Chapter 1

Database Processing

In This Chapter

- Sorting out the different classes of databases
- Discovering what databases can do for you
- Understanding database processing

Database processing is one of the more common operations performed on computers today. In fact, only word-processing and spreadsheet packages outrank database management systems among the most popular business tools. Everyone, from the largest corporate entities to private individuals, wants to keep track of something. Applications such as order entry, accounts receivable, accounts payable, and general ledger all incorporate databases. Companies keep track of their customers, product inventories, employees, and capital assets in databases. Businesses, governments, and organizations around the world would grind to a halt without databases.

The Different Classes of Databases

Large international corporations and national governments have substantially different data management needs from those of a private individual or even a small to medium-sized company. Large database users have demanding capacity and performance requirements and are willing to pay what it takes to meet those requirements. That kind of power would be serious overkill for an individual, local non-profit organization, or small business, and would be too expensive anyway. As a result, different database development tools are available for addressing different market segments. Some of these tools, called database management systems (DBMSs), are capable of supporting huge, high-performance databases, but require very powerful (and expensive) mainframe computers to do the job. Other tools run on personal computers, and are limited in the size and performance of databases they are able to support.

Enterprise databases

The first databases, back in the 1960s, although primitive by today's standards, were applied to large, enterprisewide problems, such as airline reservation systems, and maintaining bills of materials for NASA spacecraft. In those days, computers were big, expensive to buy, and expensive to run. Only large corporations or government agencies could afford to own the computers that could support a database. As a result, the first databases were enterprise class databases. The database management systems that were used to create databases were powerful, robust, and resource-hungry.

As computer power has steadily increased and become less expensive, enterprise class databases have become even more powerful and are capable of supporting much larger collections of data. The data on such systems is also accessible to thousands of simultaneous users. Today, large organizations get orders of magnitude larger and faster databases for much lower cost than was true in the early days of database, but costs of such systems are still out of reach for individual users. This is not a big problem, because few individuals need a database system that supports thousands of simultaneous users.

Personal databases

In 1975, the first, primitive personal computer kits arrived on the scene, and in 1976 you could buy one already assembled. (Pretty slick, eh?) These machines were not powerful enough to support even a very cut-down database management system, but performance improved steadily. With the advent of the IBM PC coupled with hard disk storage, database technology started to proliferate on personal computers in 1981.

Personal database products are much simpler than their enterprise class ancestors. For one thing, they have to support only one simultaneous user, rather than thousands. For another, typical single-user applications use much smaller databases than those needed to run an airline reservation system or something similarly huge. Furthermore, because there were soon millions of personal computers compared to a much smaller installed base of mainframe computers, the economies of scale kick in and it is possible to sell personal databases at a *much* lower price than mainframe databases and still make a profit. Development costs are spread over many more units.

Today, personal computers have become so powerful that the DBMS products available on them have much more capacity and much better performance than did the mainframe DBMS products of yesteryear.

The Y2K catastrophe

Remember the big Y2K scare? People were seriously concerned that on the stroke of midnight on December 31, 1999, the world as we knew it would come to an end. Well, maybe not come to an end, but terrible things would surely happen. Airliners would fall out of the sky. Elevators would drop down their shafts and crash into the building basement floor. Car engines would turn off while you were cruising at 60 mph on the freeway. Libraries would send fine money to patrons with overdue books, because the books were returned a hundred years before they were taken out. Who knows? Maybe even Pez® dispensers would cease functioning.

Billions of dollars were spent worldwide to exorcise the Y2K demon. Where did all that money go? Most of it went to modifying database files and the applications that accessed them. Some was spent on new equipment, because the threat of Y2K disaster made it easier for workers to convince management that, to be safe, they needed new Y2K-compliant computers or Pez dispensers.

Workgroup databases

After millions of personal computers had been sold and installed in companies large and small, people came to a fundamental realization. Millions of people, each with their own personal computer, now had far greater ability to do their work faster and with less effort than had been the case before. Productivity had taken a quantum leap forward. However, each one of those personal computers was an isolated island of compute power and data storage. Productivity would be boosted even more if somehow the data and compute power residing on those personal computers could be shared.

Networking connected the personal computers together, and a new class of database — the workgroup database — was invented to take advantage of the new connectivity. Workgroup databases, accessed by perhaps up to 50 or 100 simultaneous users, filled the gap between the enterprise database and the personal database. Today, in small to medium-sized organizations, workgroup databases are the most common of the three database classes.

So Much Data, So Little Time

Ever since electronic computers first came into use in the late 1940s, they have generated data of all types much faster that had ever been possible using adding machines along with paper and pencil. Since then, the power of computers has been increasing at an exponential rate. Moore's Law, named after Intel co-founder Gordon Moore, has held true for decades, stating that the power of computers doubles about every 18 months, as shown in Figure 1-1.

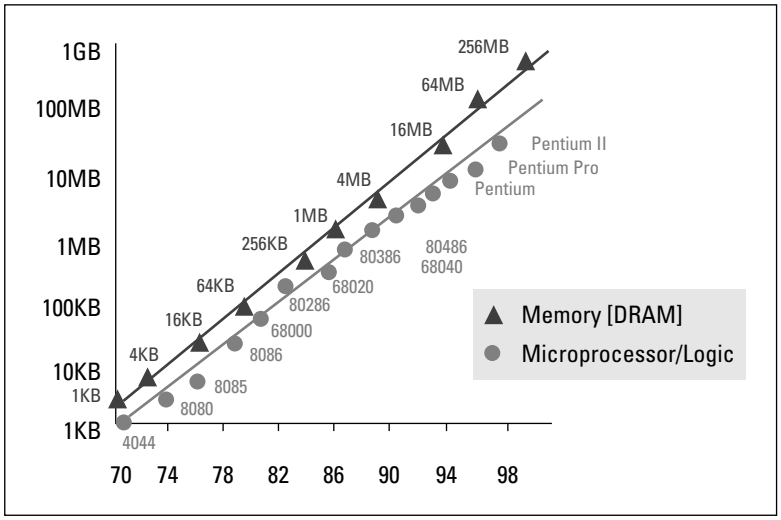

Figure 1-1: Growth of computer power as a function of time.

The amount of data that computers are able to process doubles at a comparable rate. As a result, we are being drowned in a veritable sea of data. Much of it is potentially valuable, but the situation has reached the point where data is being gathered so fast that much of it may never be put to use. Raw data has very little value. It gains value when it is organized in such a way that it conveys meaningful information to people who can use that information. Databases are our most powerful tool for organizing data into potentially valuable information.

Databases and privacy: We know who you are, and we know where you live

One of the unanticipated consequences of the tremendous growth in the amount of data that is generated every day is the erosion of personal privacy. A generation ago, as long as you were not a famous or notorious person, nobody knew much of anything about you. Your private life was just that, private. If you wanted to drop out of sight, move somewhere else and start a new life, it was not very difficult to do. Aside from a small number of people in your local community who had lived and worked with you, you were a complete unknown to the world at large. Those days are gone and will never return.

Now it is practically impossible to buy anything, sell anything, or travel anywhere by air, rail, or sea, without the fact being recorded in a database somewhere. Ever since the days of J. Edgar Hoover, the FBI has prided itself in its ability to know the whereabouts of and important facts about individuals it

considers important. Nowadays, you don't have to be the FBI or the CIA to have that kind of knowledge about anyone you care to know about. Merchants, airlines, and travel agents have data on your living and buying habits. With the recent rash of mergers of all kinds of organizations into larger entities, this data is becoming centralized. Residing in databases that can be "mined" for useful information, companies can find out not only who you are and where you live, but also what you like to eat, what you like to read, who your favorite musicians and entertainers are. They know what your favorite sports teams are, and what sports you like to participate in yourself. They know where you shop and how often. They know when you are about to run out of something you buy regularly. They know when your kids are born, when they are about to enter kindergarten, when they will graduate from high school, and when they are engaged to be married.

All this data is stored in databases. The databases are growing larger, not only because more data is added to them on a daily basis, but because new kinds of data are being captured and stored, based on the activities and transactions that you participate in, in the course of your daily living. The amount of data being stored in databases every day, based on people's actions and transactions, is already huge, but will get even larger in the coming months and years.

Bottom Line: Although databases are constantly getting larger, even data stored in huge databases can be quickly and easily processed to give users exactly the information they want.

Amazon.com and the online merchants

The rise of e-commerce on the World Wide Web has accelerated the accumulation of data about people. Records are kept of people who visit Web sites, and even more elaborate records are kept about people who actually buy things at Web sites. Many Web sites require users to register before allowing access to their best content. By registering, the user reveals personal information that the site then uses to construct a user profile. The profile enables the site to display personalized content to visitors. For commercial sites, this means users are more likely to become buyers, because they are being presented with advertisements and other content that are tailored to their interests.

Amazon.com, the largest retailer on the Web, has perfected the technique of using databases to characterize its customers. By analyzing the kinds of products you have bought or expressed interest in, in the past, Amazon.com can present you with displays of similar products that you are likely to find interesting. This sales strategy requires not only massive, well structured databases, but also sophisticated data mining software that finds associations and relationships in customers' past behavior that allow predictions of what they are likely to do and want in the future.

Other online merchants are following Amazon's lead and using databases and data mining technology to offer visitors a customized experience. This is good in that people are not presented with content they are not interested in, or advertisements for products that do not interest them. It is potentially bad because merchants will know a lot about people, and that knowledge could be abused.

Bottom Line: Like it or not, unless you are a hermit living in a cave, people you don't know and have no reason to trust know many intimate details about your life. If you use checks or credit cards, your life is an open book. If you buy things from merchants such as Amazon.com on the Web, that book is an international bestseller.

Data deluge: It came from outer space

The United States has been launching rockets into earth orbit since 1958, and ever since, satellites have been radioing high-capacity streams of data back to Earth. Russia, Japan, China, and the European Space Agency do the same thing and are also receiving vast quantities of data from their space probes. All this data is being stored in the hope that someday it will be analyzed and human knowledge will be advanced as a result. The most promising data is analyzed fairly soon after it is received, but the large majority of data returned from space is not examined for years, if ever. The speed with which we collect data far exceeds the speed with which we can analyze it and draw useful conclusions.

In 1994, the Clementine spacecraft orbited the moon for several months, taking data on its entire surface. That data is stored on 88 CD-ROMs, each holding about 640MB, for a total of 56GB of data about the lunar surface. The data is in raw form, cataloged by orbit number. Only a small portion of the data has been examined in detail, particularly data about the areas around the North and South Poles. The spectral signature of water that appears in the data from the polar regions has caused excitement among scientists and advocates of space exploration. However, because the entire dataset is not organized into a database, it is difficult to search for specific features and make generalizations about them.

Another spacecraft, Galileo, has been studying the Jupiter system for several years. It also has sent back huge amounts of data. By studying a small fraction of that data, space scientists have inferred the existence of a global ocean under the ice covering the surface of Jupiter's moon Europa, and the probable existence of a similar ocean under the surface of Callisto. However, the large

majority of Galileo's data remains unstudied, because in its unorganized state, it is very difficult to extract useful information from it, unless you already know what you are looking for. Organizing the data into a database would be of tremendous benefit, but NASA has no funding for such a massive effort.

Mars Global Observer, currently orbiting Mars, has returned huge quantities of data to Earth. Some of this, such as dramatic high-resolution photographs of the Martian landscape, has been analyzed and reported upon. Most, however, has merely been archived, against the day when resources will allow it to be studied. In its current raw form, it is practically impossible to discern patterns in the data that might lead to greater understanding.

Bottom Line: It is difficult to wring information from large datasets that are not organized into databases. As a result, much of this data sits in archives, unused. If the storage medium (magnetic tapes or disks, photographic film) is not maintained properly, the data could degrade to the point of being unusable. In that case, all the effort that went into collecting it is lost.

The fierce urgency of now

The data explosion is out of hand and getting worse every day. The only hope for making sense out of the floods of data that we are receiving is to organize it in a way that allows fast, efficient retrieval of just the information we want, regardless of how large the dataset is. The longer you wait to perform that organization, the harder it will be to do. If you are in business, your competitors are using databases to get a handle on their data. If you don't do the same, and soon, they will gain a huge competitive advantage. Whether you are just starting out, and as yet have not collected any data, or you are in an established organization that has been collecting data for years, there will never be a better time than right now to decide the best way to organize your data so that you can quickly receive answers to the questions you will want to ask both now and in the years to come. After you decide what kinds of information you are likely to want to extract from the data, you can design a database that will make it easy to do so.

Bottom Line: It is a good thing you are reading this book. Unless you have the cash to hire a highly paid database guru, you need to understand how to design a database, so you can do it yourself. Even if you *do* have the cash to hire a highly paid database guru, you still will be better off if you understand what that expert does for you. If you would like to *become* a highly paid database guru, this book will start you on your way, and serve as a valuable reference after you arrive.

What Is Database Processing?

By this time you are probably asking, "What makes database processing so wonderful?" You can understand the value of organized data compared to unorganized data, but there are many ways to organize data. What makes the database structure so superior to just plain old storing things in a consistent, logical order? That is a valid question. To answer it, I describe how computer scientists organized data before databases came into use, so you can see the advantages and disadvantages of that method. Then, I explain how the structure of an information system based on database technology differs, along with the advantages and disadvantages of the database method. On balance, the advantages of using database technology far outweigh the disadvantages.

File processing: The old way

As shown in Figure 1-2, computers consist of three principal subsystems: the processor, the memory, and the input/output subsystem, which includes such components as the keyboard and the monitor's screen. The processor performs all the computations and operations. The memory stores things when they are not being processed. The input/output subsystem conveys information back and forth between the computer and the user. You control computations and other operations with program code, which is stored in memory. The data that you operate on is also stored in memory.

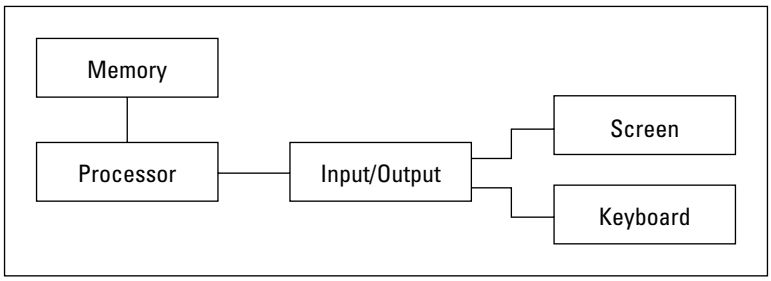

Figure 1-2: The principal subsystems of a computer.

To early computer scientists, it made a lot of sense to rigidly separate the memory used to store program code from the memory used to store data, as shown in Figure 1-3. For most applications, data changes frequently while an application is processing it. On the other hand, it is very dangerous to allow program code to change while that very same program code is executing. More often than not, such self-modifying code causes what have come to be called computer crashes.

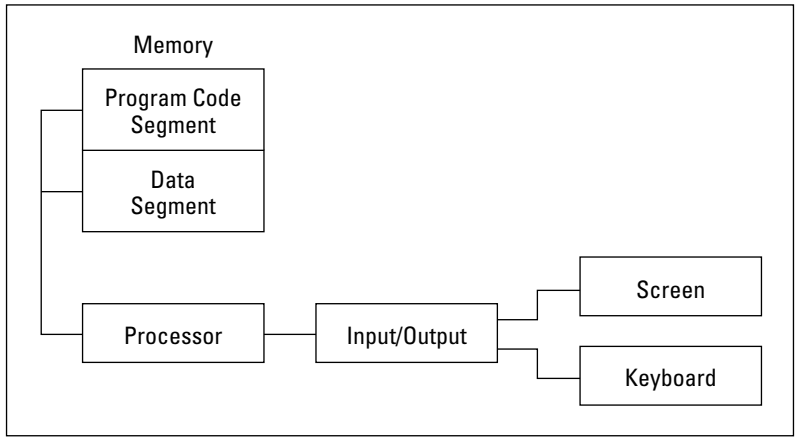

Figure 1-3: Keep program code and data separate in memory.

Without a doubt, it is good to keep program code separate from data. However, this thinking carried over into how computer files were structured. Early data files contained nothing but data, as shown in the following example:

```
Harold Percival26262 S. Howards Mill Rd Westminster CA92683
Jerry Appel    32323 S. River Lane Rd   Santa Ana   CA92705
Adrian Hansen  232   Glenwood Court     Anaheim     CA92640
John Baker     2222  Lafayette St       Garden GroveCA92643
Michael Pens   77730 S. New Era Rd      Irvine      CA92715
Bob Michimoto  25252 S. Kelmsley Dr     Stanton     CA92610
Linda Smith    444   S.E. Seventh St    Costa Mesa  CA92635
Robert Funnell 2424  Sheri Court        Anaheim     CA92640
Bill Checkal   9595  Curry Dr           Stanton     CA92610
Jed Style      3535  Randall St         Santa Ana   CA92705
```

Such files, often called *flat files*, are the kind used by early computer languages such as COBOL and Fortran. Application program files contained everything necessary to find a desired file on a storage device, and find specific desired items within the file. This architecture worked well for many years, but did cause some problems.

One big problem with the flat file architecture had to do with the fact that often multiple application programs dealt with the same data file. For instance, a business's CUSTOMER file might be used by several accounting applications, several applications used by the sales department, others used by marketing, and a few used by top management. If, for any reason, the data file needed to be changed, all the applications used by all the departments would have to be updated to run with the new data structure. For example, perhaps a field was added to show which salesperson had a particular account. This wouldn't matter to accounting, marketing, or top management,

but all their applications would have to be updated anyway. Not only was this a lot of extra work, but it also made those applications subject to errors. The old adage, "If it ain't broke, don't fix it" applies in spades here. However, with the flat file structure, those unrelated applications *did* become "broke" and needed to be fixed.

Another problem with the flat file structure is not so obvious, but just as important. In today's rapidly changing computing environment, it is not wise to tie your applications to any specific hardware implementation. You can be sure that your hardware will become obsolete sooner or later, probably sooner. After a while, obsolete hardware is not supported any longer. Once you can't get replacement parts or operating system upgrades anymore, it is time to discard your old hardware. You don't want to discard your application programs along with it.

So, computer programs should be independent of the hardware they run on. That is not possible with flat file systems, because the information about the physical location of the data is included in the application programs. This fact caused major pain in the 1960s when IBM moved its customer base from the old transistor-based 709X architecture to the new integrated circuit-based System 360 architecture. IBM's customers were not happy about having to rewrite all their application code, but they had little choice.

Lesson Learned: Somehow, structure things so that application programs can access data without having to know its physical location.

Database processing: The new way

In 1945, even before the first electronic computer was built, Vannevar Bush described a structure that would solve the problem of application dependency on hardware configuration. He originated the concept of a database long before any hardware existed that could support one.

The main idea of a database is that a third structure lies between the application program and the data, as shown in Figure 1-4. This third structure is called the database management system (DBMS). The DBMS stores all the information about the physical location of data. The application refers to the data by logical address, and the DBMS converts the logical address to a physical address to access the data. Because the logical address of an item can be the same, regardless of what hardware is hosting the system, applications can migrate from one hardware platform to another without change. Changes made to the physical addresses of items in memory are transparent to the application as long as the logical address remains the same. The DBMS makes all necessary adjustments.

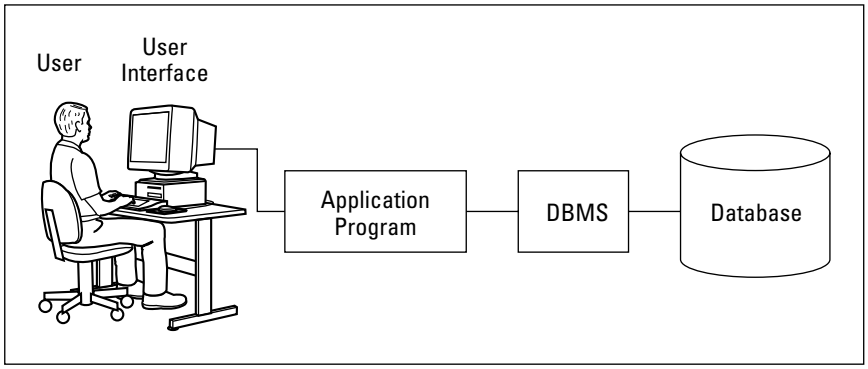

Figure 1-4: A DBMS-based information system.

Seems strange, doesn't it, that the idea of a database was known, but early computer pioneers went ahead and based their software on flat data files anyway? They had good reason, however, for this seemingly shortsighted choice. A DBMS requires a great deal of computer power, raw speed, in order to return results in a tolerable amount of time. The early computers did not have enough power to run a DBMS. As a result people designed around flat file systems. Without the overhead of a DBMS, these systems ran much faster, and meaningful work could be done on slow vacuum tube-based and later transistor-based hardware.

As computer performance improved, use of database architecture became more and more feasible. Finally, in the early 1960s, the first commercial database systems started to appear. Initially, they were used only on the largest computers, applied to the largest projects, such as keeping track of all the parts and documents associated with the Saturn V launch vehicle and Apollo moon landing spacecraft. At the time, the Saturn V/Apollo combination was the most complex machine that humanity had ever built. As computer power continued to increase, database technology trickled down to ever-smaller machines, until it became available on personal computers around 1980.

Today, robust database management systems are available on computers of all sizes and are used routinely by individuals and organizations to manage the data that is important to them.

Types of database systems

There are a number of ways that a DBMS could organize data, and there are advantages and disadvantages to each of those ways. A number of different structures have been tried, some with more success than others. IBM's DBMS for NASA's Saturn V/Apollo project, later dubbed IMS, had a hierarchical

structure. Competing products of the same era had a network structure. The evolutionary descendants of these pioneering products are still in use today. The vendors that support them have maintained compatibility over the years so that their customers can continue to benefit from the massive investments they have made in applications that use those DBMS structures.

Hierarchical databases have a simple, hierarchical structure (no surprise there, I guess) that allows very fast data access. However, as Robert A. Heinlein once pointed out, "There Ain't No Such Thing As A Free Lunch (TANSTAAFL)." You pay for that fast access with structural rigidity. Once a hierarchical database has been implemented, it is very difficult to modify. In the real world that I live in, requirements change over time. Business models change, markets change, companies grow, companies shrink, companies enter new markets or exit old ones. They introduce new product lines and abandon others that are no longer popular. This situation caused early databases to be a major bottleneck and impaired many organizations' ability to react to change in a timely manner.

Network databases were not a significant improvement, although they had different problems. In contrast to the simple relationships characteristic of the hierarchical structure, network databases allowed any item in the database to be directly related to any other item. This allowed more flexibility than the hierarchical structure, but sacrificed some speed to do it. In addition, the added complexity made network databases more difficult to maintain.

In 1970, E.F. Codd, then at IBM, published a landmark paper that outlined the basic structure of a new type of database system: the relational database. Relational databases are much more flexible than either hierarchical or network, but at the same time have a simple structure. Nevertheless, TANSTAAFL is still in force. The advantages of the relational model are offset by the fact that it carries significantly more overhead than either of the other database models. This means that it runs significantly slower. However, as computer performance has improved over time, the use of relational databases has become progressively more feasible.

Over time, the relational model has displaced the earlier hierarchical and network models in practically all new installations and is the dominant type of database in use today. In some application areas, an even newer model, the object-oriented model has gained adherents. A hybrid, the object-relational model, retains the advantages of relational DBMSs while gaining the benefits of the newer object model. However, the usage of object and object-relational DBMS products is still relatively small. In this book, I concentrate on relational database technology, with one chapter devoted to object-oriented and object-relational technology.