

Database Management

Chapter Outline

- Introduction , 211
 - Relational Databases*, 213
 - Tables, Rows, Columns, Data Types*, 214
- Data Quality, 215
 - Data Integrity*, 216
 - Multitasking, Concurrency, and Integrity*, 217
 - Data Volume*, 218
- The Database Management Approach, 218
 - Focus on Data*, 219
 - Data Independence*, 219
 - Data Integrity*, 219
 - Speed of Development*, 220
 - Control over Output*, 221
- Queries, 221
 - Single-Table Queries*, 222
 - Different Types of Conditions*, 230
 - Computations*, 233
 - Computing Subtotals*, 238
 - Joining Multiple Tables*, 241
 - Examples*, 241
 - Multiple Tables, GROUP BY, and WHERE*, 243
 - Views*, 247
 - Converting Business Questions to Queries*, 248
- Designing a Database (optional), 250
 - Notation*, 252
 - First Normal Form*, 252
 - Second Normal Form*, 254
 - Third Normal Form*, 254
- Database Applications, 257
 - Data Input Forms* , 257
 - Reports*, 259
 - Putting It Together with Menus*, 261
- Database Administration, 262
 - Standards and Documentation*, 263
 - Testing, Backup, and Recovery*, 264
 - Access Controls*, 264
- Databases and e-Business , 264
- Cloud Computing, 266
- Summary, 268
- Key Words, 269
- Web Site References, 269
- Review Questions, 269
- Exercises, 270
- Additional Reading, 275
- Cases: Pharmaceuticals, 276

What You Will Learn in This Chapter

- How do you store and retrieve the vast amount of data collected in a modern company?
- How do you ensure that the transaction data is accurate?
- Why is the database management approach so important to business?
- How do you write questions for the DBMS to obtain data?
- How do you create a new database?
- How do you create business applications using a DBMS?
- What tasks need to be performed to keep a database running?
- Why are databases so important in e-business?
- How are databases used in cloud computing?

Eli Lilly and Company

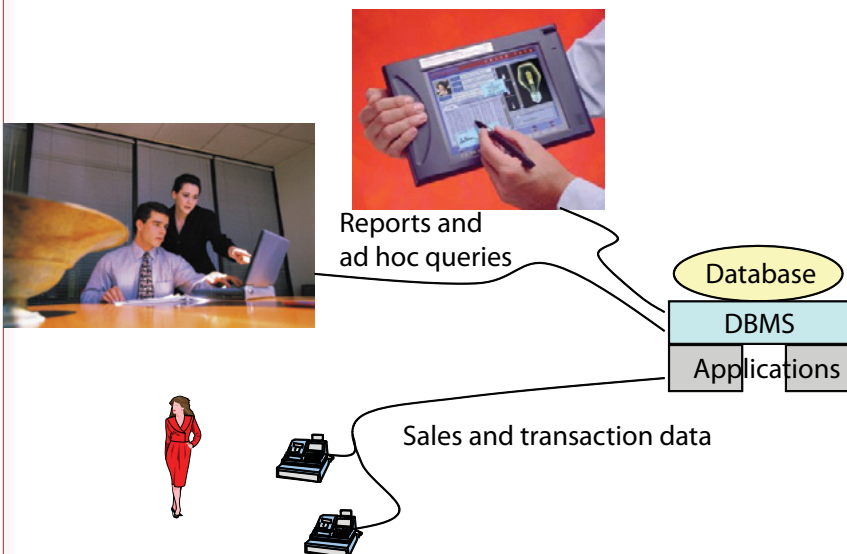
How do you store and retrieve huge amounts of data? Eli Lilly is a giant pharmaceutical company. Creating new drugs requires enormous efforts in research. Getting drugs approved takes years and dozens of lawyers. Except for the occasionally unique blockbuster, selling a drug requires delicate, but expensive, marketing. All three areas benefit from the use of information technology, particularly database systems. Evaluating chemicals, searching the huge database of existing results, and tracking progress generate huge amounts of data and require sophisticated analytical systems. Tracking clinical trials, the paperwork, and the progress of drugs through various agencies requires nontraditional databases and support. Marketing is a relatively new area for pharmaceutical companies, and they are still looking for ways to sell such complex products. Sometimes problems arise, such as when Lilly accidentally included the e-mail addresses of 669 subscribers to its prozac.com service in a message to 700 users.

Introduction

How do you store and retrieve the vast amount of data collected in a modern company? Airlines have a difficult problem: they need to track every seat and passenger on every flight. Thousands of people might be trying to book the same flight at the same time, but the system can never sell the

Figure 4.1

Without a DBMS, data can be scattered throughout the company, making it more difficult to share information. Inconsistent data, duplication and errors are common. A DBMS collects data from many sources and maintains data through a common interface. Data definition, access, consistency and security are maintained by the DBMS.



Trends

In the 1960s and 1970s, companies typically built their own transaction-processing systems by writing programs in COBOL. These programs consisted of millions of lines of code. Each program created and used its own set of files. As companies expanded, more programs were created—each with its own format for files. Whenever a manager wanted a new report or additional information, a programmer had to modify the old code or create a completely new program.

A database management system (DBMS) presents a different approach to data, reports, and programming. The most important task is to define and store the data so authorized users can find everything they need. Report writers and input screens make it easy to enter data and create reports without relying on programmers. Data is stored in a special format so that it can be shared with multiple users.

In the early 1970s, E. F. Codd created a flexible approach to storing data, known as the relational model, that avoided these problems. Today, relational databases are the dominant method used to store and access data. Relational databases have a query system that enables managers to get answers to questions without relying on programmers.

Early databases were designed to handle business types of data, such as customer names and account data. Some modern database systems can store entire books, pictures, graphs, or even sound clips as types of data. A few companies are working on object-oriented DBMSs that enable users to create their own data types and continue to manipulate and search the data.

same seat to different people. Many other business problems have similar characteristics. All of the accounting and payroll data, sales, and purchases have to be saved for any company. Database management systems were specifically designed to solve these problems. As shown in Figure 4.1, the primary elements of a database system are to collect and store data, produce reports, and provide data to answer business queries. Today, database systems form the foundation of almost every business application. What database features do you need as a manager? How can you retrieve the data stored by the system? How can you create reports?

A **database management system (DBMS)** is one of the most important tools in business and MIS. The systems have changed the way that computer applications are developed, and they are changing the way that companies are managed. The database approach begins with the premise that the most important aspect of the computer system is the data that it stores. The purposes of a database management system are to provide shared access to the data, answer questions, and create reports from the data.

A crucial factor with databases is that they can become massive. Several companies such as American Express and UPS have indicated that their databases contain several terabytes (trillions of bytes) of data. Even small companies deal with databases with megabytes (millions of bytes) of data. The size of the database greatly affects its performance and the ability of users to find the data they need. Large databases need to be designed and maintained carefully to ensure that they run properly.

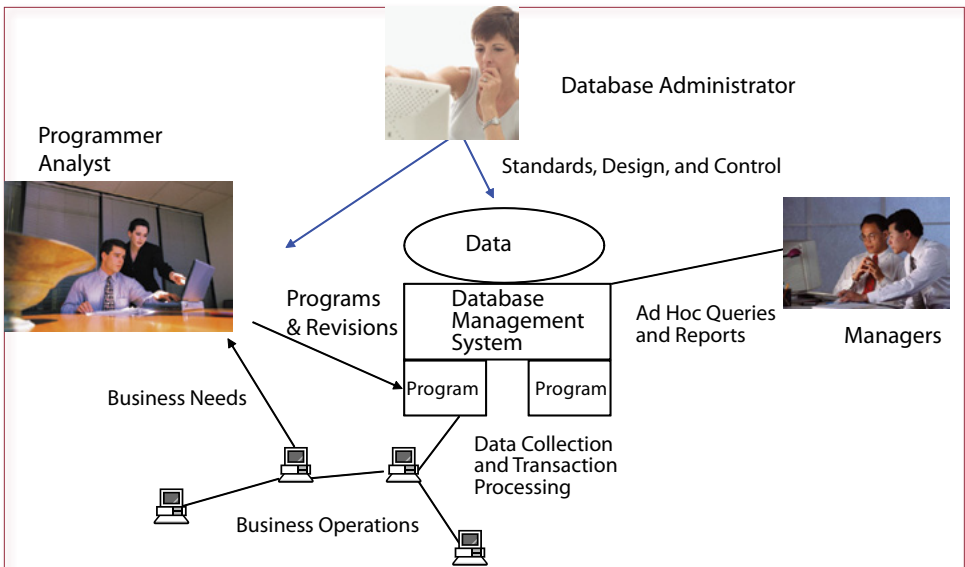


Figure 4.2

MIS employees and databases. The database administrator is responsible for defining and maintaining the overall structure and data. Programmers and analysts create applications and programs that collect data and produce reports. Business operations generate data that fills the database. Managers use the application programs and ask ad hoc questions of the data.

Another important characteristic of databases is that they are designed to help users examine the data from a variety of perspectives. Instead of simply printing one type of report, they enable users to ask questions and create their own reports. Figure 4.2 illustrates how a DBMS is used in an organization. It collects data for transaction processing, creates reports, and processes ad hoc queries for managers. Figure 4.2 also indicates that databases usually require programmers to define the initial database and maintain programs to perform basic operations. The overall design is controlled by the database administrator.

A DBMS is complex software that can be purchased separately or it can be embedded into an application. The primary vendors are Oracle (Oracle DBMS), IBM (DB2), and Microsoft (SQL Server). A few open-source systems also exist, led by MySQL and PostgreSQL. Today, MySQL is largely controlled by Oracle. The open source tools generally can be acquired at low or no cost; however, technical support or maintenance contracts can be purchased. Without the support contract, you would need skilled developers and administrators to handle all of the support tasks. Microsoft Access is commonly available with the Office suite, but it is limited to relatively small projects. Also, Microsoft is encouraging developers to switch to SQL Server.

Relational Databases

The goal of a relational DBMS is to make it easy to store and retrieve the data you need. All data is stored in **tables**, which consist of **columns** with **rows** of data. Each table has a name and represents objects or relationships in the data. For instance, most businesses will have tables for customers, employees, orders, and inventory.

Customer Table

CustomerID	Name	Address	City
12345	Jones	125 Elm	Chicago
28764	Adamz	938 Main	Phoenix
29587	Smitz	523 Oak	Seattle
33352	Sanchez	999 Pine	Denver
44453	Kolke	909 West	Denver
87535	James	374 Main	Miami

Sales Table

SaleID	CustomerID	Date	Salesperson
117	12345	3/3/09	887
125	87535	4/4/09	663
157	12345	4/9/09	554
169	29587	5/6/09	255

Figure 4.3

Creating table definitions. Tables are defined so that they can be linked by common columns. For a given row in the Orders table, you can find the corresponding customer data by locating the row with the matching phone number. In practice, this matching is handled by the DBMS query system.

Besides storing data, a modern DBMS has several useful tools. Input screens are built to help users enter data. Reports can be created by laying out data, text, and graphics on the screen—often with no programming. You can get answers to questions with a query language or even by pointing to tables and data on the screen. You can establish security conditions by granting or denying access to portions of the data. Most systems include an application generator that can tie input screens, queries, and reports together with a menu system. A complex application can be created by typing a few titles on the screen, without writing a single line of traditional program code.

Tables, Rows, Columns, Data Types

If you understand how spreadsheets work, it is easy to comprehend relational databases. A single spreadsheet consists of rows and columns of data. Each column has a unique name, and a row contains data about one individual object. A database consists of many of these tables that are linked by the data they contain.

In a database, each table contains data for a specific entity or object. For example, most companies will have a table to hold customer data. There are certain attributes or characteristics of the customers that you want to store. In Figure 4.3, each customer is assigned a unique CustomerID and has a name, address, and city. In practice, there will be more columns.

Figure 4.3 also illustrates one of the most important features of a database system: Relational databases are specifically designed to allow many tables to be created and then combined in interesting ways. If you had only one table, you could use a spreadsheet or virtually any filing system, assuming it could handle the num-

Reality Bytes: Database Terminology

When E. F. Codd created the relational database model, he deliberately introduced new terms to describe the way that databases should store information. His terms are attribute, tuple, and relation. Although Codd's terms are precisely defined mathematically, they can be confusing. As a result, many people use the slightly easier words: column, row, and table. Before relational databases, several different terms were used to refer to the various parts of a database. The problem is that many of the terms had several definitions. Common terms include field, record, and file. You should avoid these terms.

ber of rows you needed. However, most business problems involve data stored in different tables. In the example, customers can place many different orders. Each order is stored in a separate line in the Orders table.

Notice that the tables are joined or linked by the CustomerID column. The CustomerID is the **primary key** column in the customer table. Each row in a table must be different from the rest; otherwise, it is a waste of space. Consequently, each table must have a primary key. A primary key is a set of one or more columns that uniquely identifies each row. If someone gives you a key value (e.g., 12345), you can immediately locate the appropriate row and find the rest of the data for that entity (name, address, city).

Each primary key value must be unique, but that is hard to guarantee with most common identifiers. In some cases, you might use phone numbers, but what happens if a customer moves and gets a new phone number? In most cases, it is safest to have the computer generate identifier values that are guaranteed to be unique. However, do not expect customers (or salespeople) to memorize these numbers. The identifiers are simply used in the database to ensure there is a way to separate customers. You can always use names or phone numbers to look up other data for customers.

More complex tables require multiple columns to identify a row. These keys are called **composite keys**. They represent many-to-many relationships. For example, a typical order system requires an OrderItem table that contains two columns as keys: OrderID + ItemID. Both columns are keyed because many items can be ordered at one time (so ItemID is keyed), and an item can be ordered at many different times (so OrderID is keyed).

Unlike a spreadsheet, each database column can contain only one type of data at a time. For example, in the Date column you can store only dates. You would not be allowed to put names or totals in this column. Most relational databases were designed to hold business types of data. The basic choices are text, dates (and times), numeric, and objects (graphics, video, and sound). Some systems enable you to be more specific about how to store numeric data. For instance, you might want to store data with only two decimal places for monetary values. Whenever possible, dates should be stored in a date format instead of text. That way you can perform arithmetic on the values. For example, a formula like (today + 30) could be used to find a due date that is 30 days from today.

Data Quality

How do you ensure that the transaction data is accurate? Transaction-processing systems can become quite complex. Problems are going to oc-

Reality Bytes: The Government Pays Dead People

It can be hard to track the almost 2.5 million people who die each year in the United States. Over three years, the federal government sent \$180 million worth of benefit checks to 20,000 people who had died. Although the Social Security Administration maintains a database of people who have died, it was not routinely shared with other agencies. The White House Office of Management and Budget also noted that over three years, checks valued at \$230 million were sent to 14,000 convicted felons—some in jail, some still running. Although the General Services Administration maintains an “Excluded Parties List System,” few agencies actually check it for ineligible contractors. Of course, these numbers are dwarfed by the \$65 billion in erroneous payments made in a single year through the Medicare and Medicaid programs. Still, the government is working on building a comprehensive database that will make it easier for agencies to verify status before checks are sent out.

Adapted from *The Wall Street Journal*, “Database Aims to Prevent Uncle Sam From Paying Dead,” June 18, 2010.

cur in any system—especially because all business organizations change over time. That means the computer system has to change to match the business. It is virtually impossible to change the system at exactly the same time as the business, so problems will occur. Other problems arise simply because the systems are so complex. Many processes involve humans who make mistakes. If you understand what types of problems might arise, they will be easier to solve.

Data Integrity

One of the most important concepts in information processing is the issue of data integrity. **Data integrity** means keeping data accurate and correct as it is gathered and stored in the computer system. There is little value in an information system that contains out-of-date or inaccurate data. A common complaint among shoppers today is that stores using bar-code scanners might have a different price in the computer than the amount displayed on the shelf. It is easy to change prices in the computer; it is more difficult to change the signs in the store. Shoppers will feel cheated if the computer tries to charge them a higher price than the amount listed on the shelf. Some states, such as Michigan, have passed laws requiring that the scanned price cannot be higher than the amount listed on the package or display. Similar errors cause problems when the computer shows more items in stock than actually exist.

The first step to ensure data integrity lies in its capture. Each item must be correctly entered and the complete information recorded. It is sometimes possible to check the data as it is entered. Item code numbers usually include a check number that is based on the other digits. In the item code 548737, the first five digits add up to 27, so the number 7 is included as the last digit. If the person or machine makes a mistake entering one of the digits, they will probably not add up to 7, so the computer can immediately determine that there is an error. Sophisticated methods exist to catch mistakes involving more than one digit.

Even with machine entry of data, validity problems can arise. What happens when a shipment arrives but the receiving department forgets to record it? The same problem occurs when a clerk holds an item for a customer and does not re-

Transaction A	Customer Accounts Sanchez: Balance	Transaction B
1. Receive 300 payment 2. Read balance (500)	Sanchez: 500	
		3. New purchase (350) 4. Read balance (500)
5. Subtract payment 6. Store new results (200)	Sanchez: 200	
	Sanchez: 850	7. Add purchase 8. Store new result (850)

Figure 4.4

Concurrency and data integrity. Multiuser and multitasking systems can cause problems with concurrent changes to data. Two processes cannot be allowed to change the same data at the same time. A key strength of a DBMS is that it is built to handle these problems with minimal effort.

cord it in the computer. Data integrity can be destroyed by indiscriminately allowing people to change the numbers in the computer. It is one of the main reasons for creating secure computers and controlling access to each piece of information.

Multitasking, Concurrency, and Integrity

A useful feature offered by more sophisticated operating systems is the ability to perform more than one task at a time. In many situations it is useful to have several jobs running at the same time. What happens if you are searching a huge database and your boss calls and asks you for a sales figure? With a multitasking computer operating system, you could switch to a new program, look up the number, and allow the database to continue searching in the background.

If you use a multitasking operating system, it is important that your application software understand that other applications might be running at the same time. Each application needs to protect its data files from concurrency problems. **Concurrency** arises when applications attempt to modify the same piece of data at the same time. If two people are allowed to make changes to the same piece of data, the computer system must control the order in which it processes the two requests. Mixing the two tasks will result in the wrong data being stored in the computer. These problems can be avoided by only using software that was specifically written for multiuser (or multitasking) computers.

Consider the case of a mail-order firm shown in Figure 4.4. On the left side, customer Sanchez sent a payment on his account. At the same time the clerk begins to process the payment, Sanchez calls a second clerk and places a new order. The figure shows what happens if both transactions continue and interfere with each other. What should the final balance be? Does the computer have the correct number?

To solve this problem, the application program must know that several people might try to access the same piece of data at the same time. The software locks out all users except one. When the first process is finished, the other users can try to gain access again. To keep data accurate, applications used by many people at the same time must be written to handle these concurrency problems. Early personal

computers were designed for only one user, so much of the software did not prevent concurrency problems. Software designed for computer networks generally handles this issue. When you use this software, you will occasionally receive a message that says a piece of data you desire is currently being used by another person. If you get this message, simply wait for a few minutes and try again. When the first person is finished, you should be able to proceed.

Data Volume

A common problem experienced by a growing business is the increase in the amount of data or data volume. Consider the huge databases handled by Information Resources, which processes data from supermarket checkouts, or United Parcel Service, which tracks every package every day.

As the business grows, there will be an increase in the number of transactions. As the price of a computer drops, more applications are placed on the computer. Additional transactions become computerized. Several problems can be created from this increase: (1) processing overload or system slowdowns, (2) greater difficulty in making sure the data is accurate, (3) insufficient storage within the computer system, and (4) data not captured fast enough.

Visa International processes more than 6 billion electronic transactions a year. By the year 2000, the company was handling 15 billion annual transactions. There are 18,000 banks offering Visa cards, used by 10 million customers. So much data is generated on a daily basis that Visa cannot keep transaction data online beyond six months. All older records are moved to backup storage, making them inaccessible for additional research or decisions.

Sloppy practices and huge datasets can lead to inaccurate data. As the system slows down or the computer runs out of storage space, people avoid using it, so data is no longer up to date. With the increase in volume and the computerization of new types of data, it is more difficult for programmers and managers to check the data. If parts of the computer system are too slow, data may not be captured fast enough. As a result, some data might be lost. A tremendous amount of information is stored in raw data. The raw data could be analyzed to offer new services or improve the quality of existing services. However, the huge volumes require too much storage space and too much processing time.

Careful planning is required to avoid these problems. At best, new computers and storage usually take a month or two to purchase. It could take a year or more to evaluate and purchase a large, expensive central computer. The MIS department would like to forecast the demands that will be placed on the computers at least a year in advance.

The Database Management Approach

Why is the database management approach so important to business? In many ways, the database approach has revolutionized the way information systems function and altered the way businesses operate. Originally, all programs handled their own data in separate files. It took enormous coordination and documentation to try and make the multiple programs work together. The DBMS changes everything by focusing on the data instead of the programs. Its primary purpose is to store and share the data.

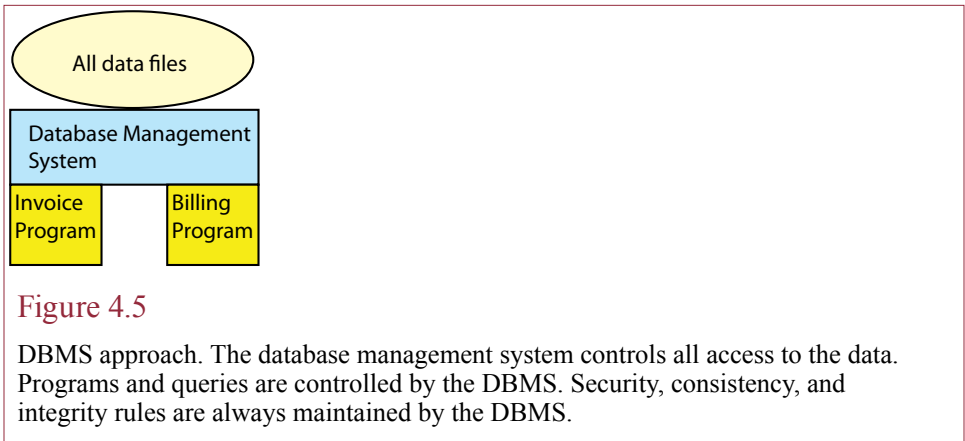


Figure 4.5

DBMS approach. The database management system controls all access to the data. Programs and queries are controlled by the DBMS. Security, consistency, and integrity rules are always maintained by the DBMS.

Focus on Data

The database management approach is fundamentally different from the older programming methods. Whenever someone needs a computer application, the first step is to identify the data that will be needed. Then a database management system is used to store the data. It takes care of storing the raw data, as well as information about that data. Everything you want to know is stored within the DBMS, not in an application program. This situation is illustrated in Figure 4.5. The goal of the DBMS approach is to collect accurate data and make it available to users. The system should also minimize unnecessary duplication of data.

Data Independence

Defining the data separately from the programs is called **data independence**. The main advantage is that it is possible to change the data without having to change the programs. For instance, you might want to add a second phone number to the customer data. With a DBMS, you can make this change, and it will not affect any of the existing programs. Similarly, the reports can be modified without having to change the data. That means when the programmer is called in at 3 A.M., she has to change only one program. All the other programs will be unaffected. Besides making the programmer's life easier, the database is more likely to be accurate and there will be less duplication of data.

Data independence means that the data and programs are separate, which makes it possible to alter the database tables as needed, without destroying the programs. As the business grows, you might decide to collect additional data, such as keeping track of sales by salesperson or by sales route. As the company expands and changes, the underlying tables can be expanded and new tables can be added—without interfering with the existing tables or current programs. Just be careful to avoid deleting tables or removing entire columns. Of course, as the business changes, managers will undoubtedly want to modify the reports to add the new information.

Data Integrity

Data integrity is an important consideration in designing and maintaining databases. Data integrity means that the database holds accurate, up-to-date data. In the airline case, it means not selling the same seat to two different people. If there are

Reality Bytes: Government Databases can be Really Big

Take 130 million people, multiply by every healthcare visit, and every diagnoses and treatment; and you end up with a giant database. As the Federal government increases its involvement in healthcare, it needs to build a giant database to track dozens of aspects of healthcare for individuals and groups. Of course, a giant database raises giant privacy questions. The government says that it can anonymize the data. The goal is to track price and quality patterns to focus on groups and not individuals. Initially, the database will focus on Federal employees, and include the ability to evaluate longitudinal effects—essentially tracking treatments over time. Analysts will have access only to de-identified data. The inspector general's office also plans to use the database to monitor for fraud and waste, which means it needs access to provider information.

Adapted from Jaikumar Vijayan, "Feds Move Toward Health Claims Database Despite Privacy Fears," *Computerworld*, June 16, 2011.

business limits on certain values, the database should force the data entry to abide by those rules. For example, prices will always be positive numbers. Another integrity concept is the importance of identifying missing (null) data. Computations should be able to skip missing data. From a manager's viewpoint, an important integrity design element is naming columns carefully. The columns should have names and descriptions so that the users understand what is stored in the database. If a column is simply labeled *Revenue*, users might wonder if that means revenue from all products, all divisions, the current year, or perhaps just monthly totals. All of this information is stored with the database in the data dictionary.

An important component of database integrity is that the data needs to be consistent. For example, consider a table that describes products for sale. Perhaps the products are grouped into categories (cleaning supplies, paper goods, clothing, etc.). Each item belongs to only one category. What happens if the categories are not entered consistently? The *Cleaning Supplies* category might be entered as just *Cleaning*, or maybe as *Clean Supplies*, or even *Cl Sup*. These variations in the data make it difficult to search the table based on that category because the user would have to know all of the variations. A good DBMS supports rules that can be used to minimize these problems. However, when dealing with databases, it is good practice to be careful when you enter data to ensure that your entries are consistent.

Speed of Development

It is possible to create an entire database application without having to write a single line of traditional programming code. As a result, an application can be built in a fraction of the time it would take to create it by writing COBOL programs. Studies indicate that most systems can be created 10 times faster using a DBMS—if the data already exists in the database. As the commercial database products—such as Oracle, SQL Server, and DB2—continue to add features, they can be used to solve even more complex problems.

Keep in mind that it is possible to use traditional programming tools such as COBOL in conjunction with the DBMS. If complex reports or complicated calculations are involved, it is sometimes easier to use a traditional programming

Four Questions to Create a Query

- What output do you want to see?
- What do you already know (or what constraints are you give)?
- What tables are involved?
- How are the tables joined?

Figure 4.6

Four questions to create a query. You will always have to answer these four questions. In many cases, there will be only one table (or view), so the second and last questions are easy.

language. These programs retrieve the base data from the DBMS and print their own reports or store computed values in the database for later use.

One of the most important steps of developing a solution is to break the problem into smaller pieces. One major piece of any problem is the data. A DBMS makes this portion of the problem easier to solve. By putting the DBMS in charge of maintaining the data, keeping track of security, automatically supporting multiple users, and printing reports, the developer can concentrate on solving the specific business problems. By starting from scratch with COBOL, each of these features would have to be rewritten for every application that was designed, which would be expensive.

Control over Output

Another strong advantage of database management systems is their ability to provide many different views of the output. In fact, a primary objective of the relational database approach is to store the data so that users can retrieve it any way they need. The other feature of databases is that they are designed to make it easy to combine related data. An older programming/file approach generally limits the user to using data in only one way.

With a DBMS, output can be created from report writers, which make it easy to format the data; some systems even draw graphs. The other common method of retrieving data is to use a query language such as query by example (QBE) or SQL discussed in the next section. Queries enable managers to search for answers to questions without using a programmer to write special programs.

Queries

How do you write questions for the DBMS to obtain data? Most of the time, managers work with databases that have been created by someone else. You will need to learn how to retrieve data to answer questions. It might be nice to be able to ask questions in a natural language (such as English), but it turns out to be hard to make computers understand these questions and you might not always be certain that the answer is what you asked for. A DBMS provides at least one method of asking questions and retrieving data. Two common methods are QBE and SQL. **SQL** is an international standard method for retrieving data from database management systems. It is supported by most of the major commercial relational database management systems. By the way, according to the standard, the name SQL is just three letters and not an acronym. **QBE** stands for **query by example** and is a visual method of examining data stored in a relational database.

Customers

CustomerID	Name	Phone	City	AccountBalance
12345	Jones	312-555-1234	Chicago	197.54
28764	Adamz	602-999-2539	Phoenix	526.76
29587	Smitz	206-676-7763	Seattle	353.76
33352	Sanchez	303-444-1352	Denver	153.00
44453	Kolke	303-888-8876	Denver	863.39
87535	James	305-777-2235	Miami	255.93

Figure 4.7

A sample table for customer data. CustomerID is the primary key and is used to uniquely identify each customer.

You ask questions and examine the data by pointing to tables on the screen and filling in templates. Queries can only answer questions for which you have collected the appropriate data.

Regardless of the method used to look up information in a database, there are four basic questions you will answer, as listed in Figure 4.6. It does not matter in which order you think of the questions. With some methods (such as QBE), it is easier to choose the tables first. With other methods (such as SQL), it is sometimes easier to choose the output first. In many cases, you will switch back and forth among the four questions until you have all of the components you need. As you learn more about databases, keep these four questions handy and write down your answers before you attempt to create the query on the DBMS.

Single-Table Queries

Consider a simple customer table that contains columns for CustomerID, Name, Phone, Address, City, State, and AccountBalance. Each customer is assigned a unique number that will be used as a primary key. The AccountBalance is the amount of money the customer currently owes to our company. The table with some sample data is shown in Figure 4.7. The tables in these examples are intentionally small to make it easier to understand the concepts. A real table of customers would contain more columns and thousands of rows of data. But the basic concepts will remain the same. Each column represents some attribute or characteristic. Each row holds data for a single customer. The CustomerID is the primary key for this table and it must always be a unique number. In this example, the ID was randomly assigned by the marketing department when a customer was added to the database. It is more common to have the DBMS generate unique ID numbers, but you have to remember that the values are often random.

Query by Example

Query-by-example systems that were designed for graphical user interfaces (GUIs) are especially easy to use. Microsoft's Access illustrates a common approach. The basic mechanism is to make selections on the screen—typically by pointing to them with a mouse. You then fill out a template like the one shown in Figure 4.8. Note that Access has several ways to create queries. The Design (or

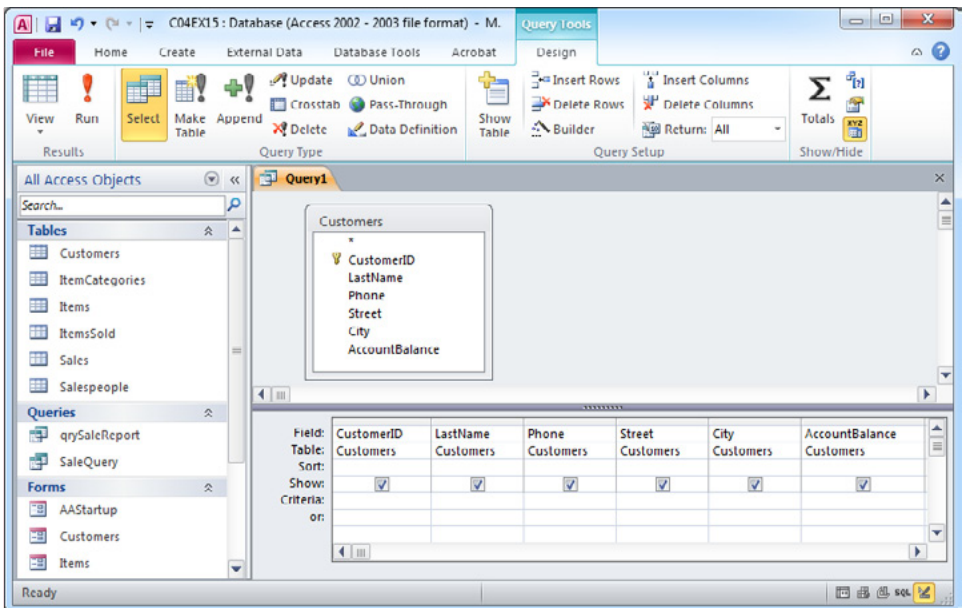


Figure 4.8

Design: List all of the customers. First select a table, then drag columns you want to see onto the grid. Checking the Show box ensures that the column will be displayed when the query is run.

QBE) approach is probably the easiest to learn. The Wizard is not very useful. You should also be able to read the text SQL statements, which are shown later in this chapter. In the end, the query systems are basically the same. You must always answer the four basic questions: (1) What output do you want to see? (2) What constraints are you given? (3) What tables are involved? (4) How are the tables joined together? This section focuses on a single table and saves the complications of the last two questions for later.

Figure 4.9

Results for the query that selects all rows of the Customers table.

CustomerID	LastName	Phone	Street	City	AccountBalance
12345	Jones	(312) 555-1234	125 Elm Street	Chicago	\$197.54
28764	Adamz	(602) 999-2539	938 Main Street	Phoenix	\$526.76
29587	Smitz	(206) 676-7763	523 Oak Street	Seattle	\$353.76
33352	Sanchez	(303) 444-1352	999 Pine Street	Denver	\$153.00
44453	Kolke	(303) 888-8876	909 West Avenue	Denver	\$863.39
87535	James	(305) 777-2235	374 Main Street	Miami	\$255.93
*	0				\$0.00

Record: 1 of 6 No Filter Search

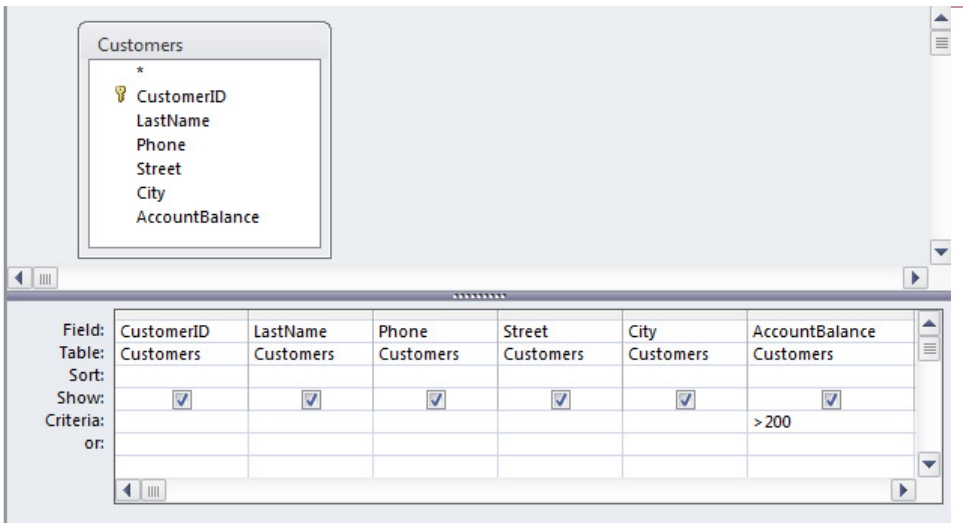


Figure 4.10

Design query. List the customers with an AccountBalance of more than \$200. Results of the query can be sorted in Ascending or Descending order. Multiple levels of sorts are created by selecting additional columns. You will use multiple column sorts when the first column contains several identical values. For example, sort by City, Name.

The Design view or QBE approach begins by asking you a hard question first: Which tables are involved. Fortunately, you can add tables later, so it is usually best to focus on one table at a time. For now, you can start a new query and choose the Customers table. Your next step is to decide what columns you want to see in the results. In this case, drag the CustomerID, Name, Phone, City, and AccountBalance columns onto the grid. You can drag multiple columns at a time, double-click a column name, or select it from the drop-down list on the top of the grid. It does not matter how you get the column name in place, as long as it is in the list and the Show checkbox is set, data will be displayed for that column.

Figure 4.11

Rows that match the conditions. It is easy to check that the AccountBalance is larger than 200 in each row. Even if your query has more rows, you should check random rows to ensure you entered the conditions correctly.

CustomerID	LastName	Phone	Street	City	AccountBalance
28764	Adamz	(602) 999-2539	938 Main Street	Phoenix	\$526.76
29587	Smitz	(206) 676-7763	523 Oak Street	Seattle	\$353.76
44453	Kolke	(303) 888-8876	909 West Avenue	Denver	\$863.39
87535	James	(305) 777-2235	374 Main Street	Miami	\$255.93
*	0				\$0.00

Record: 1 of 4 No Filter Search

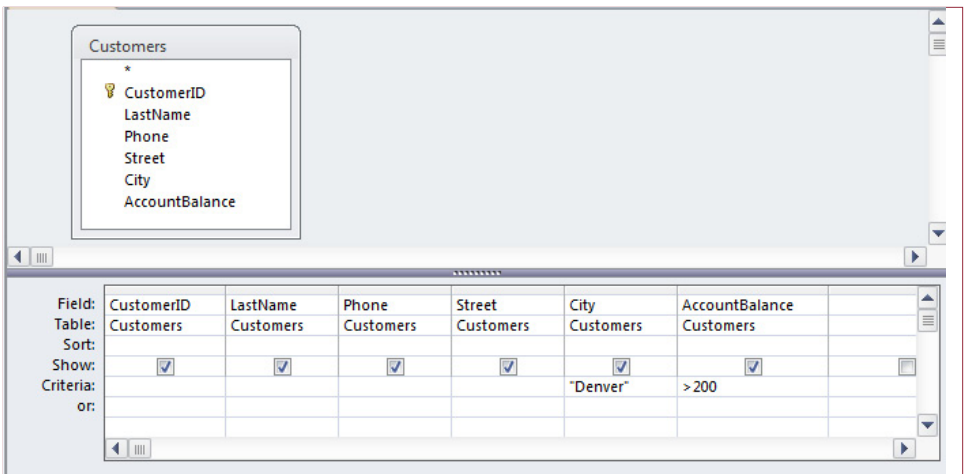


Figure 4.12

List the customers from Denver with an AccountBalance of more than \$200. Conditions entered on the same row are joined with an And. Conditions on different rows are connected with a logical Or.

You can run the query by selecting the DataSheet view or by clicking the Run button. For most queries, the DataSheet view is more convenient because you often want to select the Design view to return and edit the query. Figure 4.9 shows the results of running the simple query that selects all of the rows in the Customers table. If you ever want to copy the rows into a spreadsheet or some other document, you can click the small gray square at the top-left of the results to select all of the rows and columns. Selecting the Copy option will put the results onto the clipboard.

Most of the time, you will want to see only some of the rows of data. For instance, you want a list of customers who owe you the most money. You decide to restrict the listing to customers who have account balances greater than \$200. With QBE, you enter the appropriate restriction in the criteria row under the desired column as shown in Figure 4.10. Look at the figure carefully and check the greater-than sign. You read the condition as: AccountBalance > 200. You can think of this condition as a filter on the rows that will be displayed. Only rows that meet the specified conditions will be displayed.

Figure 4.11 shows the results of running the query. Double-check the data to ensure that all of the rows meet the condition you specified. Checking your results is a good habit to learn. One of the trickiest aspects of queries is that the DBMS will almost always return something—but you need to verify that the rows it returns are actually the rows that you want to see.

You can specify other conditions for the other columns. Placing them on the same row means they will be interpreted as AND conditions. If conditions are placed on separate rows, results will be computed for rows that match at least one of the criteria (OR condition). Figure 4.12 shows the QBE screen, which tells the DBMS to display the ID, City, and AccountBalance for customers who live in Denver and have account balances of more than \$200.

CustomerID	LastName	Phone	Street	City	AccountBalance
44453	Kolke	(303) 888-8876	909 West Avenue	Denver	\$863.39
*	0				\$0.00

Record: 1 of 1 No Filter Search

Figure 4.13

Rows that match the conditions. To be displayed, all of the conditions on the row must be true. In this case, only one customer is from Denver and owes more than \$200.

As shown in Figure 4.13, the AND condition creates a filter so that rows are displayed only if they match all of the conditions on that row. In this example, the customer must be from Denver and owe more than 200. Because of the small number of customers, only one person meets both conditions. But, you begin to see the value of using conditions to limit the rows you want to see. Many businesses have huge databases, and it is impossible to search through the data by hand. You can use a simple query to quickly find the rows that match almost any set of conditions.

AND conditions are used to narrow a search. Each time you add an AND condition, you are creating one more hurdle that a row must meet to be displayed. Sometimes you need to widen your search. In these cases, you use OR conditions to impose an alternate set of criteria. Figure 4.14 shows the query for searching for customers who live in either Denver or Chicago. Notice that there is no AccountBalance condition. If you wanted to add the AccountBalance condition,

Figure 4.14

List the customers from Denver or Chicago. Conditions entered on different row are joined with an Or, which widens the search because rows need to match only one of the conditions.

The screenshot shows a database query builder interface. On the left, a table named 'Customers' is displayed with columns: CustomerID, LastName, Phone, Street, City, and AccountBalance. Below the table, a query criteria table is shown with the following structure:

Field:	CustomerID	LastName	Phone	Street	City	AccountBalance
Table:	Customers	Customers	Customers	Customers	Customers	Customers
Sort:						
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:					"Denver"	
or:					"Chicago"	

CustomerID	LastName	Phone	Street	City	AccountBalance
12345	Jones	(312) 555-1234	125 Elm Street	Chicago	\$197.54
33352	Sanchez	(303) 444-1352	999 Pine Street	Denver	\$153.00
44453	Kolke	(303) 888-8876	909 West Avenue	Denver	\$863.39
*	0				\$0.00

Record: 1 of 3 No Filter Search

Figure 4.15

Rows that match the Or condition. Customers are displayed who either live in Denver or Chicago.

you would have to enter it onto both of the criteria rows. With multiple condition rows, the query processor tests to see if a row meets all of the conditions on one row to decide whether to display it. If you were to put the `AccountBalance>200` condition only on the Denver row—it will apply only to customers who live in Denver. All of the Chicago customers would be displayed—regardless of their `AccountBalance`.

Figure 4.15 shows the list of customers who live in Denver or Chicago. Again, it is important to check the results to ensure that you entered the conditions correctly. If you accidentally enter the conditions with an AND clause by putting them on the same row, you would not get any matches. In this database, no customer can live in both Denver and Chicago at the same time.

It is also easy to sort the results. For example, you probably want to which customer owes the most money. Instead of trying to find just one customer, businesspeople often want to see the list sorted by the `AccountBalance`. That way, if two people owe close to the same amount, they will both be displayed. Figure 4.16 shows that you can sort the results simply by selecting the sort-order in the Sort row. In this case, you want to sort the data in descending order (high-to-low) so that the customers who owe the most money are displayed at the top of the list. Create and run the query to see which person ends up on top.

You will also encounter situations when you need to sort by multiple columns. Names are the most common example. In an English-speaking nation, your database might contain many people with the last name Smith. If you sort only by last name, the rows will still be difficult to read. Instead, you should sort by `LastName` and `FirstName`. The one catch in QBE systems is that you need to display the `LastName` column to the left of the `FirstName` column so that the query sorts first by `LastName`, followed by `FirstName`.

SQL

Another method of retrieving data from a DBMS is with the query language SQL. Although some people find SQL more difficult to learn, it has two advantages. First, it is a standard language that is supported by many different database systems, so the commands will work the same in many situations. Second, it is easier to read than QBE, so it is easier to understand your queries. It is also simpler to write down on paper because it relies on words instead of pictures or layout. Keep in mind that SQL requires the same answers as QBE—the main difference is that you have to type all of the column names instead of dragging them into place.

SQL is a moderately complex language. Fortunately, this chapter uses only a few simple SQL statements. You can take a database class to learn more SQL

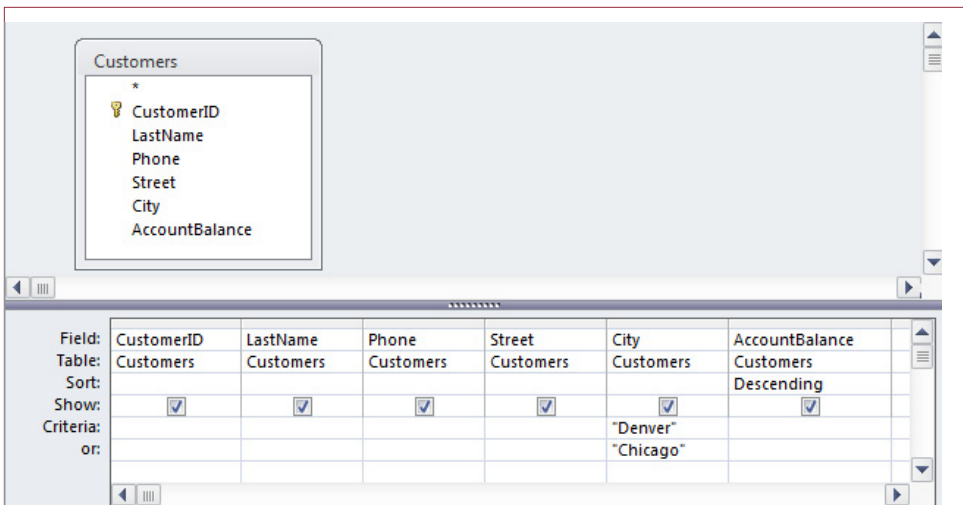


Figure 4.16

Sorting the results. Select Ascending or Descending on the Sort row for the column that needs to be sorted.

details and understand the true power of the language. If SQL seems complex, remember that it is considerably easier than trying to write programming code to retrieve the data. It is even much simpler than trying to build a spreadsheet to find the answers.

The standard command for retrieving data in SQL is `SELECT`. To be clear, SQL command words in this book will be written in capital letters with tabs to separate the words, but you can type them into the computer as lowercase. The simple form of the command is shown in Figure 4.17. The five parts are written on separate lines to make the command easier to read. One of the benefits of SQL is that you can write down the keywords and then fill in the blanks in any order. Typically, you can start with the list of columns to display and then fill in the conditions. From these two lines you can figure out exactly which tables are needed.

In most cases, the best approach is to decide which columns you want to see. When you read a business question, think about the values that you want to see that would answer the question. For example, if you want to see a list of customers that meet some condition, you want the final result to include at least their `FirstName`, `LastName`, and possibly the phone number or address. These columns can be listed in whatever order you want. The column names should be separated by commas. If you want to see all the columns, you can use the key word `ALL` or an asterisk (`*`). This approach (`SELECT *`) reduces the typing, but often results in a huge number of columns. But it is also useful if you cannot remember the names of the columns or how they are spelled.

To illustrate the power of SQL, try creating a simple query to list all of the columns and all of the rows from the `Customers` table. Figure 4.18 shows the query you want to enter. Microsoft Access supports SQL. In fact, all of the QBE queries you build are actually converted to SQL. Start a new query in Access and choose Design view. Do not add any tables. Click the SQL button to switch to SQL text mode. Type the `SELECT` command shown in the figure and run the query. You

```

SELECT    columns
FROM      tables
JOIN      matching columns
WHERE     conditions
ORDER BY  column {ASC | DESC}

```

Figure 4.17

The SQL SELECT command. This command is used to retrieve and display data. It is the foundation of many of the other SQL commands.

will receive an error message if you type one of the words incorrectly, such as forgetting the “s” in Customers. The results are similar to those in Figure 4.10, but also include the Street column.

Next, you need to know the name of the table. The SQL command to retrieve all of the customer data is `SELECT * FROM Customers`. The result can be sorted by adding the `ORDER BY` clause. For example, `SELECT * FROM customers ORDER BY City`.

To get a list of customers who live in Denver with account balances greater than \$200, you need to add a `WHERE` clause. The command becomes `SELECT * FROM Customers WHERE (AccountBalance > 200) and (City = “Denver”)`. Notice the similarity to the QBE command. Of course, with SQL, you need to remember (and type in) the names of the tables and columns. `NULL` values and `BETWEEN` commands are also available in SQL.

SQL is also useful when you have complex conditions with several `AND` and `OR` connectors. It is easy to enter conditions in the QBE grid, but they can be difficult to read. Remember that you get completely different results if a condition is entered on a different line. But, with many conditions spread across several columns, it can be difficult to see if conditions are on the same line. With SQL, you simply read the text and verify the `AND` or `OR` connectors. Figure 4.19 shows the query that lists the customers who live in Denver and owe more than \$200. Notice that it is easy to read the `AND` connector. However, you should always use parentheses to isolate the separate conditions. Also, you have to type in the quotation marks for any text comparisons (“Denver”).

One nice feature of Microsoft Access is that you can switch back and forth between SQL and QBE. You can even edit the query in either mode and get the same results. You can use drag-and-drop to create the query, and switch to SQL to verify the `AND` and `OR` connectors. To see the effect, use QBE to rebuild the query in Figure 4.16 that lists customers from either Denver or Chicago. Figure 4.20 shows the text when you switch to SQL view. First, note that Access always uses the full

Figure 4.18

Display all columns and rows from the Customers table. You begin to see the power of SQL when you realize you can instantly get all of this data with only three words (and a symbol).

```

SELECT    *
FROM      Customers

```

```

SELECT Name, Phone, City, AccountBalance
FROM Customers
WHERE (AccountBalance>200) AND (City="Denver")

```

Figure 4.19

SQL query to list customers who live in Denver and owe more than \$200. It is relatively easy to verify the AND connector. Be sure to use parentheses to separate the conditions.

name of columns, which includes the table name, such as Customers.Name. This approach is reasonable because different tables might use the same column name (such as Salespeople.Name and Customer.Name), and Access needs to keep track of exactly which column you mean. Second, Access uses too many parentheses in the conditions, which makes it hard to edit; but the OR connector is easy to see and you can always return to the QBE grid to change the query conditions.

Different Types of Conditions

It is relatively easy to enter conditions on the criteria rows of the QBE grid or to add them to the SQL WHERE clause. However, to handle more complex cases, you need to understand all of the comparison operators. Some special operators also exist that can save you effort when creating queries. As a manager, you need to be able to convert business questions into database queries. Identifying and specifying conditions is a key step in this process. Once you understand the basic elements of the conditions, the real challenge lies in correctly interpreting the business question.

Figure 4.21 shows the primary comparison operators used in business queries. SQL does support some advanced techniques, such as the IN (list), but you need to understand the basic operators before trying to tackle the complex ideas. The standard comparison operators (=, <, >) work exactly as you expect them to work from basic math. The not-equal operator seems a little tricky (<>), but you get used to it after a while.

The **BETWEEN** operator is a nice way to enter a range of values. It is commonly used for testing a date because you often need to see if a business date falls between two specified days. The condition will match any row where the value is between or equal to the start and end value. For example, to retrieve rows of sales data that occurred in January 2009, you would use: WHERE SaleDate BE-

Figure 4.20

Displaying a query built in Design view. Microsoft Access always uses the full name of the column which includes the table name. It also puts too many parentheses in the conditions. But, the OR connector is easy to read.

```

SELECT Customers.CustomerID, Customers.Name, Customers.Phone,
Customers.City, Customers.AccountBalance
FROM Customers
WHERE (((Customers.City)="Denver"))
OR (((Customers.City)="Chicago"));

```

Operator	Meaning	Examples
=	Equals	City='Denver' Salary=60000
< >	Less than Greater than	Salary < 60000 Sales > 15000
< >	Not equal	City < > 'Denver'
BETWEEN	Between x and y	SaleDate BETWEEN '01-Jan-2012' AND '28-Feb-2012' Sales BETWEEN 10000 AND 20000
LIKE	Simple pattern matching % or * matches any characters _ or ? matches one	LastName LIKE 'J%' ProductID LIKE 'BL_ _DR_ _ _'
Null	Missing data	City Is Null
NOT	Negation	Not City = 'Denver'

Figure 4.21

Query conditions. The standard comparisons are easy to understand. BETWEEN is most useful for dates but can be used in other cases. Pattern matching with LIKE is used for simple searches of text data. Be careful to use Is Null when searching for missing data.

TWEEN '01-Jan-2009' AND '31-Jan-2009'. However, note that date values are sometimes tricky to enter into a DBMS. The format used here will work on most systems, but Microsoft Access uses pound signs (#) instead of quotation marks to signify dates: BETWEEN #01-Jan-2009# AND #31-Jan-2009#. Also, when you use the BETWEEN command remember that you must always have two values separate by the AND keyword. Figure 4.22 shows the use of the BETWEEN clause to obtain a list of the sales that took place in June. Figure 4.23 shows the results. Always look through some of the rows of data to verify that you entered the conditions correctly. The DBMS will almost always return results, but you have to inspect them to ensure it is selecting the rows you want to include.

The Is Null and Is Not Null operators are useful but have an important trick. Any value that was not entered is automatically assigned a missing value and can only be found by searching for Is Null. For example, if someone failed to enter the name of the city the customer lives in, you would search: WHERE City Is Null. The important trick is that you cannot use City=NULL. Most systems will actually accept the equals sign, but will never return any matching rows—even if the City value is missing. You can stare at that last command a long time trying to figure out what went wrong, until you remember that you must use the IS operator instead of equality. Along the same lines, the condition City="" (quoted text with no value), will also fail. There is a difference between a zero-length string and a value that was never entered. This difference presents problems if someone happens to enter blank spaces for a text value. The DBMS will often display missing values as blank spaces, but they are stored differently internally.

The LIKE command is a useful tool but it can be difficult to understand at first. Essentially, you are specifying a search pattern or filter. Only rows that match that condition will be displayed. The LIKE command uses two special charac-

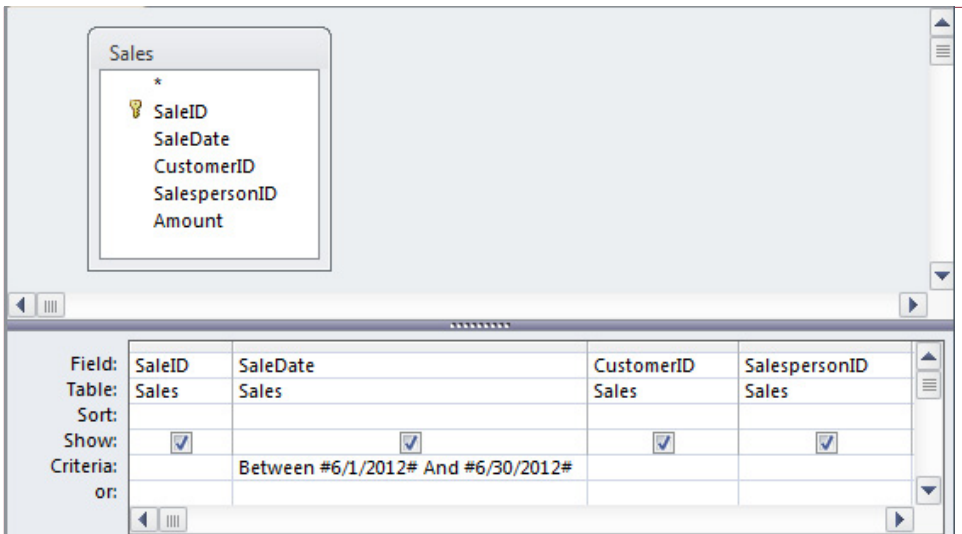


Figure 4.22

The BETWEEN clause. This query uses BETWEEN to retrieve sales that occurred in June.

ters to form patterns. The SQL standard states that a percent sign (%) will match any other characters, including no characters; and an underscore (_) will match any single character. Microsoft Access uses an asterisk (*) and question mark (?) to create the same patterns. You can see the power of pattern matching even for simple business questions. For example, to list the customers with a last name that starts with B, you would use: WHERE LastName LIKE 'B%' (or WHERE LastName LIKE "B*" in Access). The B in the pattern requires the name to have that letter in the starting position, followed by any other characters (including none). As another example, the condition WHERE Description LIKE '%smart%' will

Figure 4.23

Date results. Verify that the rows returned match the condition you desire. The DBMS will rarely tell you if you make a mistake. It will almost always return results, so you need to check to ensure you entered the conditions correctly.

	SaleID	SaleDate	CustomerID	Salesperson	Amount
	211	6/9/2012	44453	255	\$2,800.00
	213	6/9/2012	44453	255	\$12.00
	215	6/9/2012	87535	887	\$62.00
	280	6/27/2012	28764	663	\$1,036.00
	285	6/15/2012	44453	887	\$100.00
*	0	7/6/2011	0	0	\$0.00

Record: 1 of 5 No Filter Search

SUM	total value of items
AVG	average of values
MIN	minimum value
MAX	maximum value
COUNT	number of rows
STDEV	standard deviation
VAR	variance of items

Figure 4.24

The standard aggregation functions available in SQL and QBE. These functions operate on multiple rows of data at a time. SUM, AVG, and COUNT are the most-used functions in business.

search for any product that contains the word “smart” anywhere within the description. The word smart must exist, but can be preceded or followed by any other letters. The single-character search is used less often, but can be useful if you have specially-formatted codes. For instance, many companies create product codes that describe the product. Perhaps the first two letters are a color code, followed by a size code, a category, and a three-digit identifier, such as: BL09DR293 to represent a blue (BL) dress (DR) of size 9 (09) with an product identifier of 293. A salesclerk could search the inventory for all blue dresses using the pattern: WHERE ProductID LIKE ‘BL__DR___’. You need to know the details of how the product codes are created to build these conditions. In other cases, it is probably easier to search for items by using other columns of data, such as: (ProductType=’Dress’) AND (Color=’Blue’).

It is generally easy to create any individual condition. The process becomes more difficult, with more chances for error, when you are given many conditions at the same time. The key is to be patient and write one condition at a time and test the values one at a time. As you will see in the next section, it is important to test the conditions with a simple SELECT statement so you can ensure that the query is retrieving only the rows you need.

Computations

Many business questions involve totals or other calculations. All database systems have some mechanism to perform simple calculations. The calculations are usually simpler than those found in spreadsheets, but they are easier to use and can operate on millions of rows of data.

Query processors can perform two types of computations: (1) Compute totals across multiple rows of data, or (2) Use arithmetic and functions to operate on data one row at a time. You can also combine these two methods, but it is best to treat them separately for now. You often need to compute totals or average of business data. Figure 4.24 shows the aggregation functions commonly provided by SQL and QBE. SUM, AVG, and COUNT are the most-used functions in business.

SUM computes the total value of a numeric column. Essentially, it looks at each row of data that matches the WHERE clause and adds up the values stored in each row. Similarly, AVG works only on numeric columns, but it computes the simple average. COUNT is slightly trickier. It simply counts the number of rows of data that would be displayed, so it works with any column or even without a

Technology Toolbox: Building Forms in Access

Problem: Every month you have to create a report that uses several new pieces of data. You have an assistant to help enter data, but need to create a system that is simple to use.

Tools: Most DBMSs have a forms tool to quickly generate forms. In Microsoft Access, the Form Wizard is easy to use, as long as the tables are defined correctly.

It is relatively easy to make forms to enter and edit data for a single row at a time. You can quickly build a form to edit data for a single customer. Start the Form Wizard, select the Customer table and all of its columns by moving them to the right-side column. Work through the remaining selections by choosing design options. If you do not like the result, you can always delete the form and start over. When the form is complete, use it to enter data for a couple of customers.

The Form Wizard can build more complex forms, such as the Sale form that includes data from the tables: Sale, Customer, SaleItem, and Item. You need to be cautious in choosing columns. Start the Form Wizard and choose all columns from the Sale table. Then choose most of the columns from the SaleItem table, but do not include the SaleID. The SaleItem table represents the repeating section. If you include the SaleID, it will waste space by displaying the same value on every row. Next, choose columns from the Customer table, but do not include the CustomerID. You should not include the CustomerID column from the Customer table, because you will not use this form to add new customers. Keep your forms relatively simple, so each form has a clear purpose. Finally, select any desired columns from the Item table (Description, Color, Price, and so on). Once again, do not include the ItemID column from the Item table, because you will not use this form to add items. Once the columns have been selected, follow the prompts to finish building the form.

To make the form easier to use, you will want to switch to Design View and rearrange the items on the form so they are grouped in a layout that is easier to read.

Look at some of the problems with the form. To enter a new sale, you will have to enter the CustomerID. Did you memorize all of the CustomerID values? Instead, delete the CustomerID text box. Select the Combo box in the Toolbox and click on the form where the CustomerID box used to sit. Follow the Wizard prompts to select the Customer table; choose at least the CustomerID and Name columns. On the last screen, make sure you mark the option to store the result in the CustomerID column. Test the result. This procedure displays all of the customers in the drop-down box. When you select a customer, the corresponding CustomerID is transferred to the CustomerID column in the Customer table.

Quick Quiz:

1. Create a simple customer form and enter data to test it.
2. Create a basic order form and add a combo box to select customers.

Amount
\$197.54
\$526.76
\$353.76
\$153.00
\$863.39
\$255.93

Figure 4.25

AccountBalance column from the Customer table.

column name: Count(*). Although the functions are similar, you must be cautious when choosing them. It can be difficult to determine when you should use SUM instead of COUNT. Figure 4.25 shows the AccountBalance column for the rows of data in the Customer table. Due to the small amount of data, you can easily verify that there are 6 rows of data.

Figure 4.26 shows the QBE query and the values returned by the COUNT, AVG, and SUM functions. A few seconds with a calculator will convince you that the sum is correct. Always remember that Avg and Sum work only on numeric data because they compute the total of the values stored within each row. You have to carefully evaluate any business question to decide if it requires adding numbers or simply counting rows. The VAR and StDev functions are for simple statistical calculations and compute the variance and standard deviation of the data in the chosen column, using n-1 weighting. The older versions of the SQL query language did not specify standard names for these functions so they are spelled differently depending on the DBMS you are using. The more advanced

Figure 4.26

Count versus Sum. Count returns the number of rows that would be returned with a SELECT statement. Sum adds up the values within each row.

The screenshot shows a database query editor window. On the left, a tree view shows the 'Customers' table with fields: CustomerID, LastName, Phone, Street, City, and AccountBalance. The main area shows a query grid with the following structure:

Field:	AccountBalance	AccountBalance	AccountBalance
Table:	Customers	Customers	Customers
Total:	Count	Avg	Sum
Sort:			
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Criteria:			
or:			

Count	Avg	Sum
6	\$391.73	\$2,350.38

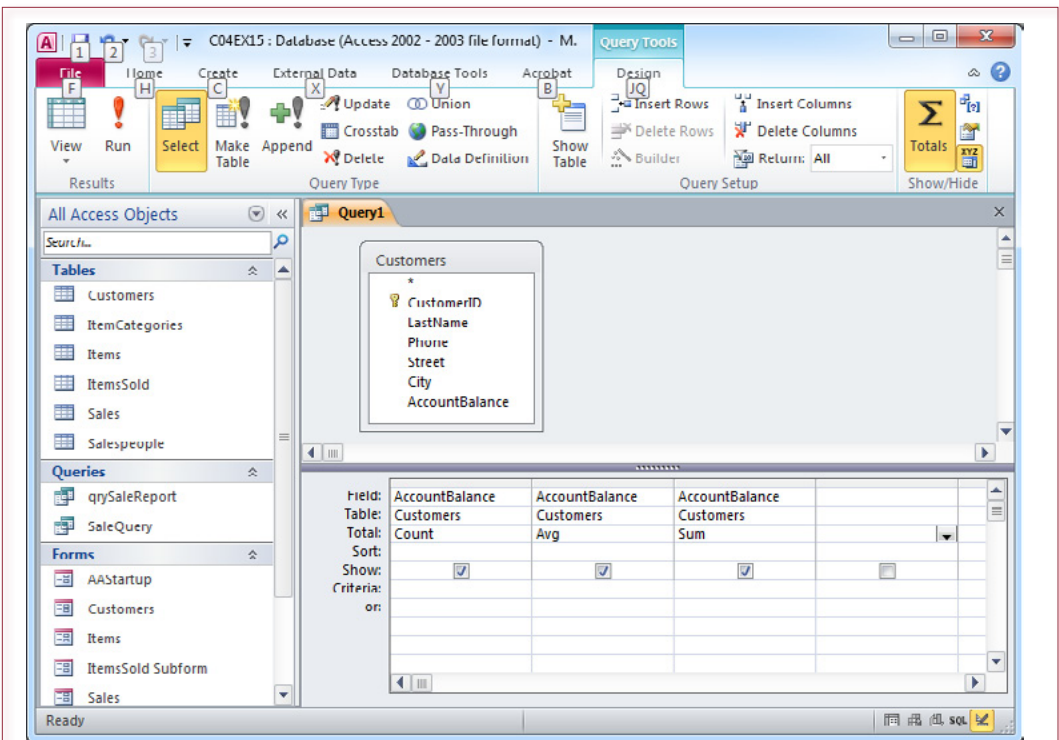


Figure 4.27

QBE for aggregation. In Access, you select the table and columns. Clicking the summation button (sigma) adds the Total row to the grid, where you can pick the Count and Avg functions.

(more expensive) DBMSs support additional aggregation functions, and you can read about them in the vendor's documentation or help system.

Query by Example

Although most database management systems provide a means to compute totals and averages, there is no standard method for entering the commands. Typically, the commands are displayed on a menu. Access uses an extended grid, which is shown in Figure 4.27. The total row is enabled by clicking the summation (sigma) button. You then select the desired function on the Total row using the drop-down list in the appropriate column. The example shows how to get the number of customers and the average account balance.

Computations within a row are a little more complicated because you have to type everything instead of picking from a list. Consider the Items table that lists all of the products for sale. It contains a Category column and a Price column. You are told that all items in the Electronics category have a cost that is 70 percent of the listed price. You want to display the list of items in the Electronics category, their price, and their estimated cost. Figure 4.28 shows the query. Start a new query, choose the Items table, and display the Category and Price columns. Enter the Clothes condition to restrict the rows. Create a new column by typing the formula: $0.7 * \text{Price}$. When you move the cursor to another location, Access will add the brackets and enter Expr1 as the title of the new column. Change Expr1 to EstCost, but be careful to leave the colon separator.

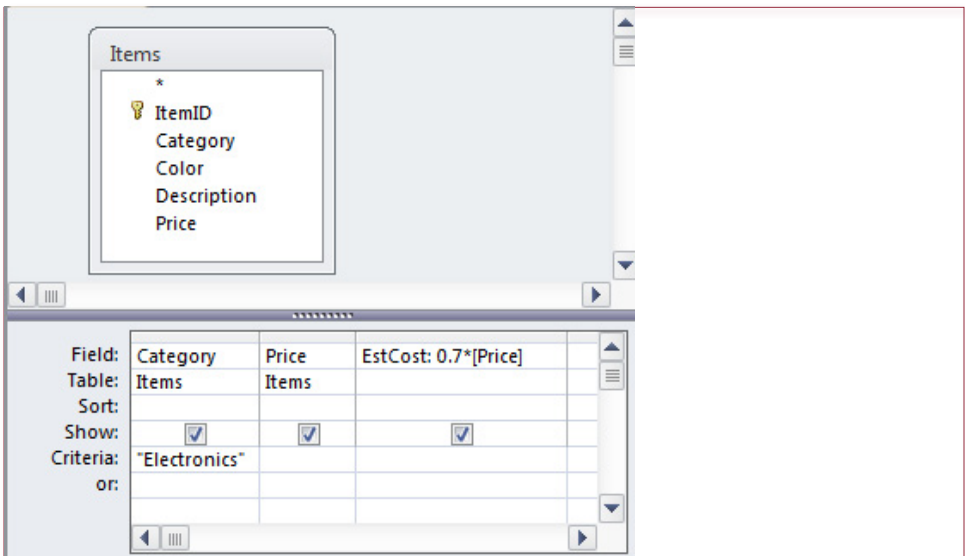


Figure 4.28

Row-by-row calculations. You can use basic arithmetic (+, -, *, /) and common math functions to create a new, computed column in a query. Just type in the formula and then change the text before the colon (EstCost) to be displayed as the new column heading.

Figure 4.29 shows the results of running the query. If desired, you can use the Round or Format functions to alter the calculation or the way it is displayed. The main thing to remember is that these computations are performed using only the data on one row. You can use data from multiple columns, but not from different rows.

SQL

Because QBE actually uses SQL, it should be clear that these same queries can be written directly in SQL. In many cases SQL is so easy that it is faster to type the simple SQL commands than it is to work through the QBE grid. Figure 4.30 shows the SQL command to count the number of customers and return the average amount owed. The functions are easy to use, but notice the use of the AS

Figure 4.29

Row-by-row calculations. You can use basic arithmetic (+, -, *, /) and common math functions to create a new, computed column in a query. Just type in the formula and then change the text before the colon (EstCost) to be displayed as the new column heading.

Category	Price	EstCost
Electronics	\$1,000.00	700
Electronics	\$400.00	280


```
SELECT Count(CustomerID) AS NCustomers,  
       Avg(AccountBalance) AS AverageOwed  
FROM Customers
```

Figure 4.30

Aggregation functions in SQL. The use of the function is straightforward, but note the use of the AS command to create a new name for the output column.

command to provide a new name for the column. If you leave out this name, the system will pick something that is unique but meaningless.

Row-by-row calculations are also straightforward in SQL. In fact, because you have to type the formula even within QBE, it is often easier to type the entire query within SQL. But, if you have complex WHERE clauses (or many tables), you might want to build the base query in QBE and switch to SQL to enter the formula. Figure 4.31 shows the SQL statement to list and estimate the cost of the items in the Electronics category.

Computing Subtotals

One of the most powerful features of SQL is its ability to create subtotals by adding one line to the SELECT statement. Many business questions require the computation of subtotals. In fact, it is difficult to find business questions that do not require subtotals. A subtotal is an aggregate calculation (usually SUM) that you want to perform for all members within a group. Common business examples include: Find the customer with the most sales, or the employee who sold the most last month, or identify the product or category that is the best or worst seller. Subtotals are also used whenever you want to compute totals by time period, such as total costs per month or sales this year compared to last year. Each of these questions requires adding up the value for each element specified (customer, employee, product category, month, year, and so on). It is difficult to compute these subtotals by hand. It is time-consuming even if you write your own program code. First you have to sort the data by the selected attribute to group the values together, then you find the break points where the value switches (such as from James to Jones). SQL (and QBE) solve this problem with one simple line: GROUP BY.

SQL

It is easier to see how subtotals are created by using SQL. Once you understand the process, you can use QBE to define the calculations. The first point to remember is that you want to see subtotals—so you have to use the aggregation functions in the output (SELECT) line. Figure 4.32 shows how to define a basic subtotal

Figure 4.31

Row-by-row computations in SQL. Simply type the formula as a new column and use the AS keyword to give the column a meaningful name.

```
SELECT Category, Price, 0.7*Price AS EstCost  
FROM Items  
WHERE (Category="Electronics")
```



```
SELECT City, Sum(AccountBalance) AS SumOfAccountBalance
FROM Customers
GROUP BY City
```

Figure 4.32

Subtotals by City. The output line contains a SUM function and the attribute you want to group. Then add the GROUP BY City line to tell the query parser what you want to see.

using SQL. The first two lines are similar to what you have already done. You are asking to see the name of the City and the total of the AccountBalance column. The third line tells the query processor that you want the subtotal computed for each value of city. The nice part is that you do not need to know which cities exist in the database. It automatically sorts and groups the data and then reports the total for each of those cities.

Figure 4.33 shows the total amount due by each city. The query processor automatically sorts the data by the City column because that is how it computes the subtotals. More commonly, you would add an ORDER BY line to sort by the subtotal—so you can find the city where customers owe you the most money.

Take a couple of minutes and study the SQL statement. You will need to use it many times. The query itself is relatively simple—just remember to keep it that way. A common mistake is to add too many columns or extra tables. For GROUP BY queries, you must use the smallest number of tables and columns possible. The easiest way to create the query is to think about the output. For any business question, decide what you want to see. In this example, you want a list of cities and the total of the AmountDue for each city, so the SELECT statement contains only the City and Sum(AmountDue). Both columns are in the Customers table, so you list that on the FROM statement. The phrases “for each” or “group” in any business questions are clues that you need to add the GROUP BY line.

An important catch with basic SQL is that any query using GROUP BY can display only subtotals (or other aggregate functions). It cannot display the detail rows and then the subtotals. If you want both details and totals, you must write a second query. A newer version of the SQL standard does have the ability to display both detail rows and subtotals, but it is relatively complex and it is not supported by Microsoft Access. More importantly, if you truly want to see that much information, better tools exist to examine the data interactively, and these are covered in Chapter 9.

Figure 4.33

Subtotals by City. Each city is listed followed by the total amount owed by customers in that city. It is important that you include only these columns and nothing extra.

City	SumOfAccountBalance
Chicago	\$197.54
Denver	\$1,016.39
Miami	\$255.93
Phoenix	\$526.76
Seattle	\$353.76

Customers

CID	LastName	Phone	City	AccountBalance
12345	Jones	312-555-1234	Chicago	\$197.54
28764	Adamz	602-999-2539	Phoenix	\$526.76
29587	Smitz	206-656-7763	Seattle	\$353.76
33352	Sanchez	303-444-1352	Denver	\$153.00
44453	Kolke	303-888-8876	Denver	\$863.39
87535	James	305-777-2235	Miami	\$255.98

Salespeople

SPID	LastName	DateHired	Phone	Commission
255	West	5/23/75	213-333-2345	5
452	Thomas	8/15/94	213-343-5553	3
554	Jabbar	7/15/91	213-534-8876	4
663	Bird	9/12/93	213-225-3335	4
887	Johnson	2/2/92	213-887-6635	4

Items

ItemID	Category	Color	Description	Price
1154	Shoes	Red	Red Boots	\$100.00
2254	Clothes	Blue	Blue Jeans	\$12.00
3342	Electronics	Black	LCD-40 inch	\$1,000.00
7653	Shoes	Blue	Blue Suede	\$50.00
8763	Clothes	Black	Mens' Work Boots	\$45.00
9987	Electronics	Silver	Blu-Ray Player	\$400.00

Sales

SaleID	CID	SPID	SaleDate	Amount
117	12345	887	3/3/2012	\$1400.00
125	87535	663	4/4/2012	\$535.00
157	12345	554	4/9/2012	\$100.00
169	29587	255	5/5/2012	\$1800.00
178	44453	663	5/1/2012	\$12.00
188	29587	554	5/8/2012	\$1180.00
201	12345	887	5/28/2012	\$100.00
211	44453	255	6/9/2012	\$2800.00
213	44453	255	6/9/2012	\$12.00
215	87535	887	6/9/2012	\$62.00
280	28764	663	5/27/2012	\$1036.00
285	44453	887	6/15/2012	\$100.00

ItemsSold

SaleID	ItemID	Quantity
117	1154	2
117	3342	1
117	7653	4
125	1154	4
125	8763	3
157	7653	2
169	3342	1
169	9987	5
178	2254	1

Figure 4.34

Multiple tables. The true power of a database lies in the ability to combine data from multiple tables. Actual databases can have hundreds or thousands of related tables. Notice that each table is related to another table through matching columns. You should be able to draw lines between column labels that will connect each of the tables.

Joining Multiple Tables

The true strength of a database management system lies in its ability to combine data from several tables. Part of the Customer table is shown in Figure 4.34, with additional tables that show a list of sales to those customers and the salespeople involved. Notice that the tables were designed so they can be connected. For example, the Sales table can be connected to the Customers table by matching the CustomerID (abbreviated as CID to save space). The Sales table can be matched to the Salespeople table through the SalespersonID (SPID). Once you have joined the tables together, the database system retrieves and displays the data as if it were stored in one table.

The chief advantage to using multiple tables is that data is stored one time—even for one-to-many relationships. For example, each salesperson may be associated with many different sales. Instead of repeating the salesperson information on every order, you only need to include the salesperson's ID (SPID) number. Joining the tables together tells the DBMS to automatically look up the corresponding data from the Salespeople table.

Query by Example

Most people find that database systems that use graphical QBE commands to join tables together are much easier to use than straight SQL commands. With a DBMS like Access you join the tables together by pointing to the column name in one table and dragging it to the matching column in the other table. The DBMS displays the connection between the two columns. Whenever you want to retrieve data from more than one table, you must first join them together.

SQL

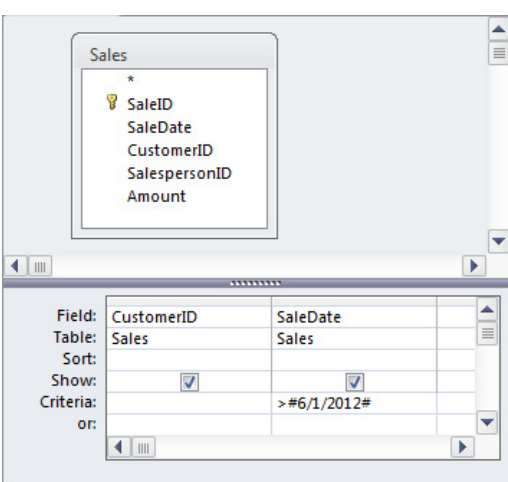
In SQL, connections between tables are typically made with the INNER JOIN clause in the FROM statement. For example, to join the Customers and Orders tables by equal customer numbers and get the combined data, use the command `SELECT * FROM Customers INNER JOIN Orders ON Customers.CustomerID = Orders.CustomerID`

Notice that both tables must be listed in the FROM statement. Always remember that if you list more than one table, the tables must be joined. The dot before the column (CustomerID) separates the table name from the column name (table.column). You can use this form any time you type in a column name, but it is only required when there might be confusion about which table the column is in. In the example, both tables have a column called CustomerID. To keep them straight, you have to specify which table you want to use.

Examples

You now have the basics to begin asking questions of the data. Start with an easy one. Which customers (CustomerID) have placed orders since June 1, 2012? The query and result are shown in Figure 4.35. Notice that customer number 44453 has placed two orders. Some systems will show you the order number twice; others will automatically delete the duplicates.

It can be difficult to remember each customer's number, so it is better to use the customer name and have the DBMS automatically look up the customer number. This second query is shown in Figure 4.36. Note that the Customer table is joined to the Orders table by the matching values of CustomerID. Be cautious when adding tables to a query and make sure that all of the tables are connected. If you



```
SELECT CustomerID, SaleDate
FROM Sales
WHERE SaleDate >= #6/1/2012#
```

CustomerID	SaleDate
44453	6/9/2012
44453	6/9/2012
87535	6/9/2012
28764	6/27/2012

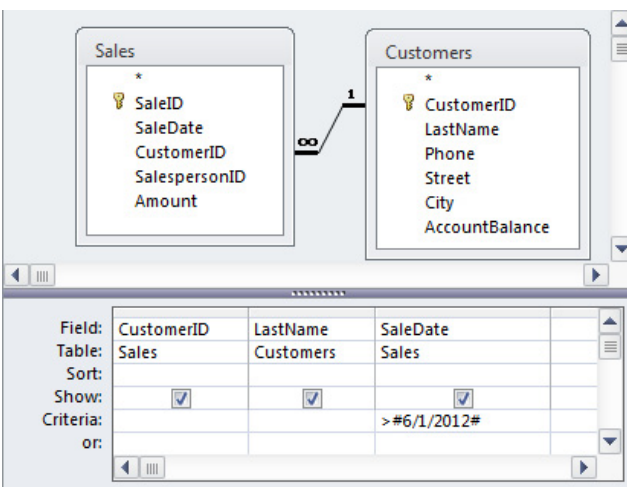
Figure 4.35

QBE and SQL. Which customers have placed orders since June 1, 2006? QBE and SQL are based on the same fundamental concepts. You build each query by asking the same four basic questions.

place two tables in the query with no connection, the query will still run—very slowly. The query processor will match every row in the first table with every row in the second table. For example, if each table has 1,000 rows, the resulting query would contain 1 million rows—and the results would be meaningless.

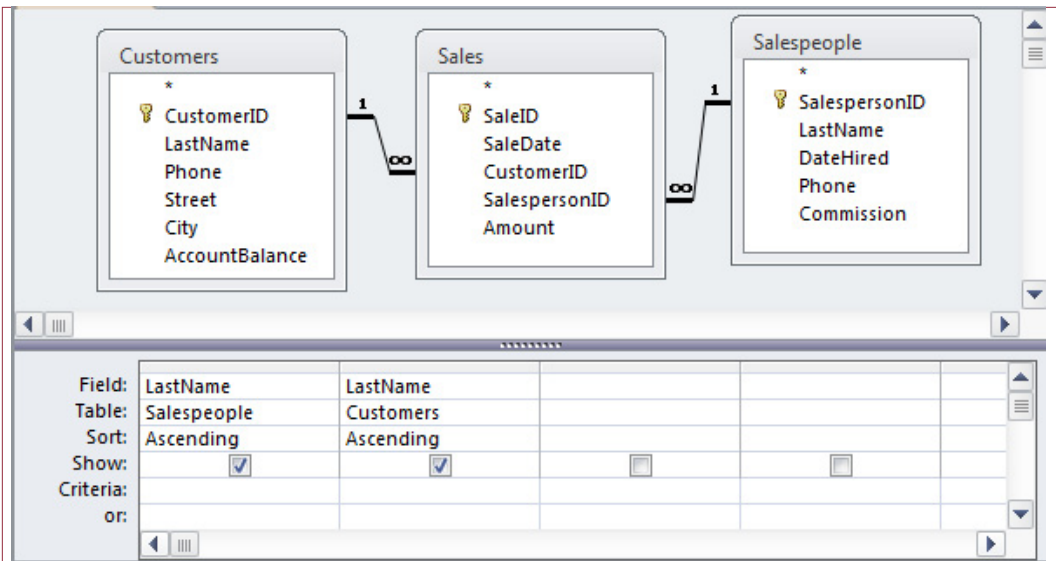
Figure 4.36

Multitable queries. Queries that use more than one table are slightly more complex. Because columns can have any name, you must tell the database system how the tables are connected. What are the names of the customers who placed orders since June 1?



```
SELECT DISTINCT
CustomerID,
Name, SaleDate
FROM Sales
INNER JOIN Customers
ON Sales.CustomerID
= Customers.CustomerID
WHERE SaleDate >= #6/1/2012#
```

CID	Name	OrderDate
28764	Adamz	6/27/2012
44453	Kolke	6/9/2012
44453	Kolke	6/15/2012
87535	James	6/9/2012



```

SELECT  DISTINCT Salespeople.LastNameName,
          Customers.LastName
FROM    Salespeople
INNER JOIN  (Customers INNER JOIN Orders ON
            Customers.CustomerID=Sales.CustomerID)
ON Salespeople.SalespersonID
= Sales.SalespersonID
ORDER BY  Salespeople.LastName

```

SalesName	Cust.Name
Bird	Adamz
Bird	James
Bird	Kolke
Jabbar	Jones
Jabbar	Smitz
Johnson	James
Johnson	Jones
Johnson	Kilke
West	Kolke
West	Smitz

Figure 4.37

Multitable queries with several joins. More complicated queries follow the same basic rules. Note that some database management systems can automatically switch displays between QBE and SQL. This feature is useful so that you can check the joins and the criteria to be sure they are being interpreted correctly. salespeople (sorted alphabetically) along with the names of customers who placed orders with that salesperson.

Now, try a more complicated query: List the salespeople (sorted alphabetically) with the names of the customers who placed orders with that salesperson. This question sounds difficult, but the command is easy when you join all three tables together. The query and the result are shown in Figure 4.37. Notice there is no entry for the salesperson (Zeke) who has no sales at this point.

Multiple Tables, GROUP BY, and WHERE

The basic concept of multiple tables is straightforward—once you have chosen the tables and joined them correctly, you can select columns from any of the tables. However, be cautious about using extra tables. Each query should use the minimum number of tables needed to answer the question. Extra tables can create problems and incorrect answers—particularly when computing totals.

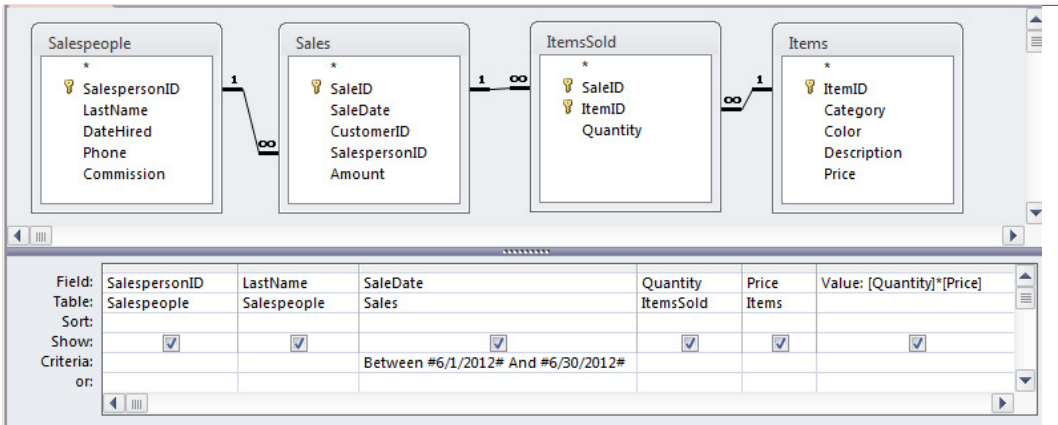


Figure 4.38

Subtotals with a WHERE condition. The WHERE condition is applied first and limits the rows used to compute the totals.

With the ability to use multiple tables, it is now time to look at more realistic questions. Consider the business question: Who are the top salespeople in June? The key to understanding this question is to recognize that the answer requires computing the total sales by employee in the month of June. Begin by creating a query that lists the sales in June along with the name of the salesperson.

As shown in Figure 4.38, you can add a WHERE condition to control the rows that will be used in the computations. The other important step is to create the Value column which multiplies price by quantity. Notice that this query requires four tables: One for the name of the salesperson, two for the Sales data, and one for the price of the items.

Figure 4.39 shows the rows from the initial query. Each row displays the sale of one item, although there can be multiple quantities of that item. Notice that Price and Quantity are displayed so you can verify that the Value column is computed correctly. When building queries, it is important to display intermediate results to

Figure 4.39

Initial query results to compute the value column. Each row represents the sale of one item. Price and quantity are temporarily displayed to verify the Value computation.

SalespersonID	Name	SaleDate	Quantity	Price	Value
255	West	6/9/2012	2	\$1,000.00	\$2,000.00
255	West	6/9/2012	5	\$50.00	\$250.00
255	West	6/9/2012	1	\$12.00	\$12.00
887	Johnson	6/9/2012	1	\$12.00	\$12.00
887	Johnson	6/9/2012	1	\$50.00	\$50.00

```

SELECT Salespeople.SalespersonID, Salespeople.Name,
       Sum([Quantity]*[Price]) AS [Value]
FROM Items INNER JOIN ((Salespeople
INNER JOIN Sales
    ON Salespeople.SalespersonID = Sales.SalespersonID)
INNER JOIN ItemsSold
    ON Sales.SalesID = ItemsSold.SaleID)
    ON Items.ItemID = ItemsSold.ItemID
WHERE (Sales.SaleDate Between #6/1/2012# And #6/30/2012#)
GROUP BY Salespeople.SalespersonID, Salespeople.Name
ORDER BY Sum([Quantity]*[Price]) DESC;

```

Figure 4.40

SQL query to compute subtotals with WHERE condition. Notice that the date condition is specified in a WHERE clause because that filters the rows that will be used in the computation.

ensure that computations are specified correctly. Because a WHERE condition is also used, the SaleDate values should also be checked to ensure the correct data is being displayed. If you make a mistake when specifying a WHERE condition, the only way to catch it is to look at the rows that are returned.

The next step is to remove the Quantity and Price columns and then compute the total of the Value column. Figure 4.40 shows the SQL statement that answers the question about total sales by salesperson in June. Note the Sum function in the SELECT clause which specifies the total computation. More importantly, observe that the WHERE clause holds the date condition (June). This condition filters the rows that will be used in the computations. Finally, the GROUP BY clause speci-

Figure 4.41

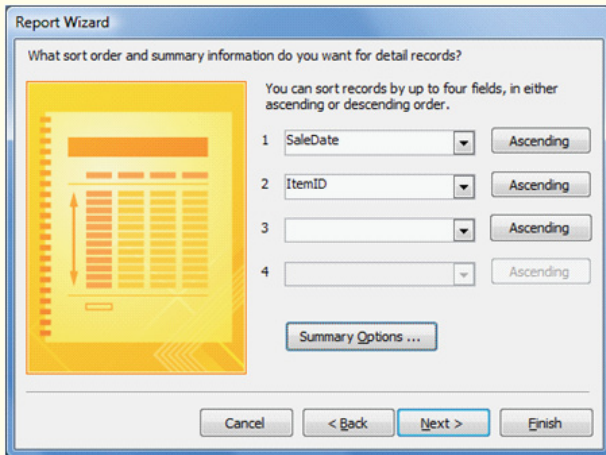
Initial attempt to compute subtotals. Click the Totals button and set the Sum option for the Value column. Notice that the other columns are automatically set to GROUP BY.

Field:	SalespersonID	LastName	SaleDate	Value: [Quantity]*[Price]	
Table:	Salespeople	Salespeople	Sales		
Total:	Group By	Group By	Group By	Sum	
Sort:					
Show:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Criteria:			Between #6/1/2012# And #6/30/2012#		
or:					

Technology Toolbox: Creating Database Reports

Problem: Your company has a large database that contains the information you need. However, you need to produce a relatively complex report every month based on data in that database.

Tools: A database Report Wizard can help you build reports with a visually oriented tool that makes it easy to create the layout and subtotals that you want.



If you need in-line computations, such as price * quantity, it is generally easiest to create a query that performs these simple calculations and selects the columns you will need on the report. Just create a query, test the data, and save it as a view. Then open the Report Wizard and select the saved view. Otherwise, you can select individual tables and choose the columns you want from each table.

The key to understanding reports is to use the groupings

and levels correctly. First, identify the data that you want to see at the most detailed level. For example, you might want to see individual sale items. In creating the underlying query, make sure you select the tables and columns necessary to display the detail level you need. Sometimes, you will want to use subtotals in the query, but many times you will simply let the report writer create the totals—it all depends on what you want to see at the detail level. The next decision is to identify what groupings and levels you need. For instance, you might want to see a list of items ordered by each customer, so you set a grouping by customer. Alternatively, you might want to see a list of orders by each customer, so you set groupings for customer and then for order.

The Form Wizard contains an option to compute totals for the numeric columns. The button to select this option is easy to miss, so look carefully. Once the Wizard creates the design, you will still have to set properties to improve the format and layout of the report. You can also add new text boxes to compute other values. For example, to create a total, just place a text box in the desired footer and enter the formula =Sum(Value), using whatever column you want to total.

Creating reports is detail work. It takes time to get the layout and the format right. It takes a sense of design to make the report look good. Remember that the goal is to present the important information so that it can be found quickly.

Quick Quiz:

1. Create a report that prints all of the items ordered by each customer.
2. Create a report that prints each customer, followed by the orders for that customer.
3. Create a report that displays a chart of total sales by customer.

SalespersonID	Name	SaleDate	Value
255	West	6/9/2012	\$2,250.00
255	West	6/10/2012	\$12.00
887	Johnson	6/9/2012	\$62.00

Figure 4.42

Unfinished query. GROUP BY with SaleDate results in two rows for Salesperson 255—one for each day.

fies that the totals are to be computed for each sales person. The ID and the name are both included because there is a chance that two employees have the same name. Including the ID ensures that employees with the same name will be treated separately.

Look at the SQL statement again and be sure that you understand the main points; then you can use the Access design grid to finish the query. Figure 4.41 shows the initial attempt. Remove the Quantity and Price columns. Use the totals button to add the Total row and select the Sum option for the Value column. Leave the other columns at the default GROUP BY option.

Figure 4.42 shows what happens when this initial query is executed. Look at the ID and Name columns. The question calls for computing totals for each salesperson, so the query should have only one row for each person. These results contain two rows for ID 255 (West). Why? Look back at the query grid, or better yet, look at the SQL statement. The query lists three columns as GROUP BY: ID, Name, and SaleDate. So the DBMS computed exactly what you asked for: The total value for each salesperson for each day. The second row for West arose because the sale took place on a second day. If the database contained a complete set of monthly data, the query would return different totals for every day in the month for each employee.

The solution to the problem is to remove the SaleDate from the GROUP BY condition. In the Access grid, this task is accomplished by changing the GROUP BY entry on the Total row to WHERE. Figure 4.43 shows the final query in the grid. Set the SaleDate row to Where and sort the Value in descending order to display the highest value at the top of the list. When you change the SaleDate to the WHERE entry, notice that the display checkbox is unset. Do not attempt to restore the check mark—that would tell the query to display both the individual detail rows and the totals. But Access, and the standard GROUP BY SQL query, cannot display details and totals at the same time. Run the query now and you will see that it returns exactly one row for each employee who participated in a sale in June. Remember this lesson. When computing subtotals that involve date conditions, you almost always want to move the date condition to the WHERE clause. The only exception is when you actually want to compute sales by day.

Views

There is one important feature of queries that you will find useful. Any query can be saved as a **view**. For example, if you have a complex query that you have to run

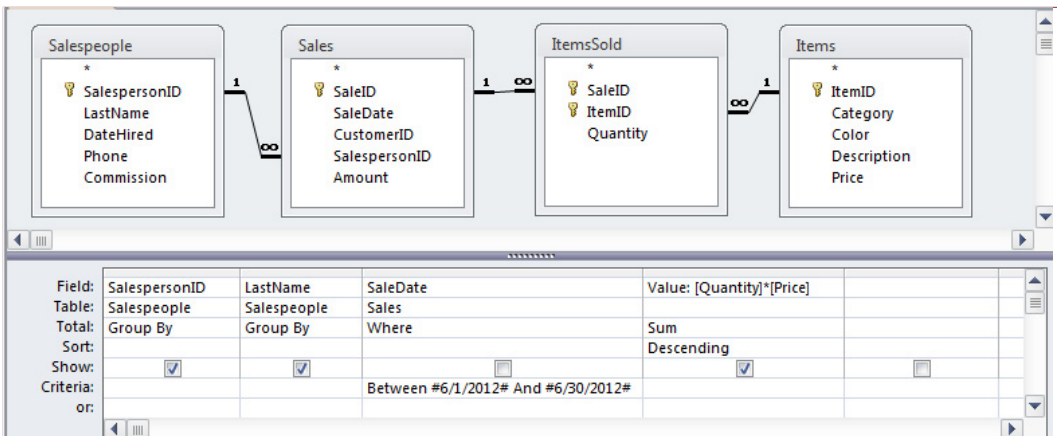


Figure 4.43

Correct version of the query grid. The WHERE identifier under the SaleDate column removes the SaleDate from the GROUP BY clause and places it in the WHERE clause.

every week, you (or a database specialist) could create the query and save it as a view with its own name. The important point is that the view can now be treated like any other table. In the example, you might define a view that combines the tables for customers, orders, and salespeople, and call it SalesOrders. Then, to get the total sales from customers in Miami, you run the query on the SalesOrders view and you no longer have to worry about joining tables because the query has already performed the step.

The advantage of views is that you can look at the data in different ways without having to duplicate the data. As a manager, you can create complex views so your employees can look up information using much simpler commands. Think of a view as a mirror. If you stand in front of a three-way mirror in a clothing store, you get different views of yourself although there is still only one person.

Converting Business Questions to Queries

The query grid and SQL are relatively easy to use. The real challenge lies in understanding the business question and correctly converting it into a query. In fact, this task demonstrates the true role of MIS. One of the most difficult tasks in applying information technology and creating systems is to identify the business needs and translate them into the computer realm. Students who become good at this translation, can solve problems quicker and become more successful.

The key to translating business questions into queries is to focus on the four questions posed at the start of this section: (1) What do you want to see? (2) What constraints are you given? (3) What tables are involved? And (4) How are the tables joined? And, it is usually easiest if you answer the questions in that order. In business, the second key is to understand exactly what data is stored in the database tables. Ultimately, you need to know what all of the tables mean and exactly what type of data is stored in each column. The good news is that once you have worked for a company for a couple of years, the terms and data will become

```

SELECT _____
FROM _____
INNER JOIN _____
WHERE _____
GROUP BY _____
ORDER BY _____

```

Figure 4.44

Primary SQL clauses. Write these down and fill in the blanks to start a new query. You can simplify the FROM and INNER JOIN parts by just listing the tables and drawing connector lines.

familiar to you. The bad news is that company databases can hold hundreds of tables and thousands of columns.

For longer queries, you should write down the basic SQL clauses as shown in Figure 4.44. As you read the business question, you fill in the blanks. Of course, you will need the relationship diagram or a list of tables in the database so you know the names of the tables and columns. Consider an example from the Rolling Thunder Bicycle company database. The business question: Which customers from California spent the most money on race bikes in 2010? The relationship diagram is shown in the Exercises at the end of this chapter, or you can open the database and examine the diagram or the tables directly.

The first step is to decide what you want to see. The business question calls for information about customers, so you should include at least the customer last name and perhaps their phone numbers. You have some flexibility to add other data but keep it simple at the beginning. You also want to see the total amount of money spent by each customer. The best way to represent that value is to look at the SalePrice of each bicycle that was purchased. This number ignores taxes and shipping costs which are not revenue to the firm, but managers might want these additional values in some cases. You can now write the SELECT clause:

```
SELECT LastName, Phone, Sum(SalePrice) As TotalSales
```

Now look at the constraints you are given. Reread the business question and you should see three constraints: California, 2010, and race bikes. Each of these becomes a condition in the WHERE clause. Since all three conditions must be met at the same time, they are all connected with AND clauses. Look through the list of columns in the Bicycle table and you can find the appropriate columns to use (SaleState, OrderDate, and ModelType). If you are unsure about the data held in each column, you can display a few rows of data from the table to check the values. You can now write the WHERE clause.

```

WHERE (SaleState="CA")
AND (OrderDate BETWEEN #1/1/2010# AND #12/31/2010#)
AND (ModelType="Race")

```

Because you need to compute the total value for each customer, you must add a GROUP BY clause. This clause must list each column in the SELECT clause that is not part of an aggregation function.

```
GROUP BY LastName, Phone
```

Now that you know all of the columns needed, you can see which tables were used. In this case, the Bicycle and Customer tables, so you can write the FROM clause.

```
FROM Bicycle, Customer
```

To avoid memorizing the INNER JOIN syntax, you will eventually use the query grid tool to enter the tables and build the joins graphically. But, you should still verify that the tables will be connected. On the INNER JOIN clause, draw lines between the tables to indicate how they are connected.

```
INNER JOIN Bicycle.CustomerID = Customer.CustomerID
```

In some cases, you will initially list tables that are not directly connected. For instance, you might use the Bicycle table and the Components table in a query. But, remember that every table must be connected when you build a query. In these situations, go back to the relationship diagram and look for an intermediary table that connects the ones you need. In this example, the BikeParts table connects to both the Bicycle and Component table, so you must add it to your FROM clause.

Finally, you can sort the data to organize it to answer the question. In this case, you want to know which customers spent the most money, so sort the results in descending order of the total.

```
ORDER BY Sum(SalePrice) DESC
```

Notice that this query will give you multiple rows. The best customers will be listed at the top, but does that really answer the question? Technically, you could cut off the list to display just the values at the top (SELECT TOP 5 LastName, Phone, ...). However, as a businessperson, you probably want to see more than just the first row. You generally want to know if there are ties, how close the second-place person is to the first, and so on. So, in reality, when a businessperson asks for the “best” or “worst” he or she generally means to sort the list and show at least the first few top items in the list.

The other big hint in building queries is that sometimes they get very complex and it is difficult to determine how to approach the problem. In these situations, always start with the parts you know how to do. For example, you generally know how to enter conditions, so write those down. Then figure out roughly what you want to see for output and write those columns down on the SELECT line. Build the query in pieces and test it to ensure that you are obtaining exactly the rows you need. If necessary, work with one table at a time and add them one at a time, verifying each step by running the query. If you need computations, verify that you know how to do them, and add them to the SELECT line. Hold off on the totals until the end because you cannot tell if the totals are correct when looking at just one number.

Designing a Database (optional)

How do you create a new database? Database management systems are powerful tools with the ability to present data in many ways. They are used by managers to answer many different types of questions. However, this flexibility is not automatic. Databases need to be carefully designed; otherwise, managers will not be able to get the information they need. Poor design also leads to unnecessary duplication of data. Duplication wastes space and requires workers to enter the same data several times. **Normalization** is an important technique to design databases.

To understand the process of normalization, consider the example of retail sales. You begin by thinking about who will be using the database and identifying what data they will need. Consider the situation of the salespeople. They first identify the customer then record each item being purchased. The computer

The screenshot shows an order form with the following fields and data:

- SaleID: 117
- SaleDate: 3/3/2012
- Customer: Jones (with an Edit button)
- Salesperson: Johnson
- Phone: (312) 555-1234

The ItemsSold table contains the following data:

ItemID	Category	Description	Price	Quantit	Extended
1154	Shoes	Red Boots	\$100.00	2	\$200.00
3342	Electronics	LCD-40 inch	\$1,000.00	1	\$1,000.00
7653	Shoes	Blue Suede	\$50.00	4	\$200.00
*					

Record: 1 of 3 (with navigation icons and a search field). Total: \$1,400.00

Record: 1 of 12 (with navigation icons and a search field)

Figure 4.45

The order form is used in almost any firm. We need to determine the best way to store the data that is collected by this form.

should then calculate the amount of money due along with any taxes. Figure 4.45 shows a sample input screen that might be used.

The key design point is that you will need multiple tables to store the data. If you try to cram all of it into a single table, you will end up with unnecessary duplication of data and plenty of problems when you try to delete or insert new data. Each entity or object on the form will be represented by a separate table. For this example, there are five objects on the form: Customers, Salespeople, Items, Sale, and ItemsSold..

Before explaining how to derive the five tables from the form, you need to understand some basic concepts. First, remember that every table must have a primary key. A primary key is one or more columns that uniquely identify each row. For example, you anticipate problems with identifying customers, so each customer will be assigned a unique ID number. Similarly, each item is given a unique ID number. There is one drawback to assigning numbers to customers: you cannot expect customers to remember their number, so you will need a method to look it up. One possibility is to give everyone an ID card imprinted with the number—perhaps printed with a bar code that can be scanned. However, you still need a method to deal with customers who forget their cards. It is usually better to build a method to lookup customers by name.

The second aspect to understand when designing databases is the relationships between various entities. First, observe that there are two sections to the form: (1) the main sale that identifies the transaction, the customer, the salesperson, and the date, and (2) a repeating section that lists the items being purchased. Each customer can buy several different items at one time. There is a **one-to-many** relationship between the Sale and the ItemsSold sections. As you will see, identifying one-to-many relationships is crucial to proper database design.

In some respects, designing databases is straightforward: There are only three basic rules. However, database design is often interrelated with systems analysis.

Reality Bytes: Internationalization: ZIP Codes

Databases often contain addresses (of customers, suppliers, employees, etc.), that typically use zip codes. In the United States, zip codes typically consist of five digits, so it is tempting to set up a ZipCode column that restricts input to five integers. However, bulk mail is often cheaper if it uses nine-digit zip codes (zip + 4).

Even more important, if your addresses might someday include international data, you have to be more careful in column restrictions. For instance, Canadian and British postal codes include alphabetic characters in the middle of the code. Some areas (such as Hong Kong) do not use any postal codes.

Similarly, when you set up databases that include phone numbers, be sure to allocate enough space for area codes. If international phone numbers will be listed, you need to add three extra digits on the front for the international country code.

In most cases, you are attempting to understand the business at the same time the database is being designed. One common problem that arises is that it is not always easy to see which relationships are one-to-many and which are one-to-one or many-to-many.

Notation

It would be cumbersome to draw pictures of every table that you use, so you usually write table definitions in a standard notation. The base customer table is shown in Figure 4.46, both in notational form and with sample data.

Figure 4.46 illustrates another feature of the notation. You denote one-to-many or repeating relationships by placing parentheses around them. Figure 4.47 represents all the data shown in the input screen from Figure 4.45. The description is created by starting at the top of the form and writing down each element that you encounter. If a section contains repeating data, place parentheses around it. Preliminary keys are identified at this step by underlining them. However, you might have to add or change them at later steps. You can already see some problems with trying to store data in this format. Notice that the same customer name, phone, and address would have to be entered several times.

Remember that some repeating sections are difficult to spot and might consist of only one column. For example, how many phone numbers can a customer have? Should the Phone column be repeating? In the case of the retail store, probably not, because you most likely want to keep only one number per customer. In other businesses, you might want to keep several phone numbers for each client. Data normalization is directly related to the business processes. The tables you design depend on the way the business is organized.

First Normal Form

Now that you have a way of writing down the assumptions, it is relatively straightforward to separate the data into tables. The first step is to split out all repeating sections. Think about the problems that might arise if you try to store the repeating data within individual cells. You will have to decide how many rows to set aside for storage, and you will have to write a separate search routine to evaluate data within each cell, and complications will arise when inserting and deleting data. Figure 4.48 illustrates the problem.

Table name
Table columns

↙
↖
↘

Customer(CustomerID, LastName, Phone, Street, City, AccountBalance)

CID	Last Name	Phone	Street	City	Balance
12345	Jones	(312) 555-1234	125 Elm Street	Chicago	\$197.54
28764	Adamz	(602) 999-2539	938 Main Street	Phoenix	\$526.76
29587	Smitz	(206) 676-7763	523 Oak Street	Seattle	\$353.76
33352	Sanchez	(303) 444-1352	999 Pine Street	Denver	\$153.00
44453	Kolke	(303) 888-8876	909 West Avenue	Denver	\$863.39
87535	James	(305) 777-2235	374 Main Street	Miami	\$255.93

Figure 4.46

Notation for tables. Table definitions can often be written in one or two lines. Each table has a name and a list of columns. The column (or columns) that makes up the primary key is underlined.

The answer to this problem is to pull out the repeating section and form a new table. Then, each item purchased by a customer will fill a new row. Figure 4.48 uses the notation to show how the table will split. Notice that whenever you split

Figure 4.47

Converting to notation. The basic rental form can be written in notational form. Notice that repeating sections are indicated by the inner parentheses. If we actually try to store the data this way, notice the problem created by the repeating section: Each time a customer checks out a video we have to reenter the phone and address.

SaleForm(SaleID, SaleDate, CustomerID, Phone, Name, Street, (ItemID, Quantity, Description, Price))

↙
↖
↘

Repeating Section
Causes duplication

SaleID	SaleDate	CID	Name	Phone	Street	ItemID	Qty	Description	Price
117	3/3/2012	12345	Jones	(312) 555-1234	125 Elm Street	1154	2	Red Boots	\$100.00
117	3/3/2012	12345	Jones	(312) 555-1234	125 Elm Street	3342	1	LCD-40 inch	\$1,000.00
117	3/3/2012	12345	Jones	(312) 555-1234	125 Elm Street	7653	4	Blue Suede	\$50.00
125	4/4/2012	87535	James	(305) 777-2235	374 Main Street	1154	4	Red Boots	\$100.00
125	4/4/2012	87535	James	(305) 777-2235	374 Main Street	8763	3	Men's Work Boots	\$45.00
157	4/9/2012	12345	Jones	(312) 555-1234	125 Elm Street	7653	2	Blue Suede	\$50.00
169	5/6/2012	29587	Smitz	(206) 676-7763	523 Oak Street	3342	1	LCD-40 inch	\$1,000.00
169	5/6/2012	29587	Smitz	(206) 676-7763	523 Oak Street	9987	2	Blu-Ray Player	\$400.00
178	5/1/2012	44453	Kolke	(303) 888-8876	909 West Avenue	2254	1	Blue Jeans	\$12.00
188	5/8/2012	29587	Smitz	(206) 676-7763	523 Oak Street	3342	1	LCD-40 inch	\$1,000.00
188	5/8/2012	29587	Smitz	(206) 676-7763	523 Oak Street	8763	4	Men's Work Boots	\$45.00
201	5/23/2012	12345	Jones	(312) 555-1234	125 Elm Street	1154	1	Red Boots	\$100.00

SaleID	SaleDate	CID	Name	Phone	Street	ItemID, Quantity, Description, Price
117	3/3/2012	12345	Jones	312-555-1234	125 Elm Street	1154, 2, Red Boots, \$100.00 3342, 1, LCD-40 inch, \$1,000.00 7653, 4, Blue Suede, \$50.00
125	4/4/2012	87535	James	305-777-2235	374 Main Street	1154, 4, Red Boots, \$100.00 8763, 3, Men's Work Boots, \$45.00
157	4/9/2012	12345	Jones	312-555-1235	125 Elm Street	7653, 2, Blue Suede, \$50.00
169	5/6/2012	29587	Smitz	206-676-7763	523 Oak Street	3342, 1, LCD-40 inch, \$1,000.00 9987, 2, Blu-Ray Player, \$400.00
178	5/1/2012	44453	Kolke	303-888-8876	909 West Ave.	2254, 1, Blue Jeans, \$12.00
188	5/8/2012	29587	Smitz	206-676-7763	523 Oak Street	3342, 1, LCD-40 inch, \$1,000.00 8763, 1, Men's Work Boots, \$45.00
201	5/23/2012	12345	Jones	312-555-1234	125 Elm Street	1154, 1, Red Boots, \$100.00

Figure 4.48

A table that contains repeating sections is not in first normal form. Each table cell can contain only basic data. Storing it in repeating form makes it difficult to search, insert, and delete data. This version is not in first normal form.

a table this way, you have to bring along the key from the prior section. Hence, the new table will include the SaleID key as well as the ItemID key. When a table contains no repeating sections, you say that it is in first normal form.

Second Normal Form

Even if a table is in first normal form, there can be additional problems. Consider the SaleLine table in Figure 4.49. Notice the two components to the key: SaleID and ItemID. The nonkey items consist of the Quantity, Description, and Price of the item. If you leave the table in this form, consider the situation of selling a new item. Every time an item is sold it will be necessary to reenter the Description and list Price. It means that you will be storing the description every time an item is sold. Popular items might be sold thousands of times. Do you really want to store the description (and other data) each time?

The reason you have this problem is that when the SaleID changes, the item description stays the same. The description depends only on the ItemID. If the Price represents the list price of the item, the same dependency holds. However, what if the store offers discounts on certain days or to specific customers? If the price can vary with each transaction, the price would have to be stored with the SaleID. The final choice depends on the business rules and assumptions. Most companies resolve the problem by creating a list price that is stored with the item and a sale price that is stored with the transaction. However, to simplify the problem, stick with just the list price for now.

When the nonkey items depend on only part of the key, you need to split them into their own table. Figure 4.50 shows the new tables. When each nonkey column in a table depends on the entire key, the table is in second normal form.

Third Normal Form

Examine the SaleForm2 table in Figure 4.49. Notice that because the primary key consists of only one column (SaleID), the table must already be in second normal form. However, a different problem arises here. Again, consider what happens when you begin to collect data. Each time a customer comes to the store and buys something there will be a new transaction. In each case, you would have to record

SaleForm(SaleID, SaleDate, CID, Phone, Name, Street, (ItemID, Quantity, Description, Price))

SaleForm2(SaleID, SaleDate, CustomerID, Phone, Name, Street)

SaleID	SaleDate	CID	Name	Phone	Street
117	3/3/2012	12345	Jones	(312) 555-1234	125 Elm Street
117	3/3/2012	12345	Jones	(312) 555-1234	125 Elm Street
117	3/3/2012	12345	Jones	(312) 555-1234	125 Elm Street
125	4/4/2012	87535	James	(305) 777-2235	374 Main Street
125	4/4/2012	87535	James	(305) 777-2235	374 Main Street

Note replication

SaleLine(SaleID, ItemID, Quantity, Description, Price)

SaleID	ItemID	Quantity	Description	Price
117	1154	2	Red Boots	\$100.00
117	3342	1	LCD-40 inch	\$1,000.00
117	7653	4	Blue Suede	\$50.00
125	1154	4	Red Boots	\$100.00
125	8763	3	Men's Work Boots	\$45.00
157	7653	2	Blue Suede	\$50.00
169	3342	1	LCD-40 inch	\$1,000.00
169	9987	2	Blu-Ray Player	\$400.00
178	2254	1	Blue Jeans	\$12.00
188	3342	1	LCD-40 inch	\$1,000.00
188	8763	4	Men's Work Boots	\$45.00
201	1154	1	Red Boots	\$100.00

Note replication

Figure 4.49

Splitting a table to solve problems. Problems with repeating sections are resolved by moving the repeating section into a new table. Be sure to include the old key in the new table so that you can connect the tables back together.

the customer name, address, phone, city, and so on. Each entry in the transaction table for a customer would duplicate this data. In addition to the wasted space, imagine the problems that arise when a customer changes a phone number. You might have to update it in hundreds of rows.

The problem in this case is that the customer data does not depend on the primary key (SalesID) at all. Instead, it depends only on the CustomerID column. Again, the solution is to place this data into its own table. Figure 4.51 shows the split. Splitting the table solves the problem. Customer data is now stored only one time for each customer. It is referenced back to the Rentals table through the CustomerID. The same rule applies to Salespeople, resulting in the fifth table.

The five tables you created are listed in Figure 4.52. Each table is now in third normal form. It is easy to remember the conditions required for third normal form. First: There are no repeating groups in the tables. Second and third: Each nonkey column depends on the whole key and nothing but the key.

Note in that if the Customers table contains complete address data, including ZIP Code, you could technically split the Customers table one more time. Because ZIP codes are uniquely assigned by the post office, the city and state could be determined directly from the ZIP code (they do not depend on the CustomerID). In fact, most mail order companies today keep a separate ZipCode table for that very reason. For our small retail firm, it might be more of a nuisance to split the table. Although you can purchase a complete ZIP code directory in computer form, it is

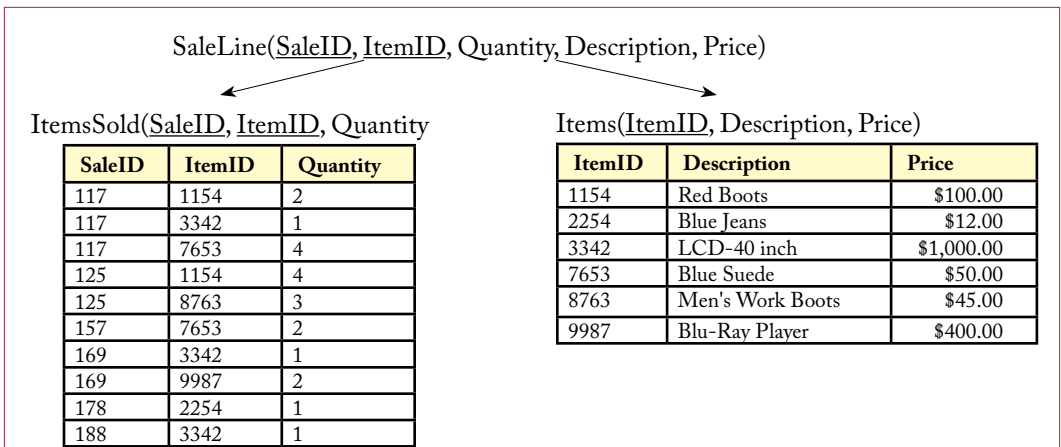


Figure 4.50

Second normal form. Even though the repeating sections are gone, we have another problem. Every time the ItemID is entered, the description has to be reentered, which wastes a lot of space. There is a more serious problem: if no one has purchased a specific item yet, there is no way to find its description or price since it is not yet stored in the database. Again, the solution is to split the table. In second normal form, all nonkey columns depend on the whole key (not just part of it).

Figure 4.51

Third normal form. There is another problem with this definition. The customer name does not depend on the key (SalesID) at all. Instead, it depends on the CustomerID. Because the name and address do not change for each different SalesID, the customer data must be in a separate table. The Sales table now contains only the CustomerID, which is used to link to the Customers table and collect the rest of the data. The same rule applies to Salespeople.

SaleForm2(SaleID, SaleDate, CustomerID, Phone, Name, Address)

Sales(SaleID, SaleDate, CustomerID, SalespersonID)

SaleID	SaleDate	CID	SPID
117	3/3/2012	12345	887
125	4/4/2012	87535	663
157	4/9/2012	12345	554
169	5/6/2012	29587	255
178	5/1/2012	44453	663
188	5/8/2012	29587	554

Customers(CustomerID, Phone, Name, Address, City, State, ZIPCode, AccountBalance)

CID	Name	Phone	Street	City	Balance
12345	Jones	(312) 555-1234	125 Elm Street	Chicago	\$197.54
28764	Adamz	(602) 999-2539	938 Main Street	Phoenix	\$526.76
29587	Smitz	(206) 676-7763	523 Oak Street	Seattle	\$353.76
33352	Sanchez	(303) 444-1352	999 Pine Street	Denver	\$153.00
44453	Kolke	(303) 888-8876	909 West Avenue	Denver	\$863.39
87535	James	(305) 777-2235	374 Main Street	Miami	\$255.93

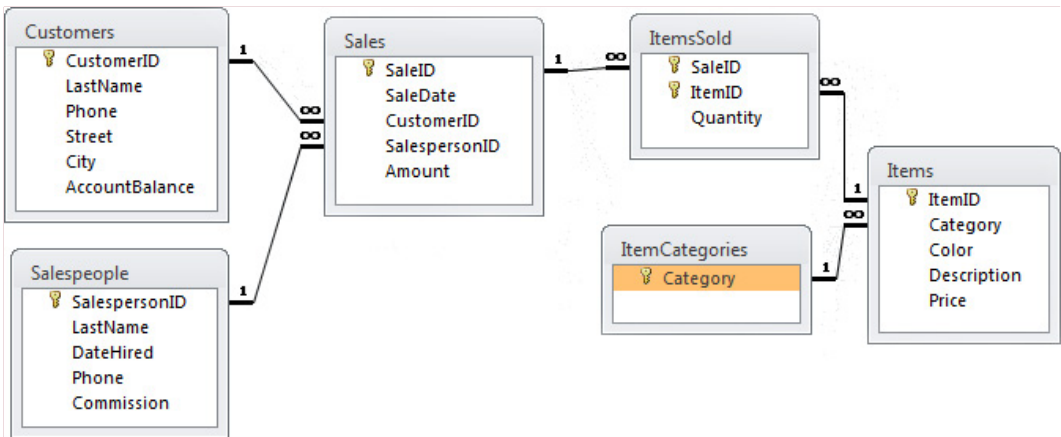


Figure 4.52

Third normal form tables. There are no repeating sections and each nonkey column depends on the whole key and nothing but the key. This figure also shows the relationships between the tables that will be enforced by the DBMS. When referential integrity is properly defined, the DBMS will ensure that rentals can be made only to customers who are defined in the Customers table.

a large database table and must be updated annually. For small cases, it is often easier to leave the three items in the Customer table.

Database Applications

How do you create business applications using a DBMS? Database systems typically have tools to help build applications. The tools make it relatively easy to create input forms and reports. Even if you never learn how to design a database, or become a programmer, you can learn to use these tools to build small applications and customized reports.

Data Input Forms

Rarely is data entered directly into the database's tables. Instead, input forms are used to enter some data automatically and to present a screen that is easier for users to understand. It is common to use colors and boxes to make the screen easier to read. Input screens can be used to perform calculations (such as taxes). Longer descriptions and help screens can be included to make it easier for the user to remember what goes in each column. A sample form from the Rolling Thunder Bicycles case is shown in Figure 4.53.

Many times, input screens look like older paper forms. Consider a typical order form, which first collects customer information such as name and address. It also contains lines for items ordered, descriptions, and prices. These are usually followed by subtotals and totals. If these forms exist on paper, it is easy to create them as a DBMS input screen. If you are creating a completely new form, it helps to draw it on paper first to get a feel for what you want it to look like.

Most input forms begin as a screen that is empty except for a menu line or some other help message. Three types of information can be placed on an input screen: (1) simple text, (2) input blanks, or (3) data retrieved from the database. A Windows-based DBMS can also include pictures, graphs, sound, and video.

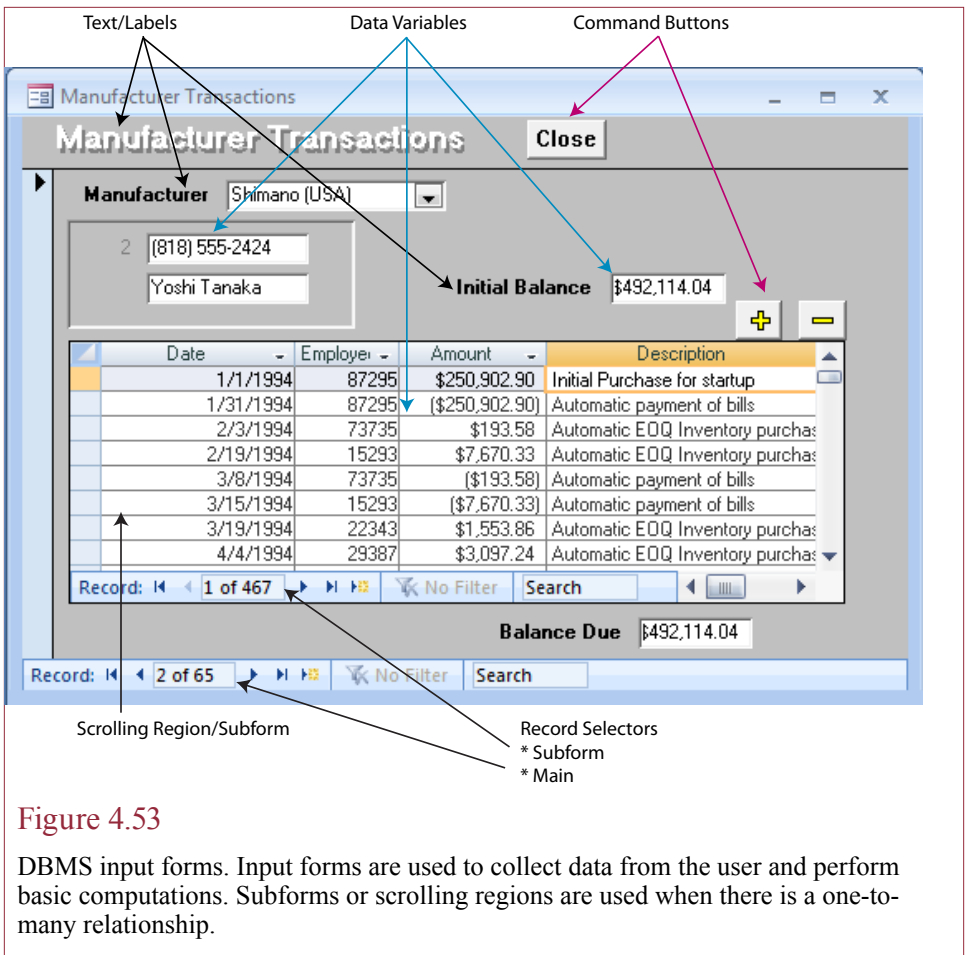


Figure 4.53

DBMS input forms. Input forms are used to collect data from the user and perform basic computations. Subforms or scrolling regions are used when there is a one-to-many relationship.

Paper forms have labels to tell the user what is supposed to be entered into each blank. For instance, many paper forms ask for a name: NAME _____. The label (NAME) tells you what you are supposed to enter on the blank line. A DBMS input form works much the same way. The first step is to type in the various labels. Move the cursor to a spot on the screen and type in a label or sentence that will tell the user what needs to be entered.

Most database systems automatically enter some types of data, such as the current date. If necessary, users can change the date, but it saves time by enabling them to press ENTER to accept the displayed value. The same situation holds for sequential items like order numbers, where the DBMS can automatically generate each unique order number.

After you have typed in the basic labels, the next step is to add the data-entry boxes. Just as you would type a blank line on a paper form, you need to tell the DBMS exactly what data will be entered by the user. For instance, move the screen cursor to a position next to the Date label, and then tell the DBMS to enter data at that point. You will specify the name of the column where the data will be stored. You can also specify default values. A **default value** is a value that is automatically displayed by the computer. For the case of the date, the DBMS will let you enter a name like Date() that will display the current date.

Reality Bytes: JPMorgan Chase Needs Better Database Design

In late 2010, JPMorgan Chase bank had a problem with its online site for three days. For at least 24 hours, customers could not conduct any transactions. The problem was eventually traced to a third-party authentication system that relied on an Oracle database. Apparently, some files in the Oracle database were corrupted and through mirroring the corruption was spread to the replicated copies. These copies then affected several other systems including Automated Clearing House (ACH) transactions and some loan applications. It also took developers several hours to clean up the problems because a large amount of extra data was being stored in the authentication database that should have been located elsewhere, including customer Web usage data.

Adapted from Jaikumar Vijayan, “Oracle Database Design Slowed Chase Online Banking Fix,” *Computerworld*, September 24, 2010.

When a DBMS prints out data, it can be formatted in different ways. You can control the way the data is displayed by using a format command. A date might be displayed as 10/24/2012 by entering the format MM/DD/YYYY. There are several common date formats; most firms tend to use one standard format. Note that many European firms use a format that is different from the common ones used in the United States. Most database software automatically picks up the regional settings from the Windows environment and displays dates in the local format.

The next section of the order form contains basic customer information. This data is stored in the Customer table, not the Orders table. When you select the Orders table, you might have to indicate that the Orders and Customer tables are connected to each other by the phone number. Now, place the text labels on the screen (customer name, address, etc.). Then place a data entry box after each label.

Next, you can add the Sales table; it is connected to the Orders table by the order number. Type in the column names for Item#, Description, Price, and Quantity. The DBMS input form will define this part of the table as a **scrolling region** or subform. To users, this subform will behave somewhat like a spreadsheet. They can see several rows at a time, and keys (or the mouse) will move the screen cursor up and down as users enter data into any row.

The only items entered in the Sales table are the Item# and the Quantity ordered. The Description and Price can be found by creating a look-up in the Items table. If the clerk using this screen types in the item number, the description and price will appear. With a good DBMS, it is possible to define a pop-up form or combo box in case the clerk does not know the number. This way, by pressing a certain key, a table listing each Item# and Description will be displayed in a window on the screen. The clerk can then scroll through the list to find the item.

Reports

Most of the time, the data listed in one table is not complete enough to help managers make decisions. For example, a listing of a Sales table might provide only phone numbers, item numbers, and the quantity ordered. A more useful report would print sales grouped by customer. It would also compute total sales for each customer. Because this report relies on data from several tables, it is best to base the report on a view.

Report Header					
Customer Sales					
Page Header					
Customer	Sale Date	Description	Price	Quantity	Extended
Customers.CustomerID Header					
CustomerID	LastName				
Detail					
	SaleDate	Description	Price	Quantity	Extended
Customers.CustomerID Footer					
					=Sum([Extended])
Page Footer					
					=Now() =Page: * & [Page] & " of " & [Pages]
Report Footer					
					=Sum([Extended])

Figure 4.54

DBMS report writers. Reports are created in sections. The report header is printed one time at the top of the report. Data in the page header section is printed at the top of every page. There are corresponding page footers and report footer. Primary data is printed in the detail section. Data can be organized as groups by creating breaks. Titles are often printed in the break header with subtotals in the break footer.

The view for the sales report example needs four tables. An OrderReport view is created that joins the Customer table to Orders by CustomerID, Orders to ItemsSold by OrderID, and ItemsSold to Items by ItemID. The DBMS will have a “create report” option to create the sales report. The report will be based on the OrderReport view. The report writer consists of a blank screen. You can put simple text statements anywhere on the page. You also can place data values on the page, and you can compute totals and make other calculations.

Most reports can be broken into categories. For example, there might be report titles that appear only at the front of the report (such as cover pages). Other information—such as the report title, date, and column labels—will be repeated at the top of each page. All of these items are called **page headers**. Similarly, there can be **page footers** at the bottom of each page. Reports may also contain **group breaks**. For instance, the sales report needs subtotals for each customer, so you need to break the report into subsections for each customer. Generally, you can specify several levels of breaks. For instance, you might break each customer order into totals by date. Each break can have a **break header**, a **detail section**, and a **break footer**. In the example, the customer name is printed on the break header. There is a detail line that lists the item information. The subtotals are displayed on the break footers. The report design or layout is illustrated in Figure 4.54. The report with sample data is printed in Figure 4.55.

To create this report, you first tell the DBMS that the report will contain one break based on customer phone number. You also define the variable Extended,

Customer Sales						
Customer	Sale Date	Description	Price	Quantity	Extended	
12345	Jones					
	3/3/2012	Blue Suede	\$50.00	4	\$200.00	
	3/3/2012	LCD-40 inch	\$1,000.00	1	\$1,000.00	
	3/3/2012	Red Boots	\$100.00	2	\$200.00	
	4/9/2012	Blue Suede	\$50.00	2	\$100.00	
	5/23/2012	Red Boots	\$100.00	1	\$100.00	
Sum						\$1,600.00
28764	Adams					
	6/27/2012	Blue Jeans	\$12.00	3	\$36.00	
	6/27/2012	LCD-40 inch	\$1,000.00	1	\$1,000.00	
Sum						\$1,036.00
29587	Smitz					
	5/8/2012	Blu-Ray Player	\$400.00	2	\$800.00	
	5/8/2012	LCD-40 inch	\$1,000.00	1	\$1,000.00	
	5/8/2012	LCD-40 inch	\$1,000.00	1	\$1,000.00	
	5/8/2012	Men's Work Boots	\$45.00	4	\$180.00	
Sum						\$2,980.00
44453	Kolke					
	5/1/2012	Blue Jeans	\$12.00	1	\$12.00	
	6/9/2012	Blue Jeans	\$12.00	1	\$12.00	
	6/9/2012	Blu-Ray Player	\$400.00	2	\$800.00	
	6/9/2012	LCD-40 inch	\$1,000.00	2	\$2,000.00	
	6/15/2012	Blue Suede	\$50.00	2	\$100.00	
Sum						\$2,924.00
87535	James					
	4/4/2012	Men's Work Boots	\$45.00	3	\$135.00	
	4/4/2012	Red Boots	\$100.00	4	\$400.00	
	6/9/2012	Blue Jeans	\$12.00	1	\$12.00	
	6/9/2012	Blue Suede	\$50.00	1	\$50.00	
Sum						\$597.00
Wednesday, July 06, 2011						Page 1 of 2

Figure 4.55

Sample report. Reports are often printed by groups or breaks with subtotals for each group. With a report writer, the layout, typefaces, and computations are easy to change.

which is price multiplied by quantity. Now you move the cursor to the top of the screen and type in the titles for the top of the page. Then place each column and variable on the report. You can format each item to make it look better. For example, you might want to format dates as MM/DD/YYYY so that all four digits of the year are displayed. Similarly, you can add dollar signs to the subtotals and totals.

When you have finished creating the report, you can print it. When you print this report, it should be sorted by customer name. The DBMS will also enable you to print the report so that it contains data just for one month. Notice that only five or six lines are needed to create a complex report. Without the DBMS report writer, it would take a programmer several hours to create this report, and it would be much harder to make changes to it in the future.

Putting It Together with Menus

If you are creating a database for yourself with just a couple of input screens and reports, you can probably quit at this point. On the other hand, for more complex databases or for projects other people will use, it would be wise to make the system easier to use.

Application generators are tools that enable you to combine the various features into a single application. The resulting application can be used by selecting choices from a menu, much like users do with commercial software. The important design feature is that you can create the entire application without writing any programming commands.

Consider a simple example. As a manager, you need a sales report printed every day that shows the best-selling items. Every week you want a list of total sales for each employee to bring to your sales meetings. You also send letters to your best customers every month offering them additional discounts. You want to put your secretary in charge of printing these reports, but you do not have time to explain all the details about how to use the database program. Instead, you create a simple menu that lists each report. The secretary chooses the desired report from the list. Some reports might ask questions, such as which week to use. The secretary enters the answers and the report is printed.

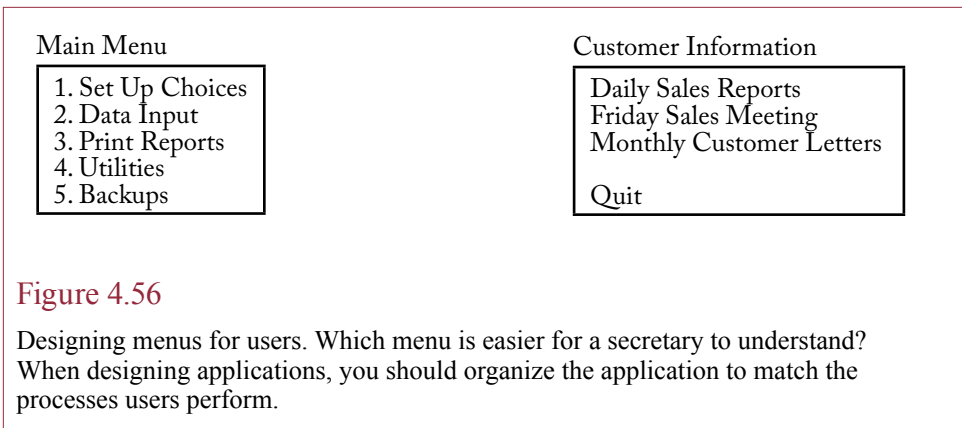


Figure 4.56

Designing menus for users. Which menu is easier for a secretary to understand? When designing applications, you should organize the application to match the processes users perform.

The first step in creating an application is to think about the people who will use it. How do they do their jobs? How do the database inputs and reports fit into their job? The goal is to devise a menu system that reflects the way they work. Two examples of a first menu are shown in Figure 4.56. Which menu is easier for a clerk to understand? The one that best relates to the job. Once you understand the basic tasks, write down a set of related menus. Some menu options will call up other menus. Some will print reports; others will activate the input screens you created.

Once you know how you want the menu structure to appear, you fill in the menu templates in the application generator. To create a menu, you type in a title and fill in the choices. Then you assign an action to each choice. Usually you just pick from a list of actions and type in specific data such as the name of the report and how you want it sorted. When you are finished, the application generator creates the application.

Database Administration

What tasks need to be performed to keep a database running?

Managing a database can be a complex job. Often hundreds of choices need to be made when the database is designed. Someone needs to be in charge of defining the data, making sure that all useful facts are captured, and managing security for this valuable asset. Databases have to be evaluated and fine-tuned on a regular basis. Someone has to keep track of these maintenance changes and decide when major updates should be installed. A **database administrator (DBA)** is usually appointed to manage the databases for the firm. The DBA needs to know the technical details of the DBMS and the computer system. The DBA also needs to understand the business operations of the firm.

The database administrator is responsible for all operations involving the database. These duties include coordinating users and designers, establishing standards, and defining the data characteristics. When new programs are created, the DBA makes sure they are tested and documented. The DBA also schedules backups and recovery, and establishes security controls.

In a few large companies, an additional person known as the data administrator (DA) is charged with overseeing all of the data definitions and data standards for the company. In this case, typically several DBAs are used to monitor and con-

trol various databases. The DA is responsible for making sure data can be shared throughout the company.

Standards and Documentation

In any company of moderate size, many different databases will be used by hundreds of workers. These databases were created at different points in time by teams of employees. If there are no standards, each piece will be unique, making it difficult to combine information from multiple databases or tables. The marketing department may refer to customers, but management calls them clients. The DBMS needs to know that both terms refer to the same set of data. Also, someone has to determine the key values for each table. Consider the Customer table. One department might assign identification numbers to each customer; another department might use customers' phone numbers, and a third department might use the customer names. To prevent confusion and to be able to combine information, it is best for all users to use only one of these methods to identify the customers.

Many standards are related to the database process. It is easier to use a database if all input screens have similar characteristics. For instance, the base screen might use a blue background with white characters. Data that is entered by the user will be displayed in yellow. Similarly, certain function keys may be predefined. ESC might be used to cancel or escape from choices. F1 might be used for help and F3 to display a list of choices. If each application uses keys differently, the user will have a hard time remembering which keys do what with which database.

Likewise, it is helpful to standardize certain aspects of reports. It might be necessary to choose specific typefaces and fonts. Titles could be in an 18-point Helvetica font, whereas the body of reports could be printed in 11-point Palatino. To provide emphasis, subtotals and totals could be printed in boldface, with single and double underlining, respectively.

One of the crucial steps in creating a database is the definition of the data. Many important decisions have to be made at this point. Besides the issues of what to call each item, the DBMS has to be told how to store every item. For instance, are phone numbers stored as 7 digits, or should they be stored as 10 digits, or perhaps stored with the 3-digit international calling code? Postal zip codes pose similar problems. The United States uses either a five-digit or nine-digit zip code, but is considering adding two more digits. Other countries include alphabetic characters in their codes. Someone has to determine how to store this information in the manner that is best for the company.

There are many other aspects of database design that need standards to make life easier for the users. However, whenever there are standards, there should be a mechanism to change these standards. Technology always changes, so standards that were established five years ago are probably not relevant today. The DBA constantly reviews and updates the standards, and makes sure that employees follow them.

Even though databases are easy to use, they would be confusing if the designers did not document their work. Picture a situation where you want to find information about customers but the designers named the table *Patrons*. You might never find the information without documentation.

Documentation can assume many forms. Most DBMSs allow the designers to add comments to each table and column. This internal documentation can often be searched by the users. Many times it can be printed in different formats so that it can be distributed to users in manuals. Because it is maintained in the database

along with the data, it is easy to find. It is also easy for the designers to add these comments as they create or change the database, so the documentation is more likely to be current. It is up to the DBA to ensure that all designers document their work.

Testing, Backup, and Recovery

One advantage of the DBMS approach is that it provides tools such as report writers and application generators that end users can employ to create their own systems. Although it is easier for users to create these programs than to start from scratch, the programs still need to be tested. Corporate databases are extremely valuable, but only if the information they contain is accurate. It is the responsibility of the DBA to keep the information accurate, which means that all software that changes data must be tested.

Most companies would not survive long if a disaster destroyed their databases. For this reason, all databases need to be backed up on a regular basis. How often this backup occurs depends on the importance and value of the data. It is possible to back up data continuously. With two identical computer systems, a change made to one can be automatically written to the other. If a fire destroys one system, the other one can be used to continue with no loss of information. Obviously, it is expensive to maintain duplicate facilities. Many organizations choose to back up their data less frequently.

The main point of backup and recovery is that someone has to be placed in charge. Especially in small businesses, there is a tendency to assume that someone else is responsible for making backups. Also, remember that at least one current copy of the database must be stored in a different location. A major disaster could easily wipe out everything stored in the same building. There are some private companies that for a fee will hold your backup data in a secure, fireproof building where you can access your data any time of the day.

Access Controls

Another important task in database administration is the establishment of security safeguards. The DBA has to determine which data needs to be protected. Once basic security conditions are established, the DBA is responsible for monitoring database activity. The DBA tracks security violation attempts and monitors who is using the database. Because there are always changes in employees, removing access for departed employees and entering new access levels and passwords can be a full-time job.

Databases and e-Business

Why are databases so important in e-business? Many people still think of Web sites as simple pages of text with a few images. But e-business requires interaction with customers and the company data. Consequently, most e-business Web sites are connected to databases. In e-commerce, customers want to know if a product is in stock—this information is in the database. Similarly, customer, order, and shipping data have to be maintained and shared throughout the company. Other e-business sites use databases to provide services, store transaction data, and provide search and matching capabilities.

Designing an e-business database is no different than traditional business applications. However, the technologies for building Web-based applications are still evolving. Currently, two leading systems are being developed: Oracle/Sun

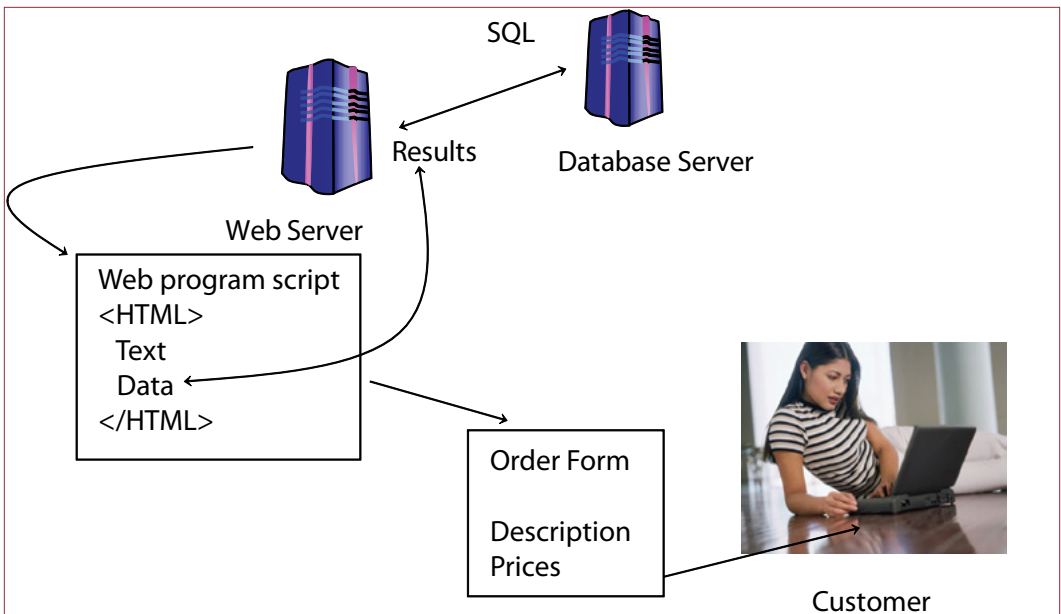


Figure 4.57

E-business database. When a customer requests a page, the server runs a script program that interacts with the database by sending queries and formatting the data to build a new Web page.

is championing a Java-based approach known as J2EE, and Microsoft is building the .NET platform. Databases are the heart of both systems, and both are server-based technologies designed to build interactive Web sites. Unfortunately, the two systems are completely independent and incompatible. If you build an application to run with one approach, you would have to completely rewrite it to use the other method.

Because the two approaches are so different, if you are building an e-business system, one of your first actions is to select one of these systems. In most cases, the J2EE approach runs on UNIX-based computers, but some versions exist for Microsoft-based systems. The Microsoft approach will probably only run on Microsoft servers, but some companies are experimenting with building versions that run on UNIX systems. Because the two systems are both new, it is difficult to evaluate them on technical grounds. The Microsoft approach offers a little more flexibility with its support for multiple languages. Microsoft .NET also provides a complete development environment with several easy-to-use tools that make it relatively easy for beginners to create database-oriented Web sites.

The basic approach is shown in Figure 4.57. Web developers create script pages that interact with the database. When customers request a page, the server executes the associated program script. The script sends queries to the database and retrieves the desired data. For example, the script might retrieve product descriptions, prices, and in-stock status. The data and images are added to the Web page and sent to the customer, who sees only the simple results.

Reality Bytes: Salesforce and Database.com

Salesforce.com is often listed as a leader in cloud computing—moving customer information to the Web. In late 2010, the company released Database.com, a cloud-based system largely based on Oracle's tools to support applications running on any platform. The system uses application programming interfaces (APIs) to deliver data to other systems. Essentially, the site becomes a giant database store to hold any information programmers need for Web-based services. The basic free version supports three users, 100,000 records and 50,000 transactions per month. Each set of 100,000 rows costs \$10 per month, and each set of 150,000 transactions carries a charge of \$10. The system does not really use a relational data model, which reduces its flexibility, but simplifies data storage and retrieval for some types of problems.

Adapted from Chris Kanaracus, "Salesforce.com Unveils Database.com," *Computerworld*, December 7, 2010.

Generally, the database runs on a separate server—which reduces the load on the Web server and makes it easier to handle backups and other database maintenance chores. However, with increasingly powerful servers, one server can handle smaller applications.

Cloud Computing

How are databases used in cloud computing? Several of the pioneers in cloud computing created their own special-purpose database systems. These are not relational database systems. For example, Google needs a way to store and retrieve huge amounts of data quickly. The company built huge sets of parallel servers, and then created a database system called BigTable that is designed to work over this distributed network. The system works well for storing documents and has built-in support for versioning. But it works nothing like SQL. For instance, there are no JOIN commands. Instead, you specify key values for the row, column, and time and store a single object at that location. BigTable can be accessed through the Google App Engine application. It is useful for storing complex document-type data. A more generic version of the database known as Hadoop is available from several open-source providers, and Facebook has created (and distributed) the Cassandra database system that works in a similar manner.

Amazon is another early Web company that provides cloud computing services. It currently offers three ways to store data: S3, SimpleDB, and Relational Database Service (RDS). S3 is primarily designed to store and deliver Web pages and other common Web objects. It is easiest to think of it as a file service. You upload files to the S3 servers, and build links to them on your Web pages. When users view your pages, the content linked to the S3 service is delivered from the Amazon network. Because Amazon has distributed servers and high-speed Internet connections, this content is delivered relatively quickly even under heavy loads. It is best used for storing large media files (music, video, and so on) that will be needed by many people. Pricing is based on the amount of data stored and transferred each month with no fixed costs. Amazon SimpleDB is similar to Google's BigTable. For starters, you need a programmer to use either one. You basically store and retrieve data by specifying an item name, an attribute name and the value. Differences exist between the Google and Amazon systems, but the

Amazon RDS (MySQL), U.S. East
1 Extra large instance
20 hours/day
20 GB/month at 50 million I/O per month
10 GB/month data transfer in
500 GB/month data transfer out
20 GB/month regional transfer
Estimated cost: \$616 per month or \$7400/year.

Figure 4.58

Amazon RDS (MySQL) estimated cost. Calculation is from Amazon's Web site calculator. Values are estimated and might not include all costs. But the example provides a mid-size database with T1-level data transfer for less than 10 percent of the cost of a single DBA.

underlying concepts are similar. More recently, Amazon introduced a cloud-based relational database service. It is largely based on the MySQL DBMS and uses the same syntax. Amazon has also indicated that it will implement an Oracle 11g database as well.

Microsoft sells the online service Azure which is a complete implementation of SQL Server as cloud computing. The system is built on multiple servers run by Microsoft administrators and your database can scale up to relatively large sizes. It is relatively easy to convert applications built using standard Microsoft development tools to run on the cloud servers.

The main benefit to using cloud computing for databases is that you do not have to run the hardware or software. That means you do not need to hire people to maintain and update everything, and you do not have the initial fixed costs. You are free to concentrate on the data and the applications. With most cloud database services, pricing is based on usage rates. If your company is starting up and you have small amounts of data and few users, monthly prices will be low. As the company grows and data increases, you will pay higher fees. But, you should be generating more revenue to cover those fees. And, most services include economies of scale so the rates decrease when you hit certain levels.

Cloud based databases are ideal for building Web-based applications. The reliability and scalability provided by the big service providers ensures that your applications will be available even as demand increases. Yet, the flexible pricing keeps the prices low during the start-up or low-use periods. Figure 4.58 shows an estimate of costs using Amazon RDS for a mid-size database. The transfer values are estimated to be about equal to a full T1 utilization rate. The point is that the annual cost is less than 10 percent of the cost of hiring a single IT worker to run the system—and one worker would not be able to provide 24-hour support. Another approach is to look at the cost of the T1 line—probably \$300-\$400 a month or half of the lease cost alone. Server and network hardware costs (3-year life) would easily make up the other half of the costs. So even if you ignore personnel costs, it is unlikely that a similar system could be configured in-house for this price.

On the other hand, the costs continue to increase even as your company becomes large. At some point, it becomes cheaper to pay the fixed costs of buying and running your own hardware, software, and networks. It is up to you as a manager to monitor the operating costs and compare them to the costs of running your

own facilities. The cloud computing providers have expertise and specialize in providing their support, so it will be expensive to acquire that same level of expertise. Also, as the number of cloud providers increases, the competition should help to hold down prices. Documentation for the various services along with pricing models are available online, so it is straightforward to estimate the costs based on the amount of data and the monthly data transfer rates.

Summary

Everyone needs to search for information. Computers make this job easier, but someone must set up and maintain the databases to ensure their integrity. Relational database systems are increasingly used as the foundation of the information system. They make it easy to share data among users while maintaining access controls. Equally important, the databases are easy to alter as the organization changes. Sophisticated databases can handle all the data types in use today, not just simple numbers and text.

It is relatively easy for users to obtain data using SQL or query-by-example tools. Because SQL is a recognized standard query language, it is worth remembering the basic elements of the SELECT command. The syntax is easy (SELECT columns, FROM tables, WHERE conditions, ORDER BY columns). Just remember that whenever you use more than one table, they must be joined by related columns.

An important step in databases is to design them correctly. The trick is to split the data into tables that refer to exactly one concept. Most organizations have a database administrator to help users create the initial database tables, define standards, establish access rights, and perform backups and testing. Once the tables have been defined, users can create input screens, reports, and views by using graphical tools to draw the desired items on the screen.

It is important to choose the right tool for each job. Databases excel at handling huge amounts of data and sharing it with other users. On the other hand, spreadsheets are designed to perform calculations and create graphs. One indication that a problem should be solved using a DBMS instead of a spreadsheet is that several tables of data are involved.

Several database systems, using different technologies, are available as cloud computing services. These services provide scalability in the amount of data stored and transferred across the Web, with almost no fixed costs. But they can be expensive for some applications so managers must compare the in-house fixed costs to the total variable costs of using a cloud service.

A Manager's View

Every business has to store data. Every manager needs to do research. Sometimes you will have to summarize and evaluate transaction data. Sometimes you will use external databases to evaluate the industry and your competitors. Database management systems provide important capabilities to managers. One of the most useful is a query language, such as QBE or SQL, that enables you to answer questions without the need for hiring an MIS expert. A DBMS also speeds the development of new systems and provides basic features such as report writers and input forms.

Key Words

application generator	detail section
BETWEEN	LIKE
break footer	normalization
break header	one-to-many
break	page footer
column	page header
composite key	primary key
concurrency	QBE
data independence	query by example
data integrity	row
database administrator (DBA)	scrolling region
database management system (DBMS)	SQL
default value	table
	view

Web Site References

	Database References
Computerworld data Management	www.computerworld.com/s/topic/173/Databases
eWeek Database	www.eweek.com/c/s/Database/
	Car Shopping Services
America's Automall	www.automall.com
Autobytel	www.autobytel.com
AutoNation	www.autonation.com
AutoTrader	www.autotrader.com
Carmax	www.carmax.com
Dealer Net	www.dealernet.com
Edmunds	www.edmunds.com
Kelley Blue Book	www.kbb.com
Vehix	www.vehix.com

Review Questions




1. How is data stored in a relational database?
2. What are the main benefits of using databases to build business applications?
3. What four questions do you need to answer in order to create a database query?
4. How do you join tables with QBE? With SQL?
5. How do you enter restrictions or constraints with QBE? With SQL?
6. How do you perform computations with QBE? With SQL?
7. Would you prefer to use QBE or SQL to access a database? Why?
8. What tasks are performed by a database administrator?
9. Why are forms (input screens) important in a relational database?

10. Do you think business users can create their own reports with a DBMS report writer? What limitations does it have?
11. Why are standards important in a database environment?
12. Why is a DBMS important for e-business?
13. How are non-SQL databases different from relational databases?

Exercises

C04Ex15.mdb

It is best to answer the first 15 exercise questions using a DBMS, but if one is not available, you can use the tables in the text and write your queries by hand. If you have a DBMS that handles both QBE and SQL, you should do the exercise with both methods. The number at the end of the question indicates the degree of difficulty.

-  1. List the customers who live in Chicago. (1)
2. List the items where the color is Blue. (1)
3. List the salespeople with a commission rate greater than 4%. (1)
4. List the customers with a last name that begins with the letter 'S'. (1)
-  5. How many customers have an account balance of more than \$400? (2)
6. Which customers have purchased a Blu-Ray player? (3)
7. List all of the customers of salesperson West. (3)
8. List all of the items purchased by Customer Kolke. (3)
9. Which salesperson made the most sales by count? (3)
10. What was the total value of sales in May? (4)
11. Compute the commission owed to each salesperson in June based on the Sales Amount. (4)
12. What is the total amount of sales value in June? (4)
-  13. Which item category had the highest sales value in May? (4)
14. Write queries to test if the Amount value in the Sales table is correct for all sales. (5) Hint: Save a separate query to perform the computations.
15. Which items had no sales in June? (5) Hint: This query requires a LEFT JOIN or a subquery, which are not covered in this book.
16. You have been hired by a small county government office to track properties for tax purposes. The annual property tax rate is set by the local governments, but the assessor's office is responsible for tracking the value of each property. Whenever a house is sold, the sale price is reported to the office. Since this price is a concrete measure of the price, it has to be recorded permanently. In other cases, the assessor estimates the value of a property from its primary features, such as the lot size (measured in fractions of an acre), house size

(square feet), age (year built), neighborhood, and general condition. New values are estimated every year if a house has not been sold. If a property has not sold for more than five years, an assessor performs a simple inspection of the property and takes pictures from the road. Property owners can appeal an assessment to a board. The appeal date and any comments are recorded. The data is currently recorded on a form organized by each property. Recently, the office has been assigning GPS coordinates (latitude, longitude, altitude) to each property. Many do not have data yet but will be added when the properties are inspected. Create a list of normalized tables need to build this database.

Property ID			
Year Built	Construction Price	Latitude	
Year Remodel		Longitude	
Address		Altitude	
Tax Code Area			
Zone Code	Township/Range/		
Subdivision			
Section			
Lot size		Bedrooms	
House size		Bathrooms	
Sewer		Fireplaces	
Water		Garage	
Utilities		Basement	
Property Description			
Year	Valuation	Method	Comments
Appeals			
Date	Claimed Value	Resulting Value	Comments

17. A friend has asked for your help in training for a triathlon. No, you do not have to run with her at 5 AM. She wants a personal log to track her progress and well-being. She has been writing entries in a journal, but she wants to be able to plot the data and compute totals. The basic layout of her existing log is shown in the form. For each day, she tracks the distance and time of each

run, bike ride, and swim. In all three cases, distance is recorded in miles; but often includes fractions. Eventually, when she plots the data, she wants the ability to exclude some days. For instance, one day her bike broke and she had to run home, so the time would not match with the others. She usually records her basic meals and estimates the total calories for the day. For the health category, she estimates the number of hours of sleep she gets at night, but the other entries are subjective comments and notes about the day.

Date	Run	Bike	Swim	Food	Health
Sunday	Dist. Time Comment	Dist. Time Comment	Dist. Time Comment	Breakfast Lunch Dinner Snacks Calories	Sleep Illness Comments

18. Using your experience in a job, school, or any organization; identify a problem that could be supported with a database management system—that requires at least three separate tables. Sketch a sample form.
19. Think about business or Web-based applications and identify a task that you believe could best be supported by a non-SQL DBMS. Explain why.



Technology Toolbox

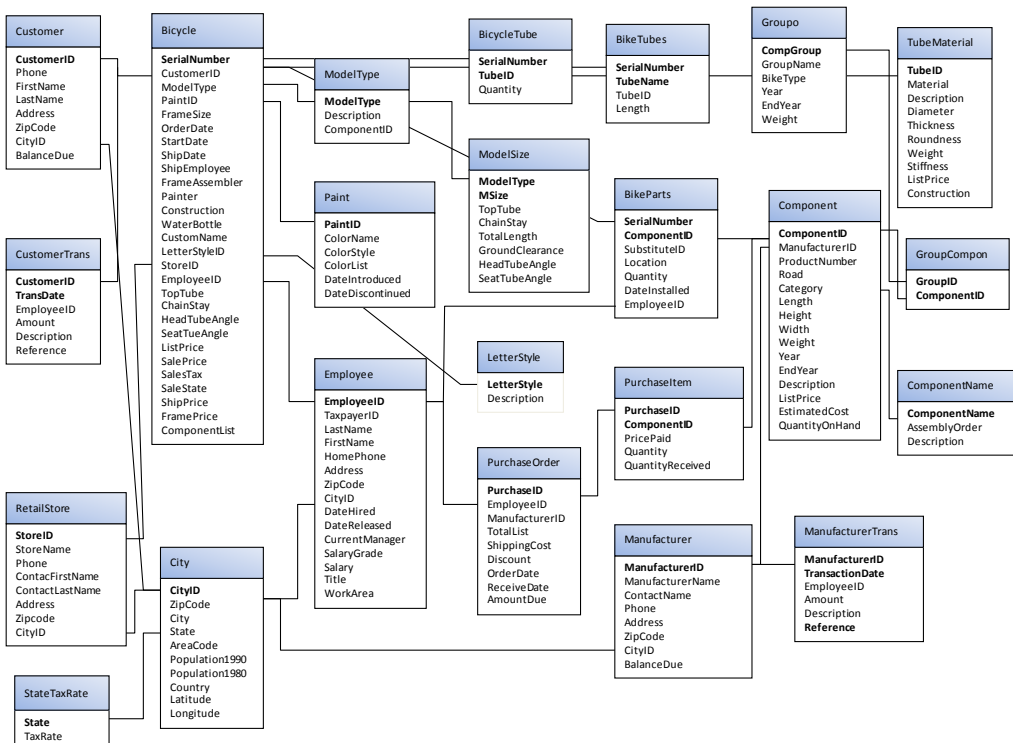
20. For the Sales database, create input screens for Items, Customers, and Salespeople. Add an ItemCategory table that holds a list of the categories so you can add a combo box to the Items form to select from the list of categories.
21. Using a different design, rebuild the Sales form.
22. Add a button to the Sales form that enables users to open the Customer form to edit the data or enter a new customer.
23. Create an inventory report that lists all of the products; group them by category; and, within each category, sort them by price.
24. Create the customer sales report that is described in the chapter.
25. Create a start-up form that can be used as a menu. Begin in Design View and add buttons that open the other forms and reports. Use colors or images to enhance the appearance of the form.



Teamwork

26. As individuals, each person should investigate a cloud-based DBMS in terms of capabilities and pricing. Combine the results as a team and select one tool that you would choose for a start-up Web-based company that anticipates starting with 1,000 customers a month and increasing to 10,000 customers a month within two years.

27. Assign the tables in the Sales database to each person on the team. As a group, identify the primary users of the system. Each person should then specify the security access rights for each user on each table: SELECT, INSERT, UPDATE, or DELETE.
28. Assume that you need to buy a DBMS to create a Web-based company. Research the components needed and have each person find information and evaluate a DBMS package. Try to identify costs as well as strengths and weaknesses of the package. Share the individual results and create a report that makes a recommendation.
29. With the cooperation of a local small business, create a database for that company. Note that you should verify the initial layout of the tables with your instructor or someone who has studied database design. Assign specific forms and reports to individual team members and combine the pieces.
30. Each team member should write up three business questions related to either the C04Ex15.mdb or Rolling Thunder database. Exchange the questions with the other team members, and then create the queries to answer each question. Share your answers.





Rolling Thunder Database

Create queries to answer the following questions. (*Difficulty level: 1 = easiest.*)



31. List all race bikes that sold in 2009 with a sale price greater than \$5000. (1)

32. List all the purchase orders in 2008 with a total value greater than 200,000. (1)

33. List the employees hired after January 1, 1995 with a salary grade higher than 10. (1)

34. List the road bikes purchased in December 2009 with a FrameSize greater than 50 cm. (1)

35. List the paint colors introduced after 1995. (1)



36. List the retail stores in Delaware. (2)

37. List the customers (name and phone) who bought full suspension mountain bikes in December 2008. (2)

38. List all the race bikes ordered in 2010 that were painted with red in the color list sold to the state of New York. (2)



39. List the manufacturers who supplied mountain bike (MTB) forks in 2010. (2)

40. List the employees who painted race bicycles in January 2010. Hint: change the relationship from EmployeeID to Painter. (2)

41. Compute the number of bicycles sold by state for 2009. (3)

42. Which letter style was the most popular for Road bikes in 2010? (3)

43. What was the average discount (ListPrice – SalePrice) given on mountain bicycles in 2009? (3)

44. Which manufacturer did RT buy the most components from by value in 2010? (3)

45. Which employee sold the most race bikes in March 2010 by count? (3)

46. By count, what was the most popular crank installed on full suspension mountain bikes in 2010? (4)

47. How many bikes were sold each month in 2010. Hint use the function Format(OrderDate, “yyyy-mm”). (4)

48. For 2009, was the average price of mountain bikes lower than the average price of full suspension bikes? (4)



49. What was the average effective sales tax rate (tax/SalePrice) by state in 2010? (4)

50. In 2009, did race bikes have more Cranks made by Campy or Shimano? (4)

51. Which customers who have purchased mountain bikes have also bought Road bikes at any time? (5) Hint: Create two separate queries and save them.

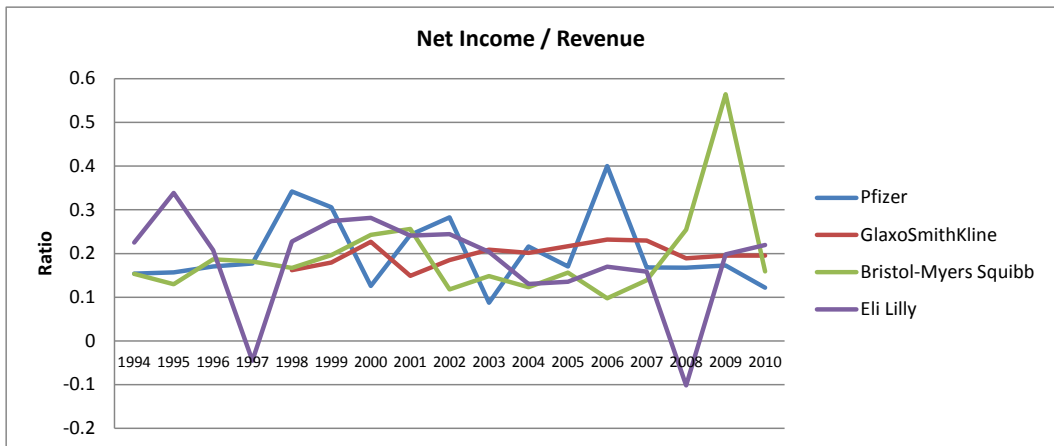
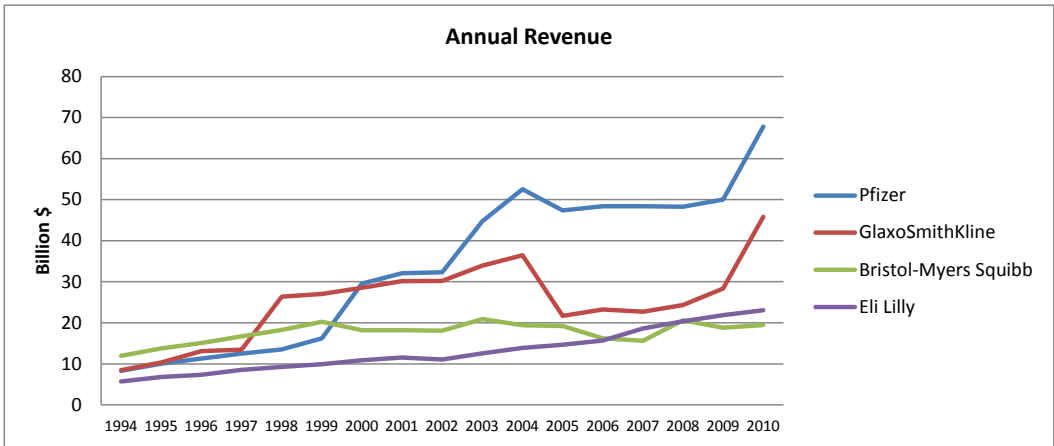
52. Which customers purchased two or more bikes in at least two different years? (5)

Additional Reading

- Disabatino, Jennifer, "NHL Scores With Database On Draft Day," *Computerworld*, July 9, 2001. [NHL uses a database to provide more information and cut hours off the time to draft players.]
- Fox, Prim, "Tax Filing Gets Connected to the Web," *Computerworld*, August 8, 2001. [The importance of connecting databases to the Web even for governments.]
- Mayer, Merry, "New DNA Database Extends The Long Arm Of Law Enforcement," *Government Computer News*, October 19, 1998, p. 47. [Government database to identify criminals.]
- Post, Gerald, *Database Management Systems: Designing and Building Applications*, Irwin/McGraw-Hill, 2002. [How to design databases and use them to build business applications.]
- Sliwa, Carol, Lee Copeland, and Don Tennant, "Dead voters in Florida?" *Computerworld*, November 13, 2000. [Only 10 states have an online central voter registration database, and 16 have no state database at all.]
- Trombly, Maria, "Schwab Database Glitch Cuts Users Off From Some Information," *Computerworld*, February 13, 2001. [The importance of database availability and reliability.]
- Tiboni, Frank, "FEMA Automates Property Inspection Scheduling," *Government Computer News*, November 9, 1998, p. 14. [Emergency agency uses telephone registration and database to help disaster victims in less time.]

Cases: Pharmaceuticals

The Industry



The pharmaceutical industry has received considerable attention in the past few years. Several organizations, including the AARP, have criticized the industry for price increases on name-brand drugs that exceed the inflation rate. The AARP reported that prices have increased by 6 to 7 percent a year in 2002 and 2003 (Kaufman and Brubaker 2004). Prescription drugs often form a significant percentage of household budgets for elderly retirees. Yet the pharmacy industry notes that identifying and testing a new drug is expensive. On average, it takes 10 to 15 years and costs \$800 million to bring one new drug to the market. And only one in five potential drugs makes it through the entire process. The industry spent an estimated \$33.2 billion on R&D in 2003 (PhRMA 2004). On the flip side, the pharmaceutical companies spend about as much on marketing as they do on R&D (financial statements). In the same report, the industry observes that 10.5 percent of total health care dollars in the United States are spent on prescription drugs.

Drug Prices

The combination of high research costs, high risk, and the need to make a profit makes it difficult to set public policy. Profits encourage additional research and development to create new drugs. High prices keep the drugs out of the reach of some people who need them. Some governments have imposed price controls in an attempt to keep prices lower for patients. Canada presents an interesting example. Because trade with Canada is relatively unrestricted, U.S. patients have turned to Canadian pharmacies to fill their prescriptions. Even some state and local governments have tried to save costs by purchasing drugs from Canada, where the prices are held down by government rules (Schiavone 2003). Drug companies have staged an all-out battle to prevent the purchase of drugs from other countries. GlaxoSmithKline went as far as stopping sales to Canadian pharmacies (CNN 2003).

In 2004, the U.S. government introduced a Medicare insurance policy for purchasing name-brand prescription drugs. Participants have to sign up for the plan and pay a specified amount. The policies are sold by private companies, and the terms vary. Essentially, the policy covers a certain level of drug costs. However, with the copayments, limits on total value, and rising prices, it is not clear how many people will be able to benefit from the plan.

Research and Development

Research and development is a difficult process. Many times, luck and accidents play a role in creating new drugs. Yet companies need a huge infrastructure to take advantage of that luck. Research requires people and labs. Increasingly, it also needs information technology—in the form of databases, analytical tools, and collaboration software.

Finding a potential drug is only the first step (research). Development requires extensive testing, finding ways to deliver the drug with minimal side effects, and manufacturing it cheaply and consistently to narrow tolerances.

A pharmaceutical firm has to recover all costs within the life of a patented drug. Once the patent expires, the drug can be manufactured generically, and the inventing firm no longer makes any money. Since 2000, several of the leading firms have suffered substantial declines in revenue as their patents expired on several blockbuster drugs.

An interesting twist on pharmacy research arose in 2004. A couple of firms, particularly GlaxoSmithKline, were charged with withholding research data. All potential drugs have to go through clinical human trials to identify the specific gains and evaluate potential side effects. The drug companies provide the results of the studies to the Federal Drug Administration (FDA), which has to give final approval to the drug. In at least some cases, negative research was withheld. Many of the trials are somewhat secretive—neither patients nor physicians can be allowed to know which patients have the specific drugs. Consequently, it is relatively easy to bury negative results and only publish the positive ones. Of course, easy is not the same thing as ethical. As a result of substantial publicity in 2004, several agencies, companies, and people are pushing for a national registry of clinical trials—before they are started. This way, everyone can check on the progress—both good and bad (Martinez 2004).

Genetics is playing an increasing role in drug discovery and testing. Several specialist firms (e.g., Genentech) have been struggling for several years to use genetic modifications techniques to create completely new drugs. A few of the drugs have been successful, but the sub-sector has not yet matched its early promises.

Marketing

Pharmaceutical firms have substantially pushed the boundaries of drug marketing in the last decade. Originally, the firms relied on publications in medical journals to highlight the success of their drugs. Of course, sales representatives made direct calls to physicians to point out the results, offer comparisons with existing drugs, and answer basic questions. These activities are still important to the industry. However, the industry changed enormously when it began marketing drugs directly to consumers. The FDA has relatively strict rules about the contents of these ads, and it routinely monitors the ads and requires companies to alter the ads. Nonetheless, physicians are now bombarded with questions from patients about a new drug they saw on television. Of course, patients rarely read the fine print—telling about the side effects—and few can make comparisons among the various drugs. So, physicians have to spend time explaining their choices. The pharmaceutical industry points out that increased communication between doctor and patient is a good thing. But, there remain many unresolved questions about the effectiveness of the campaigns—other than to boost pharmacy company profits.

Increasingly, pharmaceutical firms are being asked to track drugs and shipments more closely. California and Florida both passed laws that will require the industry to track the entire supply chain of drugs from the manufacturer to the wholesaler and retailer. California's law takes effect in January 2009. The security company VeriSign has established a database that can be used to track ownership of drugs, and sales will not be allowed until the seller can verify its ownership of the drug. AmerisourceBergen, a drug wholesaler, is working with the pharmaceutical companies to develop the system. In the early stages, the companies are considering the use of RFID tags instead of simple bar codes to make it easier to read the data (Weier 2007). Many Schedule 1 drugs, such as the addictive painkiller OxyContin, are already tracked with RFID tags. Pfizer also announced that to reduce thefts and “diversions” to drug dealers, it would begin tracking bottles of Viagra (Carr and Barrett 2005).

By 2011, pharmaceutical firms began changing their approach to marketing. The firms were facing declining profits, loss of patents, and the fact that physicians do not like being interrupted. By 2010, about 25 percent of physicians put themselves on “do-not-see” lists and about 75 percent of sales rep visits did not result in face-to-face meetings. So tens of thousands of sales reps were eliminated, and largely replaced with Web sites, iPad apps, and other digital tools. Older-style Web sites were discontinued in favor of interactive sites that enable physicians to ask questions, or get information from online reps (Whalen 2011).

Information Technology Needs

The pharmaceutical industry runs on data, information, and knowledge. The research departments generate and need to scan enormous amounts of data. They experiment and produce papers that disseminate knowledge. Researchers need to collaborate and share this data and knowledge. But the size of the companies, the variety of products, and the geographical dispersion all add complications. Additional data comes from physicians and even customers. On top of the research data, the marketers need to track salespeople, physician contacts, and marketing plans. The legal staff needs to monitor progress, evaluate patent documents, and shepherd thousands of documents through the FDA approval process. On top of all that, the CFO and CEO have to run the business—watching the financial data and creating the standard business reports.

Additional Reading

- Carr, David F. and Larry Barrett, "Philip Morris International: Smoke Screen," *Baseline Magazine*, February 1, 2005.
- CNN Online*, "GlaxoSmithKline Stops Selling Drugs To Canadian Pharmacies," January 21, 2003.
- Kaufman, Marc and Bill Brubaker, "Study: Drug Prices Outpace Inflation," *The Washington Post*, May 25, 2004.
- Martinez, Barbara, "Spitzer Charges Glaxo Concealed Paxil Data," *The Wall Street Journal*, June 3, 2004.
- Pharmaceutical Research and Manufacturers of America (PhRMA), *Pharmaceutical Industry Profile 2004*, Washington, DC: PhRMA (<http://www.phrma.org/publications/publications//2004-03-31.937.pdf>).
- Schiavone, Louise, "Governments Eye Canada For Cheap Drugs," *CNN Online*, October 16, 2003.
- Simons, John, "When Big Pharma Gets Too Big," *Fortune*, December 4, 2006.
- Weier, Mary Hayes, "Pharma Industry Debating Bar Codes Vs. RFID For Drug Tracking," *Information Week*, March 28, 2007.
- Whalen, Jeanne, "Drug Makers Replace Reps with Digital Tools," *The Wall Street Journal*, May 9, 2011.

Case: Eli Lilly and Company

Eli Lilly (ticker: LLY) is one of the leading pharmaceutical companies. Founded in 1876, the firm now has more than 46,000 employees (www.lilly.com). Like other pharmaceutical firms, Lilly is dependent on revenue from blockbuster drugs. When patent protection expires for a blockbuster drug, as it did for Lilly's Prozac in the early 2000s, revenue and profits drop. The challenge for companies is to fill the pipeline with new drugs that will provide a new revenue stream. Research and development are key elements in this process. But finding new blockbuster drugs that solve a major medical problem is not easy. Lilly, like the other firms, is turning to biotechnology to help create and test new drugs. Lilly's genetically engineered insulin has been one of the first successes in this new line of research. Lilly is also hoping to capitalize on the erectile dysfunction sales by heavily advertising its Cialis drug directly to consumers.

Research and Development

R&D is critical to generating profits. But R&D has become considerably more complicated. Research is no longer performed by one or two scientists working alone in a lab. Instead, Lilly has thousands of researchers working on various projects. Each of the compounds being created needs to be shared across the company. Collaboration has become a critical success factor. Throw in the fact that most of the large companies have expanded through mergers. Now, how do you combine the systems and the people to provide effective collaboration? Charles Cooney, codirector of the Program on the Pharmaceutical Industry at MIT, observes that "R&D is useless unless it can be integrated into a bigger picture that helps you

convert leads to drug candidates. The difficulty for the CIO is getting the team to work between the various disciplines needed to integrate the information in order to create knowledge” (Overby 2002).

Genetics is an increasingly important factor in pharmacy, and it adds many complications. Lilly, as well as the other firms, is looking at a future where drug selection may be tailored to a patient’s DNA. Some drugs are more effective when specific genes are present. This level of knowledge will require a tremendous amount of information flow—both in R&D and in marketing.

In 2006, Lilly took an interesting step by outsourcing some data management and data analysis to Tata Consultancy Services (TCS) in India. TCS is well known for providing off-shore consulting for programming and other IT services. But this contract calls for scientific analysis using a facility dedicated to working on data analysis for Lilly. Steve Ruberg, Lilly’s group director for global medical information sciences said that the agreement includes “gaining access to a global talent pool, increasing flexibility and scalability of our resources, and maintaining a global workflow that is operational 24 hours a day” (McDougall 2006).

Information Technology

Information technology with its connections to customers provides a new way to evaluate drug effects. All drugs go through scientific trials where physicians carefully monitor patients and evaluate side effects. But testing on a few hundred or even a thousand subjects does not always provide complete information. Historically, drug firms have relied on physicians to evaluate and report on effectiveness and side effects. Lilly’s drug Strattera was the first nonstimulant treatment for children with attention-deficit hyperactivity disorder. On the basis of early trials, the company knew that sleepiness could be a side effect. To evaluate the drug when it was released to customers, Lilly designed a customer relationship management system to track customer contacts through its call center. With the feedback from actual customer use, the company was able to identify that the sleepiness was minimized if the drug was taken at a meal and at night. Sidney Taurel, Lilly’s CEO, notes that “we’ve seen that IT makes you more effective. It’s a big shift. We’ve changed to seeing IT as an enabler of effectiveness” (Murphy 2004).

“Big shift” is right. When Lilly hired Tom Trainer as its first CIO in 1994, he found a mess—with 17 different IT organizations and minimal use of computers for business. By the time Trainer left in 1999 to head global IT for Citigroup, the technology had improved considerably. But IT was still largely used to reduce costs. CFO Charlie Golden says, “We had gone through quite a change in our perspective and strategy in IT. It was important to try to sort through what it would take to continue to change, but also to make IT a more integral part of our strategy” (Ewalt 2003). Roy Dunbar was chosen as the new CIO. He began as a pharmacist, had some business experience, but little experience with information technology. With some intense education, and a lot of humility, Dunbar learned the foundations of information systems. His ultimate charge was to find ways to use technology to support the strategy of the firm. For that, he needed everyone to start with a business perspective. One of the key tools his team created in 2000 was the Molecule Library. It is a knowledge management system that makes it easy for researchers to find information about chemical compounds being worked with in the company’s pipeline. The system reduces research time from days to minutes. Another new tool, the Sample Identification Database, functions as a registry of all compounds being developed. The Gene Anatomy Made Easy (Game) project

helps scientists identify DNA analysis without requiring expert help. Shaving a few hours a week off the research time of every scientist can help get projects to market earlier, saving billions of dollars in costs (Ewalt 2003).

In 2001, the IT department began expanding its use of SAP (its enterprise resource planning system) to improve connections to suppliers and customers. Combining the company's shipping, billing, and sales data makes it easier to track sales and keep on top of billing problems. The company also implemented a sales management system to provide better information to salespeople.

Coming from the pharmacy side of the business, Dunbar knew that it was critical for the scientists to not only understand the value of the technology but also define and shape it to make it useful. He notes that "even as the projects move into production, it's senior-level scientists who are championing the new tools and evangelizing about their benefits" (Overby 2002). To improve communication between the IT staff and scientists, the IT development teams hold meetings at least once every two weeks between the two groups. Moreover, Dunbar doubled his IT department staff to 2,700 people, with an emphasis on hiring people with dual computer science and biology or chemistry degrees. In 2003, Mike Heim was appointed as the new CIO. Mike began his career at Lilly as a systems analyst in 1979.

In 2008, Lilly signed an agreement with TopCoder Inc. to help build some of its large applications for global drug discovery operations. TopCoder sponsors worldwide competitions trying to attract creative programmers. Code written by the programmers is stored in libraries that can be used by sponsoring companies such as Lilly (Havenstein 2008).

From various reports, Lilly uses Amazon Web Services (AWS) to handle some computing tasks. Using the cloud services is faster than trying to buy and set up a new physical server. But, rumors also exist about issues surrounding the basic contract and potential liability issues (Vaughan-Nichols 2010).

In 2010, Lilly began reducing costs by eliminating 5,500 workers, including 340 information technology jobs in addition to 140 cuts due to layoff, retirement, and resignations; out of 1,250 IT workers (Sears 2010). By 2013, drugs generating half of Lilly's revenue will lose patent status and face competition from generics.

Marketing

Developing a new drug is only the first step to making a profit. Companies also have to convince physicians to prescribe the drugs. With competition, and a huge array of new drugs, physicians need to evaluate the benefits and side effects. In the past, drug companies had sales representatives call on physicians. But Brian Weed, global marketing and sales IT director for Lilly, points out that "doctor's offices are closing their doors to pharmaceuticals sales representatives and citing HIPAA and privacy concerns as reasons" (Greenemeier 2003).

Sales force management and customer relationship management software are becoming more important at the pharmaceutical firms. Salespeople need to tell a consistent story to physicians. In terms of patients, Lilly is encouraging customers to contact the drug firm directly, instead of first going to their physician (Greenemeier 2003). The direct contacts can be beneficial in building brand awareness. As Lilly learned with Strattera, the call centers can provide more immediate feedback on actual usage and side effects. Ultimately, the process might require tighter links between the patients, physicians, and pharmaceutical firms, but those are not yet on the drawing boards.

Over the years, Lilly, and many other pharmaceutical firms, have faced bad publicity with various lawsuits. A common complaint is that the drug firms withhold or bury important information from physicians, usually regarding side effects. Thousands of plaintiffs claimed that Lilly's Zyprexa's anti-psychotic drug led to obesity and diabetes. In an interesting twist, a doctor and a lawyer involved in the case obtained many of the documents from plaintiffs and posted them on a Web site. Lilly attempted to obtain an injunction to (1) get the documents returned and (2) have all copies removed from all Web sites. Jack B. Weinstein, a federal judge in Brooklyn, ruled that the documents had to be returned, but it would be impossible to enforce any court order to prevent Web sites from displaying copies (Thomas 2007).

Questions

1. How does Eli Lilly use databases to improve efficiency?
2. What types of data does Lilly store in its databases?
3. How could Eli Lilly use additional information technology in marketing?

Additional Reading

Claburn, Thomas, "Eli Lilly Recovers Confidential Documents But Loses Secrets To The Web," *Information Week*, February 15, 2007.

Ewalt, David M., "Roy Dunbar," *Information Week*, December 15, 2003.

Greenemeier, Larry, "Pharmaceutical Makers Turn To CRM," *Information Week*, October 29, 2003.

Havenstein, Heather, "Eli Lilly Aims to Tap 'Rock Star' Programmers," *Computerworld*, May 14, 2008.

McDougall, Paul, "Drug Company Eli Lilly Outsources Clinical Data to India," *Information Week*, November 20, 2006.

Murphy, Chris, "Answers That Matter," *Information Week*, February 2, 2004.

Sears, Don E., "Eli Lilly Eliminates 340 IT Jobs," *eWeek*, July 16, 2010.

Overby, Stephanie, "They Want a New Drug," *CIO*, October 15, 2002.

Vaughan-Nichols, Steven J., "Falling Through the Clouds," *Computerworld*, August 3, 2010.

Case: GlaxoSmithKline

GlaxoSmithKline (ticker: GSK) is one of the leading global pharmaceutical firms. Created through the merger of several firms over the past decade, it is headquartered in England and has over 100,000 employees worldwide. Like the other large pharmaceutical firms, GlaxoSmithKline (GSK) is suffering from the transition of its blockbuster drugs to generic status. It could easily lose \$3 billion to \$4 billion in revenue from its antidepressants Wellbutrin and Paxil and the antibiotic Augmentin ("Business: Glaxo" 2004). Because the company is headquartered in England, but almost half of its sales are in the United States, profits have suffered from the 35 percent depreciation of the dollar in 2003-2004. Nonetheless, CEO Jean-Pierre Garnier is optimistic. Through the mergers, he was able to trim \$6

billion in costs. He has rearranged the research teams into divisions focused on specific areas (cancer, AIDs, and so on). More important, the company has 80 new products in the pipeline and half have reached advanced stage clinical trials (“Business: Glaxo” 2004).

GSK faces other issues. In 2004, the New York attorney general, Eliot Spitzer, filed suit against GSK for concealing negative data about Paxil. Spitzer contends that of all the studies Glaxo performed, only one generated the results they wanted to promote the drug. “Their effort to suppress the other studies was harmful and improper to the doctors who were making prescribing decisions and it violated the law” (Martinez 2004). The primary concern is that Paxil might lead to suicidal behavior if used in children under 18 years of age. Glaxo notes that Paxil was never approved for use by children in the United States and was not promoted that way. Glaxo spokesperson Mary Anne Rhyne notes that “there are many, many studies each year. It’s impractical to believe that every company in the industry will be able to publish from every study” (Martinez 2004). But Glaxo’s problems are compounded by a couple of internal memos. One noted that the company would have to “effectively manage the dissemination of these data in order to minimize any potential negative commercial impact.” Another stated that “it would be commercially unacceptable to include a statement that efficacy had not been demonstrated” (Martinez 2004).

In response to these problems, several organizations have called for a clinical trial registry to track the progress of all trials and outcomes. The leading pharmaceutical companies are also backing the registry. However, the process is not that simple. Many of the Phase I trials are designed to test for side effects and do not focus on efficacy. So, some drugs might not appear to be useful. Some companies have noted that the National Institute of Health already maintains an online registry (www.clinicaltrials.gov), and it could be expanded to include results from all Phase III trials (Hovey 2004).

Relying on blockbuster drugs for profits is a risky strategy—particularly when bad news erupts concerning the heavily-used drugs. In 2007, some medical information was reported suggesting that GSK’s second-biggest selling drug Avandia, a diabetes treatment, was linked in some way to a risk of heart attack. The data on both sides of the argument appeared weak and additional studies were being conducted. Nonetheless, physicians had basically stopped prescribing the drug for new patients. In managing the crisis, Jean-Pierre Garnier, CEO of GSK suggested five tips (Whalen 2007):

- (1) Fight data with data.
- (2) Communicate to employees. Daily phone calls are essential.
- (3) Study doctor opinion to catch any changes.
- (4) Put data on company Web site so everyone can see.
- (5) Keep working on long-term business goals.

In 2007, it was estimated that GSK spent about \$560 million on U.S. consumer ads; which is only a fraction of the \$5.2 billion spent annually on U.S. pharmaceutical ads. But, in part to head off potential new regulations, Glaxo’s CEO Andrew Witty said the company would cut back on television advertising. (Whalen 2009).

Information Technology in Research

In one sense, pharmaceutical R&D is similar to a university. Sharing knowledge within the organization is a key factor. The company generates and acquires information in the form of documents—some electronic, some in paper. GSK built a

centralized library system (Lynx) to give researchers immediate access to the documents. The system provides access to over 2,000 electronic journals. Requests for paper books or articles are handled by librarians, who first try to locate the appropriate title within one of the company's nine libraries. If it is not available, they check with their outside suppliers. Of course, the company would prefer to digitize all paper documents, making it simpler to deliver the information, plus make it available to other users later. However, copyright limits the number of documents that can be digitized and prevents the company from disseminating them (Delaney 2003).

Genetics, particularly individual differences among patients, is one of the big target areas for pharmaceutical firms. Glaxo has forged a relationship with First Genetic Trust to study how variations in individual DNA can affect the efficacy and side effects of various drugs. Glaxo even built a proprietary high-speed network to communicate with First Genetic Trust. The company uses data mining and huge databases to compare drug effects across individuals. The main system runs on several HP 16L-Series 9000 UNIX servers. The company uses commercial applications to enable patients to provide informed consent and specify how their genetic data can be used. In general, only the genetic data without personal patient identifiers is given to Glaxo researchers (Greenemeier 2002).

In 2009, GSK was looking at many of the same cost issues facing other pharmaceutical firms—notably the number of profitable drugs losing patent status. VP of IT Ingo Elfering said that he was using this pressure to push for changes in the IT operations. The company signed a deal with Microsoft to shift standard e-mail, calendars, and collaboration services to Microsoft's hosted system (Lai 2009). The deal will transfer 100,000 employees from the existing IBM Lotus Notes to Microsoft Exchange, SharePoint, and Office Live Meeting. The company expects to reduce costs by 30 percent.

Questions

1. How does information technology help Glaxo in its research?
2. Would a national database of clinical trials prevent the problems that Glaxo had with approval of its Paxil drug?
3. Assuming that new drugs are developed that are tailored to specific DNA markers, how would you build an information system to take advantage of this data? How will you handle the privacy issues?

Additional Reading

Delaney, Emma L., "Glaxosmithkline Pharmaceuticals Research And Development," *Interlending & Document Supply*, Vol 31(1), 2003.

The Economist, "Business: Glaxo's Big Challenge," May 15, 2004.

Greenemeier, Larry, "Genetic Research Drives High-End Computing," *Information Week*, February 18, 2002

Hovey, Hollister, H., "J&J Adds Its Support For Idea Of Clinical Trial Registry," *The Wall Street Journal*, June 18, 2004.

Lai, Eric, "GlaxoSmithKline Deal Highlights Microsoft's Overseas Launch of Hosted Collaboration Software," *Computerworld*, March 2, 2009.

Martinez, Barbara, "Spitzer Charges Glaxo Concealed Paxil Data," *The Wall Street Journal*, June 3, 2004.

Whalen, Jeanne, "Glaxo's Garnier Is Taking the Heat," *The Wall Street Journal*, July 9, 2007.

Whalen, Jeanne, "Glaxo to Pare Ads on U.S. Television," *The Wall Street Journal*, January 9, 2009.

Case: Bristol-Myers Squibb

Bristol-Myers Squibb (ticker: BMY) has headquarters in New York City and employs about 44,000 people worldwide. One of the company's recent innovations was the development of TAXOL to treat breast cancer. The company was able to create a synthetic compound that duplicated the original drug derived from the bark of the Pacific Yew (an endangered tree). In 2004, Bristol-Myers Squibb was approved to market the genetically created drug Erbitux in conjunction with ImClone Systems. You might remember the drug as the one that led Martha Stewart to be found guilty of lying to SEC investigators. Bristol-Myers Squibb also sells many consumer-level products (www.bristolmyers.com). In 2006, the company earned \$1.6 billion on sales of \$17.9 billion. However, 18 percent of sales revenue came from one drug: Plavix. The patent for Plavix expires in 2008 in Europe and 2011 in the United States (2006 Annual Report).

Research Data

Laboratory equipment today is generally automated. Often, it can be controlled by computers and most machines generate data that can be collected by computers. The catch is that machines can generate enormous amounts of data. BMY was having trouble keeping up with the amount of data generated. The BioAnalytical Sciences (BAS) group turned to the Scientific Data Management System (SDMS) developed by NuGenesis Technologies Corporation. The system collects all incoming data and reports and archives them to a central repository—which frees up the hard drives on the local workstations. The data and reports are then available to other researchers from a secure Web site. The system also provides access to the binary data, which is needed for compliance with federal laws. The system enables researchers to tag the raw data by researcher, project, analytical method, and so on. It collects data from diverse systems and makes the data accessible via a common interface. The BAS group connects to over 100 instruments in two sites in New York and New Jersey. The central server also backs up two main servers. Because the system consolidates historical data, it makes it easy for researchers to compare current results to older analyses. Data is stored in a secure Oracle database and marked with time-stamp and audit trail data so everything can be tracked. In a six-month pilot test, the system generated 5,000 reports. The system automates most data management tasks, freeing up time by researchers and IT administrators. More important, it enables greater collaboration because the data and reports are accessible throughout the company, regardless of the geographic location or software installed on a specific machine (DeVincentis 2002).

Knowledge is critical to pharmaceutical researchers. And knowledge is vastly more complex than simple data. How can a researcher in one location quickly find an expert in another division of the company? How does the research in one area compare to that by another department? Bristol-Myers Squibb purchased tools

from LexiQuest to help solve these types of problems. LexiQuest is a linguistic search specialist, providing the ability to search based on information instead of just key words. LexiQuest Guide uses language-recognition technology to search documents based on everyday language. It semantically analyzes each document to evaluate its content. The two companies are also building a custom dictionary and search system to handle the specific terms used in pharmaceutical research (Jezzard 2001). In essence, the system reads the documents, enabling it to provide a more precise match to researcher requests. By providing more accurate and faster responses, researchers can save considerable time. And time is critical when you need to bring a new drug to the market.

Business Operations

Pharmaceutical firms are pressured by expiring patents, government attempts to reduce drug costs, and increased competition. Like any business, they are turning to information technology to reduce costs and improve the business operations. Bristol-Myers Squibb is turning to the Web for its purchasing needs. It acquired Ariba's Buyer e-procurement software in 2000, saving over \$90 million a year within the first year. The system standardizes purchasing methods. Employees who need to buy something, such as a PC, check the suppliers and options on the Web site. Then a request for quote (RFQ) is posted on the Ariba network and suppliers bid for the contract. Employees can also use the system to purchase smaller items directly. The system aggregates the purchases where possible and orders in bulk. In 2001, as much as 16 percent of sales orders were conducted over the system (Yasin 2001).

In 2007, Bristol-Myers Squibb lost an important initial lawsuit. U.S. District Court Judge Patti B. Saris in Boston ruled that Bristol, AstraZeneca, and Schering-Plough were guilty of inflating average wholesale prices (AWP) for certain drugs. Some insurers use the AWP to establish reimbursement rates. On the other hand, Medicare stopped using the list in 2003, largely because of the inaccuracies that it perceived. Judge Saris wrote that "unscrupulously taking advantage of the flawed AWP system...by establishing secret mega-spreads far beyond the standard industry markup was unethical and oppressive" (Tesoriero and Korn 2007).

In 2008, Bristol faced the many of the same issues as the other pharmaceutical firms—and is also responded by cutting jobs. It cut 10 percent of its workforce in 2008 and again in 2009. The goal was to reduce costs by \$2.5 billion (Wang 2008).

Questions

1. Can machines replace lab workers and scientists in conducting experiments and evaluating data? What problems might arise and how would you minimize the risks?
2. How does Bristol-Myers use information technology to reduce costs—particularly in purchasing?
3. Why is a central research database system so important to Bristol-Myers?

Additional Reading

DeVincentis, John, "Performance, Speed Improve With Scaleable SDMS," *R&D*, October 2002.

Jezzard, Helen, "LexiQuest Aids Drug Co.," *Information World Review*,

December 2001

Tesoriero, Heather Won and Melissa Korn, “Ruling on Drug Pricing Faults Three Companies,” *The Wall Street Journal*, June 22, 2007.

Wang, Shirley S., “Bristol-Myers Sets More Job Cuts,” *The Wall Street Journal*, December 16, 2008.

Yasin, Rutrell, “E-Procurement is Drug for Bristol-Myers Squibb,” *InternetWeek*, June 11, 2001.

Case: Pfizer

With almost \$68 billion in sales in 2010, Pfizer (ticker: PFE) is the largest of the pharmaceutical firms (www.pfizer.com). As a research organization, the company has produced some well-known brands, including prescription drugs such as Celebrex, Diflucan, Lipitor, and Viagra. It also has a big presence in the consumer market with Benadryl, e.p.t., Listerine, Neosporin, and Roloids, among others. In 2010, the company had 15 brands generating over \$1 billion a year in sales (2010 Annual Report). It has 130 potential drugs in the pipeline (Overby 2002). But, \$12 billion of revenue came from Lipitor, the biggest-selling medicine ever produced by any pharmaceutical company. But Lipitor faced competition from generics in 2010, and the company has had no success in creating another blockbuster drug. The company had been betting on a new drug torcetripib to boost “good” cholesterol—but a 15,000-patient study was cancelled when patients developed high blood pressure and started dying (Simons 2006). Pfizer officials tried to impress investors by revealing it had 242 research programs in progress. But it is difficult to say that any of the drugs will make it to market or produce the blockbuster revenue Pfizer depends on. Hank McKinnell, the CEO who engineered many of the mergers that made Pfizer a giant, stated in 2002 that “size helps. Chemical suppliers return our calls faster, we can run very large-scale global clinical studies, and we can try out the newest technologies and implement them rapidly in order to do things others can’t. So, there are real advantages to scale.” But the board of directors ousted McKinnell in 2005 when the promised gains did not appear. Additionally, some people are concerned that the big companies are creating fewer new drugs. Submissions to the FDA fell from an average of 41 per year to 27 per year, and many of those are created by smaller companies but licensed by the big four (Simons 2006). The company also created a Web site to enable investors and potential customers to track drugs in the pipeline (www.pfizer.com/pipeline). The concept of making this information openly available is a radical change for any pharmaceutical company.

Pfizer continued its expansion by purchasing Pharmacia Corporation in 2003, Esperion Therapeutics in 2004, and Vicuron Pharmaceuticals in 2005. Closing off a co-development agreement with sanofi-aventis, Pfizer purchased the worldwide rights to Exubera an inhaled insulin therapy. However, Pfizer did sell off its consumer products division to Johnson & Johnson for \$16.6 billion in 2006 (2006 Annual Report). In 2010, Pfizer acquired Wyeth, another large pharmaceutical firm (2010 Annual Report).

The company faces many of the same problems as the other pharmaceutical companies. One of the biggest issues is how to integrate data and knowledge across the company—particularly when growth came through mergers.

Walter Hauck, vice president of worldwide informatics at Pfizer, knew he faced a huge problem in trying to integrate the many research departments. He needed to create tools that would support collaboration not only among the scientists but also among the lawyers and marketers who bring the drugs to market. He began with an ambitious task: integrate all of the changes at once. At La Jolla Labs in California, his IT team introduced one new application a week for six months. The scientists almost rebelled, saying that the sheer number of changes was taking away time from their research. Hauck heard similar complaints from many other divisions: “It was a lot of change to drop on people while expecting them to continue to deliver” (Overby 2002). With little control over introducing new tools, some employees were asked to sit through training sessions on the same product multiple times. Hauck knew it was time to implement a change management strategy. One of the keys was to standardize the process by creating a checklist, making sure that the business units needed the changes and understood the value. The lists also ensured that the developers captured the feedback and evaluated the pilot tests accurately.

Pfizer also tried to help employees by creating a new application to enter expense reports. But, the new system was harder to use than paper forms, help calls increased, and many people stopped using it. Joseph C. Schmadel Jr., senior director of business technology, observed that “that was an indication to us that something was wrong.” To solve the problem, Pfizer brought in netNumina, Inc., a consulting firm, to rethink the user interface. By rebuilding the user screens and linking them to the back-end reporting system, the system became easier to use and still collected the necessary data. The system also creates a digital dashboard that provides summary statistics for executives (Weiss 2004).

Integrating data is critical to a company as large as Pfizer. The company had 14 different financial systems and wanted to combine the data into a data warehouse. The IT group built a data warehouse as a central repository, and then built links to each of the individual systems. But the IT department quickly encountered a problem. Each of the systems had different definitions for the data. It took months to clean up the data so that it could be combined and managed as a single source. Danny Siegel, senior manager of business technology, says, “We saw that we had to put in place some rigorous data standards. This kicked off a six-month, totally non-technical effort to devise a set of standards that allow users to slice and dice data in whatever context they need it” (King 2003).

Vita Cassese, VP of global business technology at Pfizer summarized the challenges the company faces in trying to reduce costs but still support research. She said “there are two opposing challenges. We need to drive a level of consistency and standardization to operate effectively while still fostering innovation.” She also noted that the size of the business makes it more challenging to manage IT, noting “in many ways, it demands different approaches to data demand and information management. That makes for very big challenges, because we also need to drive down the costs of managing this information” (King 2006). Pfizer has tackled these challenges by centralizing the primary IT services, including ERP, the data center, help desk, and networking. But local support is provided by placing IT staff members into the various divisions around the world. Ms. Cassese noted that “IT still needs to be close to the business and have a deep, deep understanding of the business, but we also need a deeper level of discipline around how we deploy technology” to control the costs. To reduce costs, the company is trying to centralize much of the data handling. In 2005, the company initiated a

project to consolidate 30 document management systems. The company is also trying to standardize desktop hardware. The document consolidation program is also driven by new federal rules that require greater sharing of data on drug trials and submission of labeling data in XML. Consequently, the system will use XML to store all of the data, integrating information across divisions in 26 nations.

Because Pfizer has acquired or partnered with many companies since the introduction of computerized database management, the company has a staggering amount of legacy data. The difficulty is “to get the data into a shape where it can be easily accessed by researchers that span the globe and be nimble enough to be able to accept new data and new systems as more companies come in via mergers and acquisitions. In addition, they need to convince scientists that what they have is not just an archive of past experiments but also a rich resource of information for future drugs” (Derra 2004).

Pfizer, like many other pharmaceutical companies, is pushing to become more efficient in drug discovery, and is looking to its legacy data as a source of useful information and insights into new drug development. But the sheer number of legacy systems within even one research facility is overwhelming. Bhooshan Kelkar, PhD, advisory software engineer at IBM Life Sciences, Dallas, says that the comments of a colleague speak to the enormity of the issue. “He was working with a big pharmaceutical company last year and reported that they had 500 legacy applications just in their discovery and clinical area...It was costing them 50% of their IT budget every year, just in maintenance and reconciliation of different legacy systems. That is huge.” The key to unlock the power of mining legacy systems will be a combination of robust systems, standard methods, consistent data, and cultural change from within the drug discovery environment (Derra 2004). While the task is daunting, it is essential to new drug development to effectively catalog all the known compounds within Pfizer’s extensive library in order to rapidly develop new pharmaceuticals or delivery methods. Transferring all of the corporation’s legacy data to a common format is the best possible way for researchers to have quick access to important older data.

This same need for a universal data standard for collecting, sorting, and processing clinical trial data for FDA submission is currently occupying the entire pharmaceuticals industry. But the data-collection standards have been slow to catch on. However, in June 2004, a consortium of American pharmaceutical companies debuted a data-interchange standard for electronically submitting drug-approval applications to the U.S. Food and Drug Administration. This could speed the entire drug-review process, reducing the time between submission and FDA approval. The major pharmaceutical companies—including Eli Lilly, Merck, Aventis, and Pfizer—are the sponsors of the move, under the heading of the Clinical Data Interchange Standards Consortium (CDISC). Among the standards are a universal file structure and submission form, both within XML programming. CDISC previously developed standards for collecting lab data and clinical-trial data such as patient information. Those are designed to simplify the collection of clinical-trial data by pharmaceutical companies and the contract research organizations—such as Quintiles Inc.—that drug makers hire to manage clinical trials (Whiting 2004). The FDA will not require CDISC standards for drug-approval applications, but it has endorsed CDISC’s efforts.

In 2007, a Pfizer employee used a company laptop to work at home. She installed a peer-to-peer file sharing application on the computer, and inadvertently made the entire machine a sharing device on the P2P network. The action exposed

the personal data of 17,000 Pfizer employees, including Social Security numbers. Pfizer reported the breach and noted that various outsiders had downloaded data on about 15,700 employees (Vijayan 2007).

With 86,000 employees, and several thousand scientists in R&D, it can be challenging to coordinate and share information. So Pfizer customized Imaginatik's Idea Center tool to enable employees to submit ideas for new products or process improvements. Rob Spencer, a senior research fellow noted that the tool saved the company at least \$20 million and helped solve hundreds of business problems (Rosencrance 2010). The goal is to use collective intelligence of the employees to attack difficult problems. The ideas or comments from one person can trigger ideas or solutions by other people.

Pfizer is facing the same cost-cutting challenges as other companies. The acquisition of Wyeth and King help increase total sales, but the company closed major research facilities in 2010 to achieve a 24 percent reduction in R&D costs. Pfizer's CEO Ian Read noted that "This is a fundamental change in culture. We have to fix this innovative core" (Loftus 2011).

Questions

1. Why did Hauck introduce so many new applications at one time? Why was it such a failure? Could you have predicted that outcome?
2. Why does Pfizer have so many different research systems and why is it so difficult to integrate them? What information technology tools might help?
3. How did Pfizer manage to have so many different financial systems?

Additional Reading

- Derra, Skip "Legacy Systems: More than Old Data in the Attic" *Drug Discovery and Development*, May 1, 2004.
- Havenstein, Heather, "Rules Prompt Pfizer to Consolidate Content Management Systems," *Computerworld*, May 23, 2005.
- King, Julia, "Business Intelligence: One Version of the Truth," *Computerworld*, December 22, 2003.
- King, Julia, "Pfizer: Striking a Balance," *Computerworld*, October 30, 2006.
- Loftus, Peter, "Pfizer to Cut Research, Shift Funding to Buybacks," *The Wall Street Journal*, February 1, 2011.
- Overby, Stephanie, "They Want a New Drug," *CIO*, October 15, 2002.
- Rosencrance, Linda, "Tap the Wisdom of Employees -- and Boost the Bottom Line," *Computerworld*, February 24, 2010.
- Vijayan, Jaikumar, "Personal Data on 17,000 Pfizer Employees Exposed," *Computerworld*, June 12, 2007.
- Weiss, Todd, "Pfizer Case Study: Hiding Complex Apps Behind User-Friendly Interfaces," *Computerworld*, March 9, 2004.
- Whiting, Rick "Standards May Speed Approval of New Drugs" *Information Week*, June 14, 2004.

Summary Industry Questions

1. What information technologies have helped this industry?
2. Did the technologies provide a competitive advantage or were they quickly adopted by rivals?
3. Which technologies could this industry use that were developed in other sectors?
4. Is the level of competition increasing or decreasing in this industry? Is it dominated by a few firms, or are they fairly balanced?
5. What problems have been created from the use of information technology and how did the firms solve the problems?