* the Internet, is used to transmit traffic back and forth between the users' computers and the DBMS server computer.

Single-user databases like those in Microsoft Access are an exception. With such databases, the application, the DBMS, and the database all reside on the user's computer.

Traditional forms appeared in windows-like displays like that in Figure 5-2. They serve their purpose; users can view, insert, modify, and delete data with them, but by today's standards, they look clunky. They are certainly a far cry from the modern interface discussed in Chapter 4.

Figure 5-11 shows a traditional report, which is a static display of data, placed into a format that is meaningful to the user. In this report, each of the emails for a particular student is shown after the students' name and grade data. Figure 5-12 shows a traditional query. The user specifies query criteria in a window-like box (Figure 5-12a), and the application responds with data that fit those criteria (Figure 5-12b).

Traditional database applications programs are written in object-oriented languages such as C++ and VisualBasic (and even in earlier languages like COBOL). They are thick applications that

Database technology puts unprecedented ability to conceive information into the hands of users. But what do you do with that information when you find something objectionable? See the Ethics Guide on pages 172–173 for an example case.

Figure 5-10
Processing Environment of a Traditional Database Application

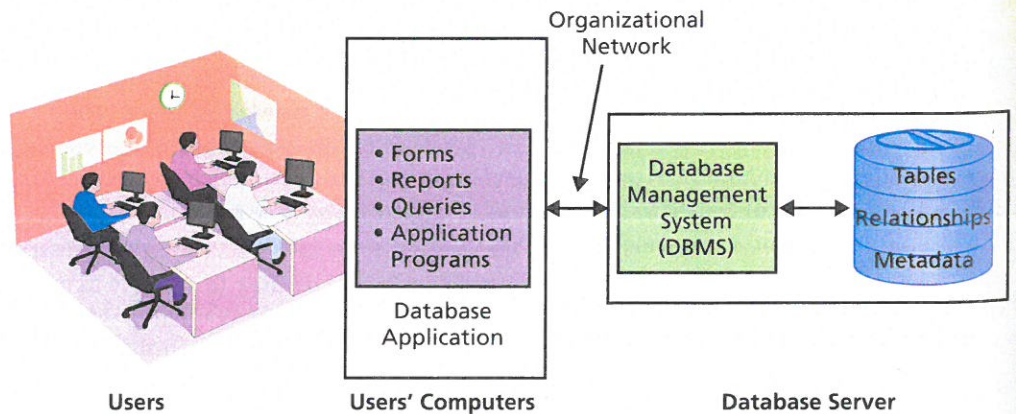


Figure 5-11
Example of a Student Report

Student Homework Progress with Emails			
Student Name	Student Number	HW1	HW2
BAKER, ANDREA	1325	88	100
	Email Date	Message	
	3/15/2016	My group consists of Swee Lau and Stuart Nelson.	
	2/1/2016	For homework 1, do you want us to provide notes on our references?	
LAU, SWEE	1644	75	90
	Email Date	Message	
	3/15/2016	Could you please assign me to a group?	

need to be installed on users' computers. In some cases, all of the application logic is contained in a program on users' computers and the server does nothing except run the DBMS and serve up data. In other cases, some application code is placed on both the users' computers and the database server computer.

As stated, in the early years of your career, you will still see traditional applications, especially for enterprise-wide applications like ERP and CRM (discussed in Chapter 7). Most likely, you will also be concerned, as a user if not in a more involved way, with the transition from such traditional applications into thin-client applications.

THIN-CLIENT FORMS, REPORTS, QUERIES, AND APPLICATIONS

The databases in thin-client applications are nearly always shared among many users. As shown in Figure 5-13, the users' browsers connect over the Internet to a Web server computer, which in turn connects to a database server computer. (Often many computers are involved on the server side of the Internet, as you will learn in Chapter 6.)

As you know, thin-client applications run in a browser and need not be preinstalled on the users' computers. In most cases, all of the code for generating and processing the application elements is shared between the users' computers and the server. JavaScript is the standard language for user-side processing. Languages like C# and Java are used for server-side code, though JavaScript is starting to be used on the server with an open source product named Node.js. (All of this is discussed further in Chapter 6.)

Browser database application forms, reports, and queries are displayed and processed using html and, most recently, using html5, css3, and JavaScript, as you learned in Chapter 4. Figure 5-14 shows a browser form that is used to create a new user account in Office 365. The form's content is dynamic; the user can click on the blue arrow next to *Additional Details* to see more data. Also, notice

Figure 5-12a
Sample Query Form Used to
Enter Phrase for Search

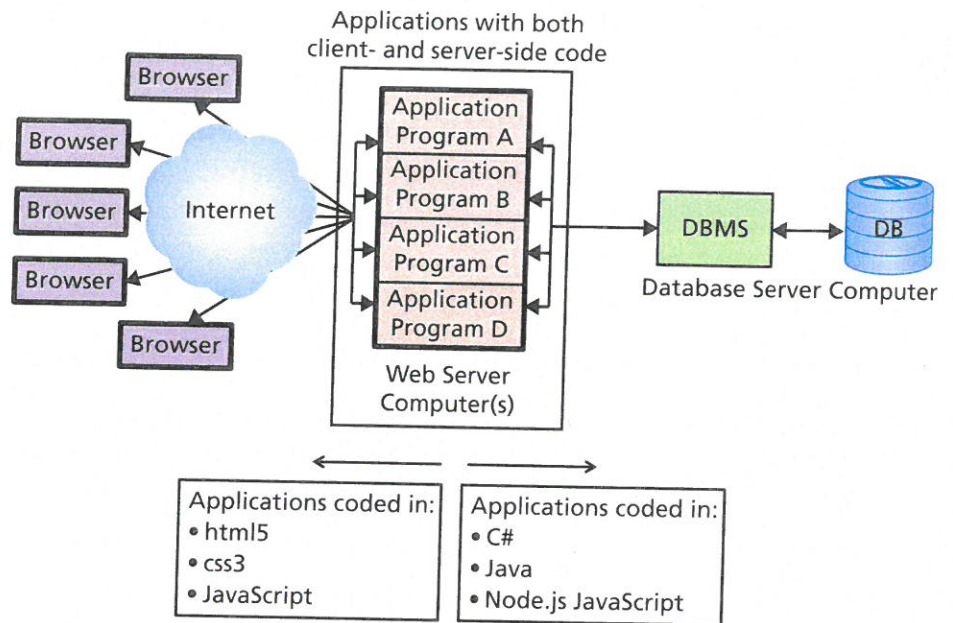
Source: Access 2013, Microsoft Corporation

Figure 5-12b
Sample Query Results of
Query Operation

Source: Access 2013, Microsoft Corporation

Student Name	Date	Notes
BAKER, ANDREA	2/13/2016	Andrea had questions about using IS for raising barriers to entry.

Figure 5-13
Processing Environment of
Browser-Based Database
Applications



the steps in the left-hand side that outline the process that administrators will follow when creating the new account. The current step is shown in color. Compare and contrast this form with that in Figure 5-2; it is cleaner, with much less chrome.

Figure 5-15 illustrates a browser report that shows the content of a SharePoint site. The content is dynamic; almost all of the items can be clicked to produce other reports or take other actions. The user can search in the box in the upper-right-hand corner to find specific items.

Browser-based applications can support traditional queries, but more exciting are **graphical queries** in which query criteria are created when the user clicks on a graphic. Figure 5-16 shows a map of one of the facilities protected by Falcon Security. The user can click on any one of the video icons on the map, and that click will initiate a query to return a list of all the videos available from that location.

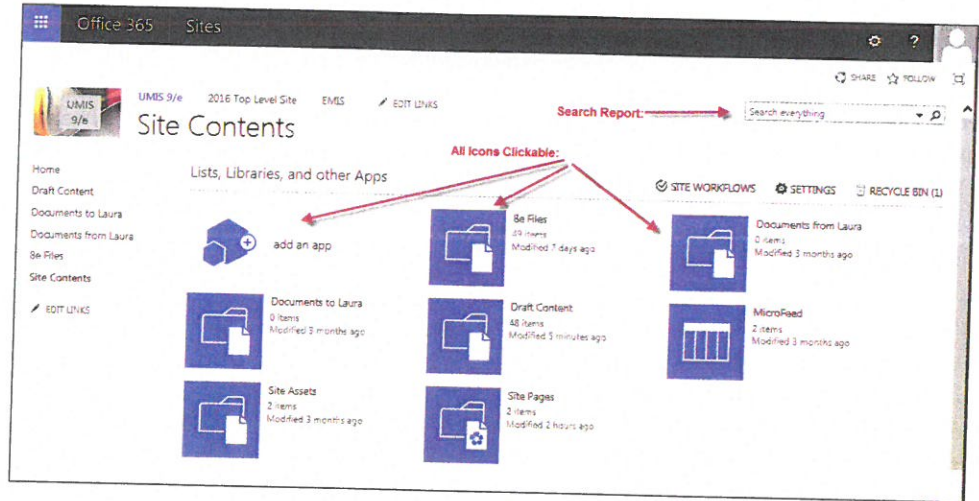
Security requirements are more stringent for thin-client applications than for traditional ones. Most traditional applications run within a corporate network that is protected from the

Figure 5-14
Microsoft Office 365 User
Account Form

Source: Access 2013, Microsoft Corporation

Figure 5-15
Browser Report for
SharePoint Site

Source: Access 2013, Microsoft Corporation

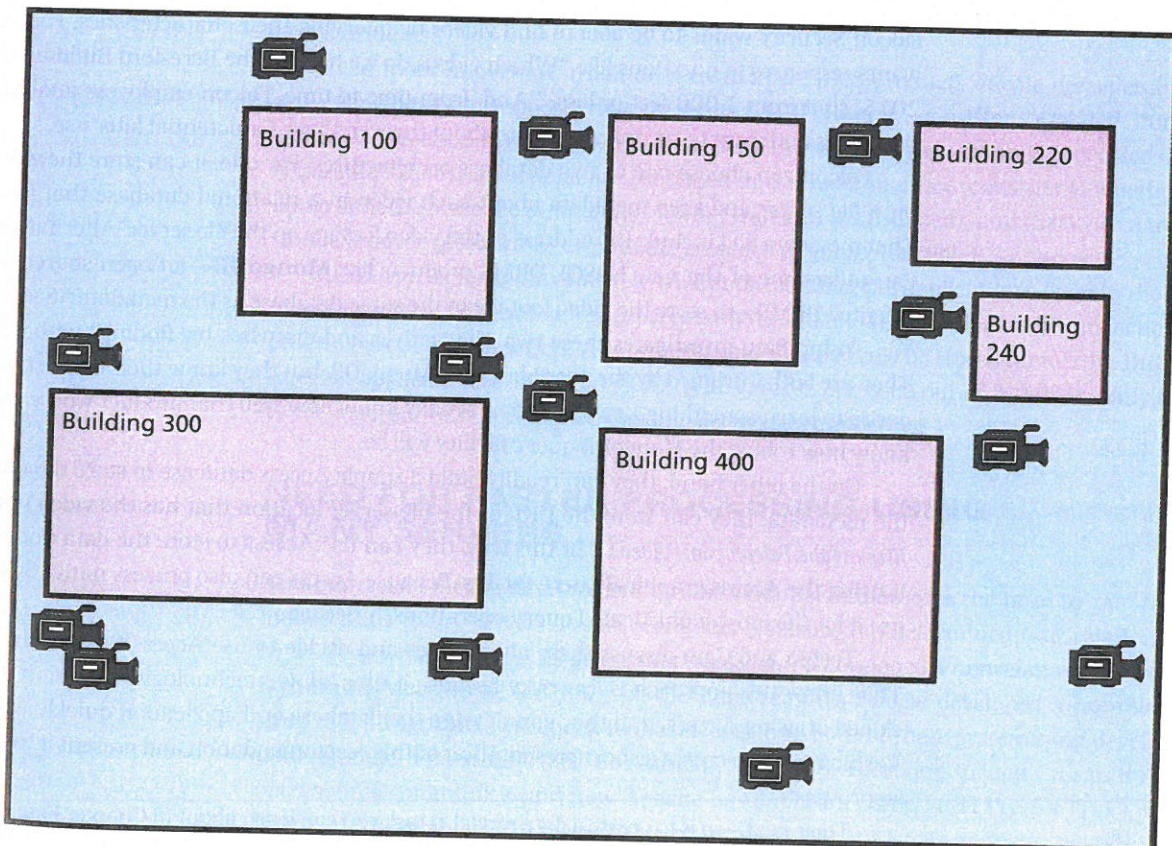


threats common on the Internet. Browser-based applications that are open to the public, over the Internet, are far more vulnerable. Thus, protecting security is a major function for thin-client application programs. Like traditional database application programs, they need to provide for data consistency and to handle special conditions as well. As an example of the need for data consistency, consider the problems introduced by multiuser processing.

MULTIUSER PROCESSING

Most traditional and thin-client applications involve multiple users processing the same database. While such **multiuser processing** is common, it does pose unique problems that you, as a future

Figure 5-16
Graphical Query: User Clicks
on Video Icon to Find All
Videos from That Location



manager, should know about. To understand the nature of those problems, consider the following scenario, which could occur on either type of application.

At a ticket vendor's Web site, two customers, Andrea and Jeffrey, are both attempting to buy the last two tickets to a popular event. Andrea uses her browser to access the site and finds that two are available. She places both of them in her shopping cart. She doesn't know it, but when she opened the order form, she invoked an application program on the vendor's servers that read a database to find that two are available. Before she checks out, she takes a moment to verify with her friend that they still want to go.

Meanwhile, Jeffrey uses his browser and also finds that two tickets are available because his browser activates that same application that reads the database and finds (because Andrea has not yet checked out) that two are available. He places both in his cart and checks out.

Meanwhile, Andrea and her friend decide to go, so she checks out. Clearly, we have a problem. Both Andrea and Jeffrey have purchased the same two tickets. One of them is going to be disappointed.

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 know nothing about one another. Locking brings its own set of problems, however, and those problems must be addressed as well. We will not delve further into this topic here, however.

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 IS department for assistance.

Q5-5 HOW CAN FALCON SECURITY BENEFIT FROM A DATABASE SYSTEM?

Falcon Security wants to be able to find videos by querying their characteristics. For example, it wants responses to questions like "Which videos do we have of the Beresford Building in October 2015, shot from 3,000 feet or less?" And, from time to time, Falcon employees analyze some of the videos and want to record comments about their analyses for potential later use.

Falcon can choose one of two database architectures. For one, it can store the video footage on a file server and keep metadata about each video in a relational database that it can query. That metadata will include the address of the video footage on the file server. Alternatively, Falcon can utilize one of the new NoSQL DBMS products like **MongoDB**—an open source document-oriented DBMS—to store the video footage in the same database as the metadata. (See Q5-6.)

Toshio Sato investigates these two alternatives and discusses his findings with Cam Forset. They are both intrigued by the possible use of MongoDB, but they know their interest is, in part, a desire to learn something new. They don't really know how well that product works, nor do they know how robust the MongoDB query facility will be.

On the other hand, they can readily build a simple Access database to store the metadata. In the metadata, they can store the URL of the file server location that has the video (for example, <https://abc.Falcon.com/Video1>). In this way, they can use Access to store the data and then query it using the Access graphical query facility. Because Access can also process native SQL, they can use it for the most sophisticated query operations, if needed.

Toshio and Cam discuss these alternatives and decide to use Access to store the metadata. They know this approach is less risky because it uses known technology. Also, both of them are skilled at using Access, and they can develop the database and application quickly with less risk. Toshio and Cam create a short presentation of this recommendation and present it to Mateo, who approves it.

Their next step is to create a data model, a task you can learn about in Chapter Extension 5.

Q5-6 WHAT ARE NONTRADITIONAL DBMS PRODUCTS?

The relational model was the single, standard way of processing databases for more than 30 years. Recently, however, that has started to change. Part of the reason is that the major principles of the relational model—fixed-sized tables, representing relationships with foreign keys, and the theory of normalization—came about because of limited storage space and limited processing speeds back in the 1960s and early 1970s.³ At some point, maybe the mid-1990s, these limitations were removed by improved storage and processing technology, and today they do not exist. In other words, the relational model is not needed today.

Furthermore, the relational model was never a natural fit with business documents. For example, users want to store sales orders; they do not want to break up sales orders via normalization and store the data in separate tables. It's like taking your car into a parking garage and having the attendant break it up into pieces, store the pieces in separate piles, and then reassemble the pieces when you come back to get your car. And why? For the efficiency and convenience of the parking garage management. Thus, the primary reason for the relational model's existence is gone, and document piece-making via normalization is no longer necessary.

NEED TO STORE NEW DATA TYPES DIFFERENTLY

There are other reasons for the appearance of new styles of database processing. For one, many organizations, like Falcon Security, want to store new types of data such as images, audio, and videos. Those files are large collections of bits, and they don't fit into relational structures. As you learned in Q5-5, collections of such files still need metadata; we need such data to record when, where, how, and for what purpose the files exist, but we don't need to put them into relational databases just to obtain metadata.

Also, many Internet applications process many, many more transactions against much simpler data than traditional applications do. A tweet has a trivial data structure when compared to the configuration of a Kenworth truck, but there are so many more tweets than truck configurations.

Even more important, traditional relational DBMS products devote considerable code and processing power to support what are termed **ACID** (atomic, consistent, isolated, durable) transactions. In essence, this acronym means that either all of a transaction is processed or none of it is (atomic), that transactions are processed in the same manner (consistent) whether processed alone or in the presence of millions of other transactions (isolated), and that once a transaction is stored, it never goes away—even in the presence of failure (durable).

ACID transactions are critical to traditional commercial applications. Even in the presence of machine failure, Vanguard must process both the sell and the buy sides of a transaction: It cannot process part of a transaction. Also, what it stores today must be stored tomorrow. But, many new Internet applications don't need ACID. Who cares if, one time out of 1 million, only half of your tweet is stored? Or if it's stored today and disappears tomorrow?

NEED FOR FASTER PROCESSING USING MANY SERVERS

Another reason for the development of nonrelational databases is the need to gain faster performance using many servers. A few years ago, Amazon.com determined that relational database technology wouldn't meet its processing needs, and it developed a nonrelational data store called **Dynamo**.⁴ Meanwhile, for many of the same reasons, Google developed a nonrelational data store called **Bigtable**.⁵ Facebook took concepts from both of these systems and developed a third nonrelational data store called **Cassandra**.⁶ In 2008, Facebook turned Cassandra over to the open source community, and now Apache has dubbed it a Top-Level Project (TLP), which is the height of respectability among open source projects.

NONTRADITIONAL DBMS TYPES

These new requirements have led to three new categories of DBMS:

1. **NoSQL DBMS.** This acronym is misleading. It really should be NotRelational DBMS. It refers to new DBMS products that support very high transaction rates processing relatively simple data structures, replicated on many servers in the cloud, without **ACID** transaction support. MongoDB, Cassandra, Bigtable, and Dynamo are NoSQL products.
2. **NewSQL DBMS.** These DBMS products process very high levels of transactions, like the NoSQL DBMS, but provide ACID support. They may or may not support the relational model. Such products are a hotbed of development with new vendors popping up nearly every day. Leading products are yet unknown.
3. **In-memory DBMS.** This category consists of DBMS products that process databases in main memory. This technique has become possible because today's computer memories can be enormous and can hold an entire database at one time, or at least very large chunks of it. Usually, these products support or extend the relational model. SAP HANA is a computer with an in-memory DBMS that provides high-volume ACID transaction support simultaneously with complex relational query processing. Tableau Software's reporting products are supported by a proprietary in-memory DBMS using an extension to SQL.

WILL THESE NEW PRODUCTS REPLACE THE RELATIONAL MODEL?

Does the emergence of these new products mean the death knell for relational databases? It seems unlikely because organizations have created thousands of traditional relational databases with millions of lines of application code that process SQL statements against relational data structures. No organization wants to endure the expense and effort of converting those databases and code to something else. There is also a strong social trend among older technologists to hang onto the relational model. However, these new products are loosening the stronghold that relational technology has enjoyed for decades, and in the future it is likely that many NoSQL, NewSQL, and in-memory databases will exist in commerce.

Furthermore, existing DBMS vendors like Oracle, Microsoft, and IBM will not sit still. With substantial cash and highly skilled developers, they will likely incorporate features of these new categories of DBMS into their existing or new products. Acquisitions of some of the NewSQL startups, in particular, are likely.

Using the vocabulary of Chapter 3, for the first time in more than 20 years, the database software market experienced viable new entrants. So, will Microsoft and Oracle and other DBMS vendors lose some of their market to nontraditional products and vendors? Or will they follow IBM's path? Become less of a vendor of software and more a seller of services supporting open source software such as Cassandra? Or will we soon see companies like Oracle, which is rich with cash, purchasing one of these new companies? Indeed, that may have happened by the time you read this.

WHAT DO NONRELATIONAL DBMS MEAN FOR YOU?

During the early years of your career, many nontraditional databases will be developed, and not just by leading-edge companies like Amazon.com, Google, and Facebook. So, what does that mean to you as a business professional? First, such knowledge is useful; stay abreast of developments in this area. If you were a junior person at Falcon and you went to a meeting today with Toshio and said something like, "Have you thought about using MongoDB for

storing our videos?" you would gain his attention and admiration immediately. You'd likely find yourself on Falcon's key users' committee (or whatever Falcon calls it), and that would be a great career opportunity for you. Also, watch nonrelational DBMS product developments from an investor's perspective. Not all such products will be open source; even if they are, there will be companies that integrate them into their product or service offerings, and those companies may well be good investment opportunities.

If you're interested in IS as a discipline or as a second major, pay attention to these products. You still need to learn the relational model and the processing of relational databases; they will be the bread-and-butter of the industry for many more years. But exciting new opportunities and career paths will also develop around nonrelational databases. Learn about them as well, and use that knowledge to separate yourself from the competition when it comes to job interviews.

Lots of interesting, promising developments are under way!

How does the **knowledge** in this chapter help **you?**

You now understand the purpose of a database and have an introduction to the ways that databases are processed. You also know about new categories of nontraditional DBMS. This knowledge will enable you to be an effective team member when your organization has needs like those at Falcon Security. You can add substantial value to this knowledge by studying Chapter Extensions 5, 6, and 7.

Ethics Guide

Querying Inequality?

MaryAnn Baker works as a data analyst in human relations at a large, multinational corporation. As part of its compensation program, her company defines job categories and assigns salary ranges to each category. For example, the category M1 is used for first-line managers and is assigned the salary range of \$75,000 to \$95,000. Every job description is assigned to one of these categories, depending on the knowledge and skills required to do that job. Thus, the job titles Manager of Customer Support, Manager of Technical Writing, and Manager of Product Quality Assurance are all judged to involve about the same level of management expertise and are all assigned to category M1.

One of MaryAnn's tasks is to analyze company salary data and determine how well actual salaries conform to established ranges. When discrepancies are noted, human relations managers meet to determine whether the discrepancy indicates a need to:

- Adjust the category's salary range;
- Move the job title to a different category;
- Define a new category; or
- Train the manager of the employee with the discrepancy on the use of salary ranges in setting employee compensation.

MaryAnn is an expert in creating database queries. Initially she used Microsoft Access to produce reports, but much of the salary data she needs resides in the organization's Oracle database. At first she would ask the IS Department to extract certain data and move it into Access, but over time she learned that it was faster to ask IS to move all employee data from the operational Oracle database into another Oracle database created just for HR data analysis. Although Oracle provides a graphical query interface like that in Access, she found it easier to compose complex queries directly in SQL, so she learned it and, within a few months, was a SQL expert.

"I never thought I'd be doing this," she said. "But it turns out to be quite fun, like solving a puzzle, and apparently I'm good at it."

One day, after a break, MaryAnn signed into her computer and happened to glance at the results of a query that she'd left running while she was gone. "That's odd," she thought. "All the people with Hispanic surnames have lower salaries than the others." She wasn't looking for that pattern; it just happened to jump out at her as she glanced at the screen.

As she examined the data, she began to wonder if she was seeing a coincidence or if there was a discriminatory pattern within the organization. Unfortunately for MaryAnn's purposes, the organization did not track employee race in its database, so she had no easy way of identifying employees of Hispanic heritage other than reading through the list of surnames. But, as a skilled problem solver, that didn't stop MaryAnn. She realized that many employees having Hispanic origins were born in certain cities in Texas, New Mexico, Arizona, and California. Of course, this wasn't true for all employees; many non-Hispanic employees were born in those cities, too, and many Hispanic employees were born in other cities. This data was still useful, however, because MaryAnn's sample queries revealed that the proportion of



employees with Hispanic surnames who were also born in those cities was very high. "OK," she thought, "I'll use those cities as a rough surrogate."

Using birth city as a query criterion, MaryAnn created queries that determined employees who were born in the selected cities earned, on average, 23 percent less than those who were not. "Well, that could be because they work in lower-pay-grade jobs." After giving it a bit of thought, MaryAnn realized that she needed to examine wages and salaries within job categories. "Where," she wondered, "do people born in those cities fall in the ranges of their job categories?" So, she constructed SQL to determine where within a job category the compensation for people born in the selected cities fell. "Wow!" she said to herself. "Almost 80 percent of the employees born in those cities fall into the bottom half of their salary ranges."

MaryAnn scheduled an appointment with her manager for the next day.

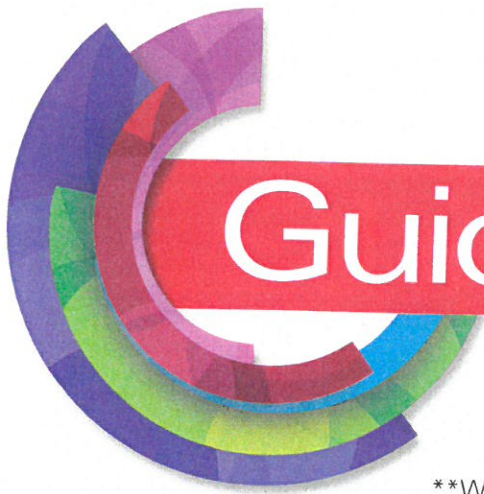


DISCUSSION QUESTIONS

When answering the following questions, suppose that you are MaryAnn:

1. Given these query results, do you have an ethical responsibility to do something? Consider both the categorical imperative (pages 52–53) and the utilitarian (pages 78–79) perspectives.
2. Given these query results, do you have a personal or social responsibility to do something?
3. What is your response if your manager says, "You don't know anything; it could be that starting salaries are lower in those cities. Forget about it."
4. What is your response if your manager says, "Don't be a troublemaker; pushing this issue will hurt your career."
5. What is your response if your manager says, "Right. We already know that. Get back to the tasks that I've assigned you."
6. Suppose your manager gives you funding to follow up with a more accurate analysis, and, indeed, there is a pattern of underpayment to people with Hispanic surnames. What should the organization do? For each choice below, indicate likely outcomes:
 - a. Correct the imbalances immediately.
 - b. Gradually correct the imbalances at future pay raises.
 - c. Do nothing about the imbalances, but train managers not to discriminate in the future.
 - d. Do nothing.
7. Suppose you hire a part-time person to help with the more accurate analysis, and that person is so outraged at the outcome that he quits and notifies newspapers in all the affected cities of the organization's discrimination.
 - a. How should the organization respond?
 - b. How should you respond?
8. Consider the adage "Never ask a question for which you do not want the answer."
 - a. Is following that adage ethical? Consider both the categorical imperative and utilitarian perspectives.
 - b. Is following that adage socially responsible?
 - c. How does that adage relate to you, as MaryAnn?
 - d. How does that adage relate to you, as a future business professional?
 - e. With regard to employee compensation, how does that adage relate to organizations?





Guide

Theft by SQL Injection

****Warning**** You are about to learn a technique for compromising an information system called *SQL injection*. Do not try it on existing systems. SQL injection attacks leave log entries with your IP address attached. Attempting SQL injection on a system without permission is illegal. You can be identified, tracked, and charged. Felony hacking convictions are not resume builders.

SQL injection is a popular way to steal data because it can be done from anywhere in the world. You don't even need to physically enter the target country. You need some smart people with time to invest and a couple modest computers. From a criminal's point of view it's a low-risk and high-reward proposition.

SQL injection is a criminal attack on an information system to illegally extract data from a database. It can add or delete data, drop tables and their data, and even shut down an information system. And, because it can be done from anywhere in the world, criminals can rob from countries that don't extradite criminals, such as Russia, China, North Korea, and others.

Criminals have caught on to theft-by-SQL-injection. Imperva[®], an enterprise data security firm, listed the following key findings in its 2013 Imperva Web Application Attack Report:⁷

1. Retailers suffer two times as many SQL injection attacks as other industries.
2. Most Web applications receive four or more Web attack campaigns per month, and others are constantly under attack (176 out of 180 days).
3. One Web site received 94,057 SQL injection attack requests in one day.

Let those numbers sink in: Your corporate Web site is likely being attacked on a regular basis.

How Does SQL Injection Work?

SQL injection, as it sounds, is a way of inserting your own SQL code into someone else's information system. To understand this, consider what happens when you normally log in to a Web site. You enter your username (**JohnDoe001**) and a password (**password1234**) and then press the Enter key.

In a site that is vulnerable to SQL injection, the following SQL statement is sent to the Web site's DBMS:

```
SELECT * FROM Users WHERE username=' JohnDoe001'  
AND password='password1234' ;
```

If the username and password are both correct, you'll be allowed in. The "injection" part of SQL injection happens when you enter in *unexpected* text into that Web form. You enter text into the login form that changes the way the SQL statement is processed.

Username:	<input type="text" value="JohnDoe001"/>
Password:	<input type="text" value="password1234"/>



Instead of entering a real username and password, put in a random username (in this case, we kept it **JohnDoe001**), and a malformed, but tricky, statement into the password field (**anything' or 1=1 --**).

Note that the single quote (') in the password changes the SQL statement by enclosing the word "anything" and allowing 1=1 to be included. (The double hyphen indicates the rest of the SQL statement, which is not shown because it's not relevant to this guide.)

```
SELECT * FROM Users WHERE username='JohnDoe001'
AND password='anything' or 1=1 --';
```

The word "anything" will *not* match the correct password in the database, but because "or 1=1" was included the resulting comparison will always be "true." This is because 1=1 is true, and only one side of the comparison needs to be true if "or" is included. This SQL statement will enable you to bypass the login screen and gain access to the system. Similar malformed SQL statements can be used to extract, add, or delete data. There is even software available that largely automates the SQL injection process.

Username:	JohnDoe001
Password:	anything' or 1=1 --

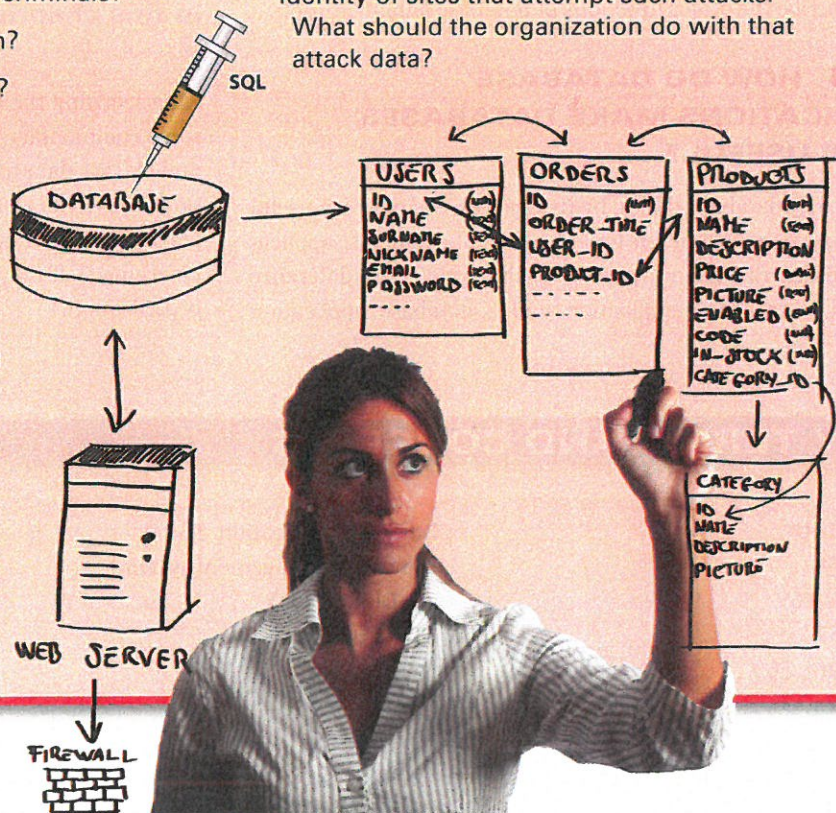
SQL injection can readily be prevented. The particular techniques are beyond the scope of this text, but they come down to never writing computer programs to append user-entered data to a SQL statement. Instead, the users' data is passed to a program controlled by the DBMS that inspects that user-entered data and then uses it without changing any SQL code.⁸

Unfortunately, not all companies take the time to protect themselves from SQL injection. Sony Corp. lost more than 100 million accounts to SQL injection attacks in 2011. In 2014, two U.S. Navy systems administrators on a nuclear aircraft carrier used SQL injection to get the private data of 220,000 sailors. They said they did it out of "boredom."

DISCUSSION QUESTIONS

1. Why is data theft attractive to criminals?
2. How common is SQL injection?
3. How does SQL injection work?
4. What can an attacker do to a database using SQL injection?
5. How can organizations prevent SQL injection attacks from being successful?
6. If you were a senior manager at an organization that had serious losses due to SQL injection, what would you do about it?
7. Suppose an organization not only prevents SQL injection from success, but also tracks the

identity of sites that attempt such attacks. What should the organization do with that attack data?



Source: Federico Caputo/iStock/ Thinkstock/Getty Images



ACTIVE REVIEW



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

Q5-1 WHAT DO YOU NEED TO KNOW ABOUT DATABASES?

Explain three ways in which you interact with a database each day, even though that database is not directly apparent. Summarize four reasons why a business professional should learn database technology.

Q5-2 WHAT IS A DATABASE?

Define the term *database*. Explain the hierarchy of data and name three elements of a database. Define *metadata*. Using the example of *Student* and *Office_Visit* tables, show how relationships among rows are represented in a database. Define the terms *primary key*, *foreign key*, and *relational database*.

Q5-3 WHAT IS A DATABASE MANAGEMENT SYSTEM (DBMS)?

Explain the acronym DBMS and name its functions. List five popular DBMS products. Explain the difference between a DBMS and a database. Summarize the functions of a DBMS. Define *SQL*. Describe the major functions of database administration.

Q5-4 HOW DO DATABASE APPLICATIONS MAKE DATABASES MORE USEFUL?

Explain why database tables, by themselves, are not very useful to business users. Name the four elements of a database application and describe the purpose of each. Explain the difference between a database application and a database application

program. Describe the nature of traditional database applications. Explain why browser-based applications are better than traditional ones. Name the primary technologies used to support browser-based applications.

Q5-5 HOW CAN FALCON SECURITY BENEFIT FROM A DATABASE SYSTEM?

Describe the two system architecture alternatives available to Falcon Security. Explain the advantages and disadvantages of each. State the alternative they chose and explain why they chose it.

Q5-6 WHAT ARE NONTRADITIONAL DBMS PRODUCTS?

Define *NoSQL data store* and give three examples. Explain how NoSQL will likely be used in organizations and state why learning Microsoft Access is still important to you. Explain what is unusual about the development of these systems. Describe possible consequences of NoSQL on the DBMS product market.

How does the knowledge in this chapter help you?

After learning the concepts presented in this chapter and with some rudimentary knowledge of Access, you'll be able to query and extract data to help you solve the problems your organization faces...or at least you will be able to pinpoint the problems. You also know that new categories of DBMS products are emerging that may be important to the database needs of your organization.

KEY TERMS AND CONCEPTS

Access 160
ACID 170
Bigtable 169
Byte 157
Cassandra 169
Columns 157
Database 157
Database administration 162

Database application 163
Database management system (DBMS) 160
DB2 160
Dynamo 169
Fields 157
File 157
Foreign keys 159


Form 163
Graphical queries 166
In-memory DBMS 170
Key 159
Lost-update problem 168
Metadata 159
MongoDB 168
Multiuser processing 167

MySQL 160
NewSQL DBMS 170
NoSQL DBMS 170
Oracle Database 160
Primary key 159




Query 163
Records 157
Relation 159
Relational database 159
Report 163

Rows 157
SQL Server 160
Structured Query Language (SQL) 162
Table 157

MyMISLab™

To complete the problems with the , go to EOC Discussion Questions in the MyLab.

USING YOUR KNOWLEDGE

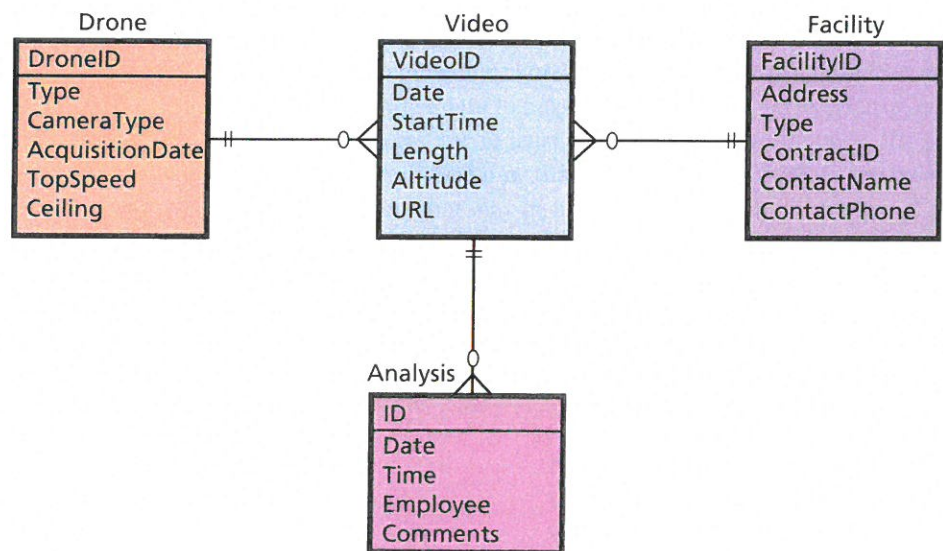
-  **5-1.** What are the steps involved in creating a database? Which step is the most crucial and why?
-  **5-2.** A professor teaches many courses in a university. However, some professors who are the administrative heads (such as department heads) may not teach a course. Some courses are taught jointly by more than one professor. Identify the entities and their relationship in the above scenario. What is the maximum cardinality of this relationship? Can this relationship be implemented in a database? Why or why not?
-  **5-3.** Identify two entities in the data entry form in Figure 5-29. What attributes are shown for each? What do you think are the identifiers?

COLLABORATION EXERCISE 5

The Falcon Security problem is an excellent example of the use of a small database in business. It is also within reach for you to develop as a practice exercise. To do so, work with your team to answer the following questions:

- 5-4.** Study Figure 5-17 to understand the entities and their relationships. Justify each of the cardinalities in this model.
- 5-5.** Working with your team, develop a list of seven queries that together use all of the entities in Figure 5-17.
- 5-6.** Modify the E-R model in Figure 5-17 to include a Contact entity that is related to the Facility entity. Create the relationship, and specify and justify the relationship's cardinalities.
- 5-7.** Discuss the advantages and disadvantages of the model you created in your answer to question 5-6 and the model in Figure 5-17.
- 5-8.** Transform the data model in Figure 5-17 into a relational database design. *Hint:* Create a table for each entity and relate those tables.
- 5-9.** Create an Access database for your design in question 5-8.
- 5-10.** Fill your database with sample data. Because you do not have files on a server, leave the URL column blank.
- 5-11.** A big data analytics organization has launched a connecting program, around the UK to train faculty members across 500 campuses. Every year at least 20 faculty members from each campus need to register for this program. The faculty members will be trained on the big data analytics and other emerging technologies. As a coordinator of this program, you need to track faculty members, their campuses and the technology they have registered for in the current term. You have decided to represent this situation with an ER diagram. Hence,
- Identify various entities in this given scenario.
 - After identification of entities, find out the attributes of the identified entities.
 - The next task is to list the identifiers to be used in the final ER diagram.
 - Before going for the depiction of the given scenario using ER Diagram, figure out the relationship among identified the entities.
 - After performing the above steps, make an ER Diagram to depict your findings.

Figure 5-17
E-R Diagram for Falcon
Security's Database



CASE STUDY 5

Searching for classic and vintage car parts ...

Shane Ashley-Carter is a highly qualified and experienced mechanic, having served apprenticeships with both British luxury car marqueses Jaguar and Bentley in the early part of his career, which now spans more than 20 years. He is a typical of luxury car mechanic whose expertise and experience is sought after in the United Kingdom and world-wide, including the Middle-East, Africa, Asia, Australia, and the United States. Shane specializes in classic and vintage models as well those in the current ranges. In his spare time, he also renovates and repairs interior fixtures and fittings, and exterior body panels at his workshop in Oxfordshire, England. (See <http://www.a-cl.co.uk>). During the months of June, July, and August, he works on repairs and restorations of car exteriors and interiors when the weather is reliably warm and dry, and during the remainder of the year he repairs and replaces mechanical components.

Before the global financial crisis of 2008, he did a brisk trade in acquiring, restoring, renovating, reselling, and retrofitting car parts to classic and vintage Jaguars and Bentleys, especially of the former because of the global brand recognition that resulted from the famous Oxford-based detective television series, Inspector Morse. During this time, he rarely needed to maintain significant stocks of the various car components as there was a balance between supply and demand. However, during the period leading up to 2010, he found that his stock of car parts increased due

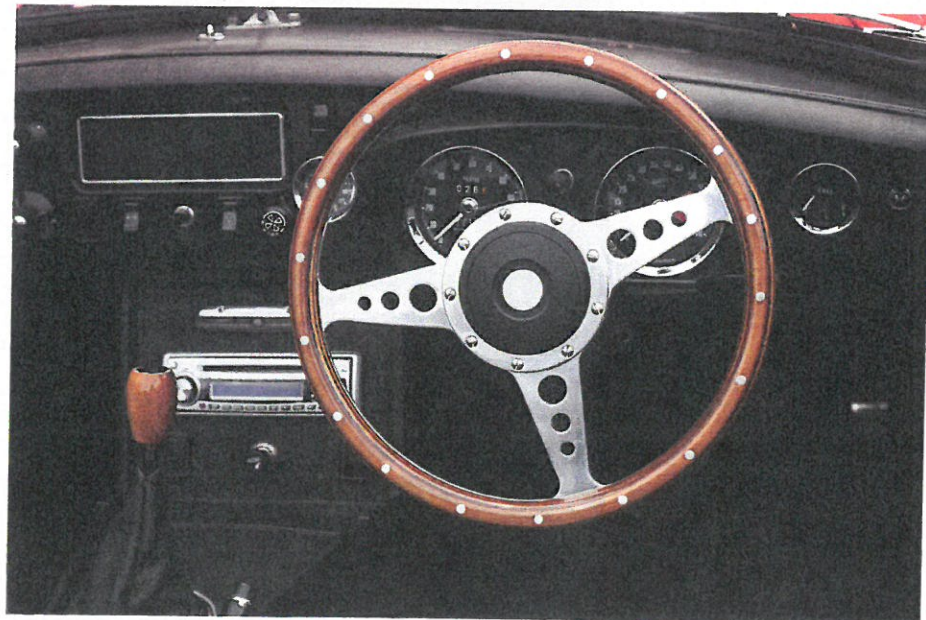
to a severe down-turn in the world economy, which significantly affected his trade in fitting and reselling classic and vintage car parts. At one point he had a stock of 200 car components.

Clearly, 200 car components can range in size and complexity depending on their location in the classic and vintage cars, from small temperature sensor units to large engine components and their assemblies and sub-assemblies of ancillary systems as are found in fuel, exhaust, and electrical systems. Initially, Shane stored them in his workshop in the rack system that he had been extending as the number of parts grew. Although his business had slowed significantly, he found that he was spending an increasing amount of time managing access to his stock of parts and determining their availability for customers' classic and vintage car repairs and restorations. The manual labeling and inventory system that he had successfully used previously was becoming cumbersome, unwieldy, and increasingly inaccurate.

Once the economic recovery began in 2013, demand for his classic and vintage car parts increased again as his car repairs and restoration business expanded. At this point, he decided that he would minimize his stock of parts as quickly as possible by capitalizing on the available demand for his services. However, Shane had two problems. First, he did not reliably know which classic and vintage car parts he had in stock and where they were located in his workshop. Second, he wasn't willing to exhaust himself climbing the storage racks in order to locate components that are sometimes small in size.

Figure 5-18
Steering wheel of a
vintage car

Source: Claire Plumridge/Shutterstock



To address the problems, Shane created a Microsoft Access database with only one table: Part. To populate the database with data, Shane had to initially take stock of all the classic and vintage car parts and record the data as shown in the columns of Figure 5-21.

Obviously, a single-table database could have been stored in Excel, but Shane needs to query his data in a number of different ways. For example, he wants to know the location of the engine temperature sensors in his workshop racking system that have a temperature range suitable for the cooler conditions of the Scotland region. He also wants to know which fuel system component parts are available for different grades of fuel. Further, customers have special needs. For example, one might want an

exhaust system component that generates a particular sound note during acceleration, which is in accord with the original manufacturer's specification of the classic and vintage car. In the absence of a database, he doesn't know whether he has the part or not. Alternatively, he might want to know of all of the electrical parts concerned when he decides that he needs to refurbish an electrical part with a new component, and so on. Because of his changing needs, particularly concerning his questions regarding the data stored in his database and his vast reserve of motor parts, Shane uses the Access query facility. Figure 5-22 shows an example query that returns the location of temperature sensors in the workshop racking system that is suitable for the Scotland region, and Figure 5-23 shows the result of that query.

Figure 5-19
Engine Parts

Source: © MR. TEERASAK
KHEMNGERN/Shutterstock

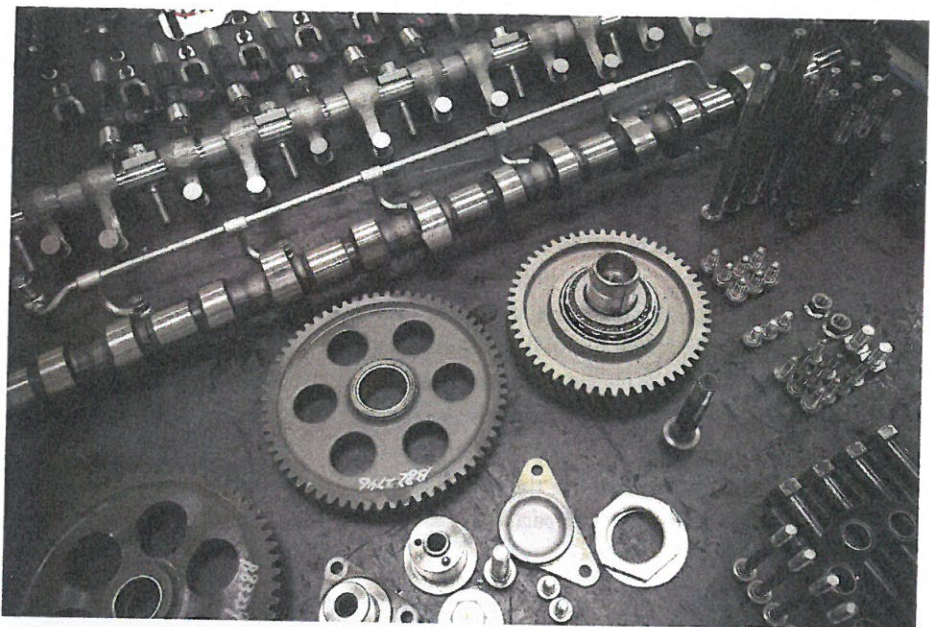


Figure 5-20
Car Parts

Source: © sima/Shutterstock

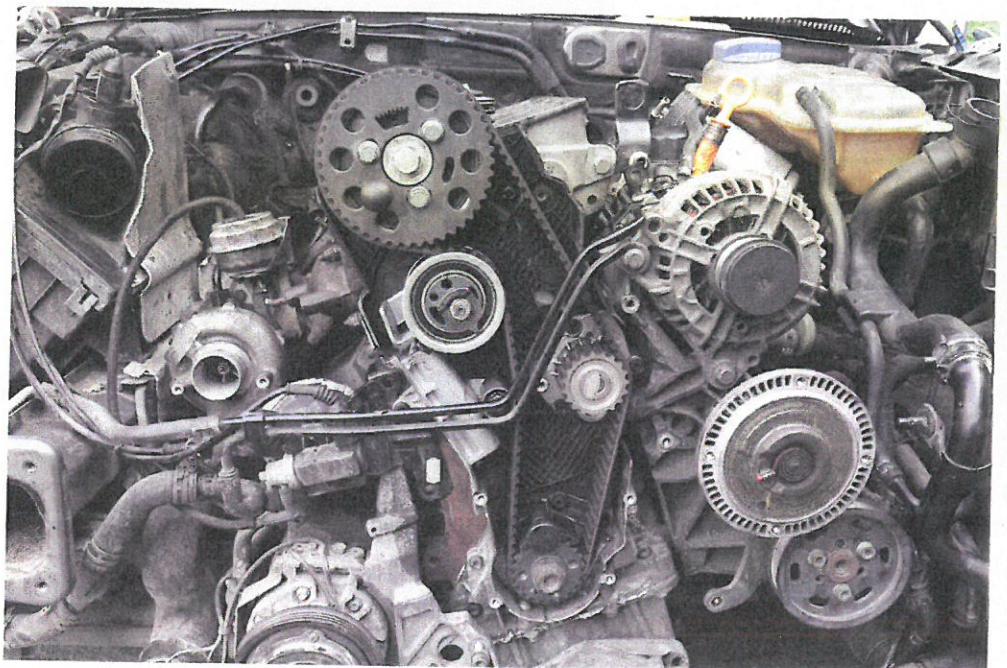


Figure 5-21
Columns in the Part Table

Source: Access 2013, Microsoft Corporation

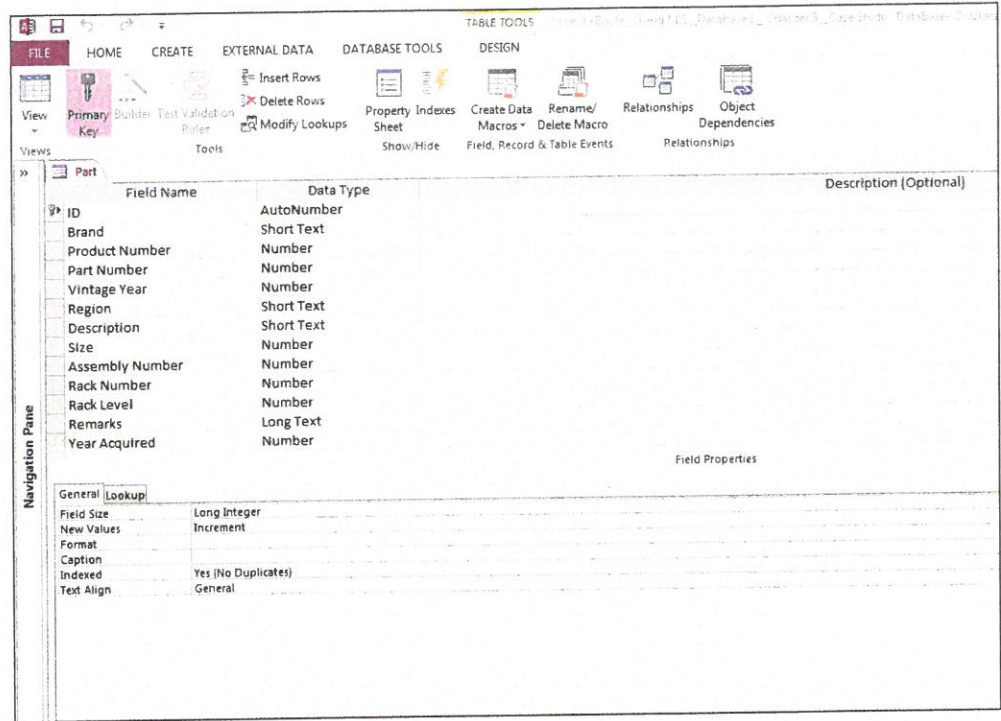


Figure 5-22
Example Query

Source: Access 2013, Microsoft Corporation

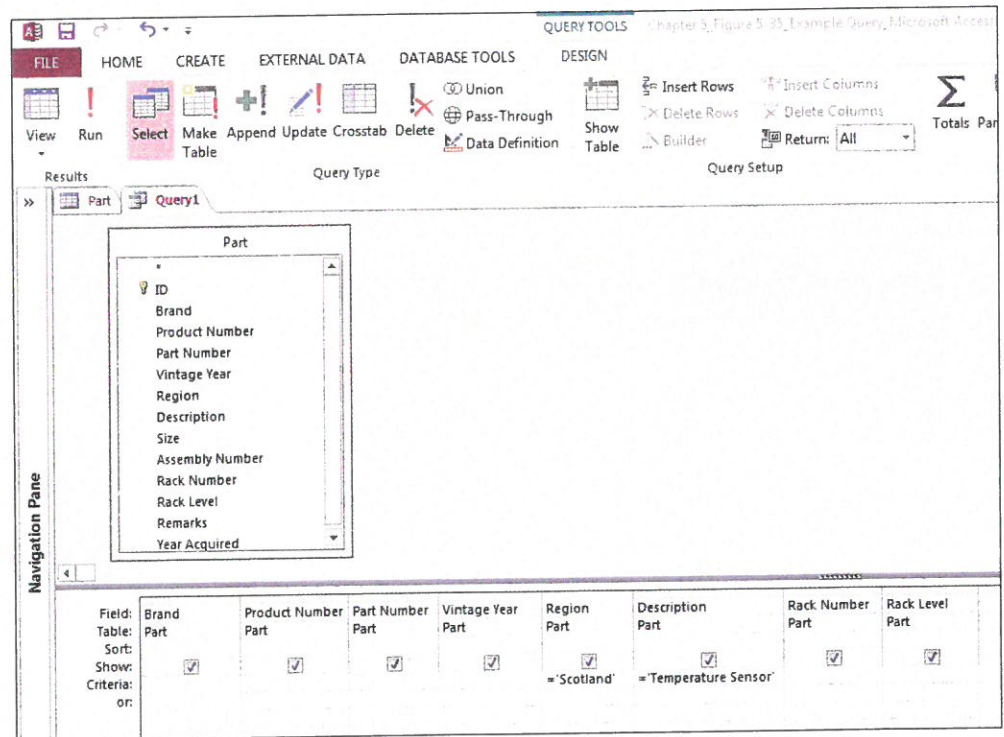


Figure 5-23
Results from Query in 5-22

Source: Access 2013, Microsoft Corporation

Brand	Product Number	Part Number	Vintage Year	Region	Description	Size	Assembly Number	Rack Number	Rack Level
Jaguar	1234	5678	1972	Scotland	Temperature Sensor	23	18	1	3
Jaguar	1234	3432	1983	Scotland	Temperature Sensor	18	23	3	1
Bentley	1234	7654	1965	Scotland	Temperature Sensor	22	12	2	0

QUESTIONS

- 5-12.** Explain why a simple, one-table only, Access database can be defined as a database with all of its benefits even though the data could be straightforwardly stored as an Excel spreadsheet.
- 5-13.** Justify the decision to use Access to store the classic and vintage car parts data.
- 5-14.** Examine the columns in Figure 5-21.
- Name three features of classic and vintage car parts that are not represented in this table.
 - If you were a business analyst advising Shane, what criteria should you and he use in deciding whether to include the additional data?
- 5-15.** Suppose that, in addition to the data about car parts, Shane wants to store data about the manufacturer such as its address, its years of manufacture, and general comments about the manufacturer.
- Design a Manufacturer table.
 - Alter the design of the Part table (Figure 5-21) to represent the relationship between Part and Manufacturer. State and justify any assumptions.
- 5-16.** Using the data in Figure 5-23, draw conclusions about the location and characteristics of the temperature sensor parts, their brand, and location in the racking system.
- 5-17.** Explain the statement "A database is an abstraction of some aspects of a business". Using this example, the ways that processing an abstraction is more effective than examining car parts. Explain the ways that processing an abstraction is more efficient than examining car parts. Generalize your observation to databases for business in general.
- 5-18.** This database will soon become useless if it is not kept up-to-date. List procedures that Shane needs to create and follow to keep his database current.

MyMISLab™

Go to the Assignments section of your MyLab to complete these writing exercises.

- 5-19.** Go to <http://aws.amazon.com> and search for AWS database offerings. Explain the differences among Amazon's RDS, DynamoDB, ElastiCache, and Redshift services. Which of these three would you recommend for storing Falcon Security's data? (By the way, whenever you query the Internet for any AWS product, be sure to include the keyword AWS in your search. Otherwise, your search will result in Amazon's lists of book about the item you're searching for.)
- 5-20.** Suppose you have just been hired by a tech startup that focuses on helping small businesses in South America market their handmade products to consumers around the world. They store a large amount of images and video of the products they sell. In addition, videos of the craftsmen making their products and talking about new products are very popular with customers. The CEO has asked you to look at the potential benefits of using a NoSQL DBMS like MongoDB or Cassandra for these images and videos. Describe the benefits of using a NoSQL DBMS to store this type of data. Would a relational database be a good option? Why?

ENDNOTES

1. MySQL was supported by the MySQL company. In 2008, that company was acquired by Sun Microsystems, which was, in turn, acquired by Oracle later that year. However, because MySQL is open source, Oracle does not own the source code.
2. Watch out for confusion between a *database application* and a *database application program*. A database application includes forms, reports, queries, and database application programs.
3. For a summary of this early history and an amplification of these ideas, see David Kroenke, "Beyond the Relational Model," *IEEE Computer*, June 2005.
4. Werner Vogel, "Amazon's Dynamo," All Things Distributed blog, last modified October 2, 2007, www.allthingsdistributed.com/2007/10/amazons_dynamo.html.
5. Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber, "Bigtable: A Distributed Storage System for Structured Data," OSDI 2006, Seventh Symposium on Operating System Design and Implementation, Seattle, WA, last modified November 2006, <http://labs.google.com/papers/bigtable.html>.
6. Jonathan Ellis, "Cassandra: Open Source Bigtable + Dynamo," accessed June 2011, www.slideshare.net/jbellis/cassandra-open-source-bigtable-dynamo.
7. Imperva, "Imperva Web Application Attack Report," July 2013, accessed May 19, 2014, www.imperva.com/docs/H11_Web_Application_Attack_Report_Ed4.pdf.
8. To learn more about how to prevent SQL injection you can visit OWASP.org. It has a helpful SQL Injection Prevention Cheat Sheet that explains how to parameterize queries and use stored procedures to stop SQL injection. See www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet.
9. Performance and Prestige Cars Limited, The Premier Car Company for Specialist Car Servicing and Repairs, www.a-cl.co.uk, accessed on January 2016.