Chapter 1

Databases: The What, Why, and How

IN THIS CHAPTER

Identifying the elements of a database

Relating data

Solving problems using a database

Looking at FileMaker's role in streamlining data management

You see (and hear) the word *database* bandied about almost everywhere you turn. On TV, you hear it when a crime show character talks about the AFIS database (*AFIS* is short for *Automated Fingerprint Identification System*) or on lawyer shows when characters talk about *LexisNexis* (a searchable compendium of information collected from newspapers, magazines, legal documents, and so on). Similarly, in newspapers you might read about privacy concerns raised over various (often unspecified) government databases.

The frequency with which the term *database* is used actually pales, though, when you consider how pervasive databases are. A *database* is, basically, nothing more than an organized collection of information. Your address book is a database containing contact information regarding people and companies with whom and with which you interact. When you look at the baseball results in the sports pages, the box scores, current averages, and so forth are organized data — generated from the information stored in a database cataloging the activity in said baseball games. The tables in your newspaper's financial pages are reports of stock, bond, and mutual fund activity. The index and table of contents in this book are also database reports, telling you where to find specific information.

Peoples' memories are also databases, although some function much better than others when it comes to retrieving the stored data! Computerized databases strive to store and retrieve data somewhat like a person's mind does; however, computers have both strengths and weaknesses. Two of the weaknesses are that the garbage-in, garbage-out principle (or GIGO for short) applies and that the computer has no inherent capability to learn from experience. In short, the computer has no way of discerning the accuracy, relevance, or relative importance of the information it receives. On the other hand, major strengths of computers are that they're incredibly fast, obedient, and consistent in performing the instructions they're given, and they're single-minded in attempting to follow their instructions.

Database Management Systems (of which FileMaker Pro 9 is an excellent example) are the digital tools that people use to design, implement, and manage their computer databases.

The Many Faces of Databases: Lists, Tables, and Forms

When you encounter a database in the real (as opposed to cyber) world, what you see are paper-based lists, tables, forms, index cards, and the like. Like most technologies, database technology has a bit of jargon associated with it, and to make use of our (or FileMaker, Inc.'s) guidance, learning at least some of the jargon is important. Fortunately, you've probably already encountered most of the terminology. The following list provides the most elementary terms you'll need to know:

- **Field:** A discrete piece of information, such as a surname, street address, ZIP code, or date of birth
- **Record:** A collection of fields pertaining to one specific entity, such as an inventory item, a person, or a transaction
- **Table:** A set of records
- **Database:** A table or organized group of tables

Understanding the limitations of paper-based databases

In the physical world, data is usually collected from forms people fill out, and the forms are stored in filing cabinets, while the pertinent data is transposed into ledgers or onto index cards.

When a piece of data needs to be retrieved, someone (for example, a clerk) scans through the accumulated data to find the record in question. As you can easily see, this process is not terribly efficient, leaves a lot of room for human error, and can consume a lot of physical space. Even ignoring all that, creating a consolidated report summarizing some aspects of the stored data (for example, how many male students signed up for any shop course) could pose a daunting task, also fraught with the possibility for error.

Regardless of how the data is stored, it is only organized in a single order — alphabetically is probably the most common. Taking contact information as a common example, suppose you want to send out a mass mailing. The post office gives you a price break if the data is sorted by ZIP code, so you now have to manually address all the envelopes or flyers using the data in your files and then sort the mailing by ZIP code. If time is money, the time required to do all this tedious labor makes the task incredibly expensive.

Modifying data is another time-consuming task in the physical world. Suppose you're keeping your contact information in a Rolodex and one of your friends gets a new phone number — you need to cross out the old number and write in the new one or replace the Rolodex card with one containing the updated information (along with retyping or rewriting all the information that didn't change).

Entering the digital age

For a long time during the last century a debate raged based on the assumption that computers were going to replace people. By now, it should be obvious that what computers do is perform certain tasks that people performed previously, albeit much more quickly and consistently. The time freed up by computers (actually, the programs run on those computers) opens whole new sets of tasks for people to perform, tasks that aren't as repetitive or boring. For example, instead of spending hours (or days) taking inventory, you can use your sales data to track inventory levels. This data enables you to know when it's time to reorder, to identify which products are hot and which are not, and to focus on deciding which items to feature and how to promote them.

Moving your data into a computer-based system provides a couple of options right at the outset:

- You can continue to use your paper-based forms to acquire the data and then transpose it into the computer rather than to ledgers and index cards.
- You can create a digital replica of your form and acquire the data on the computer to begin with, eliminating the transposition effort for at least some of your data.

Modifying data becomes much simpler: just browse to the record, select the data that changed, and type over it — all done, without mess or superfluous effort.

Now that your data is stored electronically, you benefit from the extremely rapid searching and sorting capabilities that a computer program provides. Additionally, a task like the earlier printing example becomes faster and much less labor-intensive when you have the program print the envelopes, flyers, or labels for you, already sorted to give you the presorted mail discount.

Even if simply entering, storing, and retrieving data were the only factors, we've already shown you sufficient reason to computerize your data. However, a great database program like FileMaker Pro 9 provides additional incentives. Among other possibilities, you can summarize your data in a report, perform data analysis via calculations, share your data with others via a Web page, and automate the process of producing scheduled reports.

Preparing to get organized

Whenever you construct something, whether in the physical or cyber realm, you need to know what you have available to work with and what result you seek. For example, you can't assemble a bicycle if you only have one wheel — maybe a unicycle, but not a bicycle.

Organization is critical to success in manufacturing, construction, and database development. In manufacturing, you need to know which components (and in what quantity) are necessary to build your product. For a database, you need to know what data your end product requires to be available. For example, to produce a product catalog, at a minimum, you'll require item names, stock-keeping numbers, and prices. You might also want (or need) images of the items, color or size information, and information on whether the item is seasonal. If you want to make the catalog available online as part of an online store, you'll also want to access inventory information (to put up a "Temporarily Out of Stock" image so the customer will be aware that the item is back-ordered).

Additionally, you need to know what you want to get from your database. For example, a company might want to generate a roster of employees in any given department, another report of all employees by seniority, another by salary range, and so forth. The reports that you want to create will determine the information that you need to store.

One useful and common approach is to prototype your output to determine the necessary input data.

The Concept of a Relational Database

The simplest form of database is the flat-file database. A *flat-file database* is a single table containing all the fields in your database. One common implementation of a flat-file database is a spreadsheet. In database terms, each cell in a column is a field and each row of cells is a record.

Flat-file databases and data redundancy

As the number of fields grows, a single table rapidly becomes unwieldy. Using the company personnel example, you don't need salaries or Social Security numbers commingled with the less sensitive phone extension information, especially when producing a phone roster. If you break the data into separate, more manageable tables, you wind up duplicating some of the fields from table to table. Similarly, when recording more than one piece of information about an object, such as home and work addresses, you may end up with more than one record for the same person. Such duplication makes large volumes of data difficult to track and manage.

Back in 1970, mathematician and programmer E. F. Codd introduced the idea of what he called the *relational model* to make Database Management Systems less dependent upon a specific usage. (You can find his article at `www.acm.org/classics/nov95/toc.html`.) Codd's work provides a mathematical model that requires an understanding of set theory and predicate logic to fully comprehend. What's important to understand about Codd's work, however, is that it led to database systems that enable you to *relate* different tables to each other based upon a common field.

If your school days were far enough in the past to pre-date teachers using computers to track attendance, test scores, homework, and the like, you've quite possibly encountered a very simple example of a paper-based relational system. In those bygone days, many teachers used grade books consisting of a master page with student names and generic data and additional, narrower pages (so that the student names still showed at the left) holding specific kinds of data — attendance, quiz scores, homework, and exams, for example. In a computerized relational database, the additional pages would each be separate tables related to the master table on the common field (student name or student ID).

Opportunities for making connections

Databases are everywhere. Your address book, the library's card catalog, your checkbook stubs, your iTunes Library, the TV schedule, just about any place you turn to find a piece of information, you're accessing a database. Every time you double-click a document on your computer, you're accessing a database (on Macs, it's part of Launch Services; in Windows, it's called the Registry) to find the appropriate application to open your document.

Using a Database Management System like FileMaker Pro 9, you can create structure and order around your information, making data easier to retrieve and more relevant when combined with related data.

One useful example would be organizing all your personal financial data. The master table might be comprised of the names, locations, account types, and account numbers for savings accounts, checking accounts, credit card accounts, brokerage accounts, mortgages, IRAs, SEP-IRAs, 401(k)s, and so forth. Then you would create a table in which you would record all transactions — deposits, withdrawals, interest paid, interest collected, and so on. As a result, you could create reports from the master table summarizing net worth, retirement funding (IRAs, SEP-IRAs, 401(k)s, and other retirement accounts), or outstanding obligations. Additionally, you could create reports based on any selection of entries in the transactions table, detailing status and activity for just one account, for several accounts, for a specific time period, or for a specific type of transaction (for example, all interest payments).

Do you collect coins, stamps, comic books, or Beanie Babies? If you collect coins, you might create a master table listing the coinages you collect (pennies, nickels, dimes, and so on) and related tables of those coin types detailing the specific coins, their condition, and cost. Storing this data in such a way makes it easy to find gaps in your collection or coins of which you have multiples. Similarly, you can create inventory reports or valuation information for when you insure your collection. You can do comparable things with other types of collections.

The Anatomy of a Database Solution

A *database solution* is the combination of the tables, relations, data-entry forms, reports, and everything else surrounding your database enabling you (or your clients, your customers, or your co-workers) to enter data, control access to the data, and report on the data.

The data: Foundation and substance

If you don't have information (data) to manage, you don't need a database. The core of any database is the data *in* that database. This data is organized in tables consisting of records, with each field of a record containing a discrete piece of data.

The way your data is organized provides your database with structure — sometimes called its *data architecture* — which provides the basis for every function and procedure you perform. The decisions you make about data structure are important because they determine what will be possible and what won't, when you're working with your data.

The model for data relationships developed in the 1970s may seem abstract; however, it provides an effective way of capturing relationships that exist in the real world and replicating them in the information stored in your database. The goal is devising a structure for your data that's a good match for the things that data represents — and the relationships between them.

An ideal database structure is one that captures information about things (people, objects, places, and so on) and also accurately represents the relationships between them. People have relationships with each other — family and work relationships, for example — but they also have relationships of ownership and association with objects and places. Your databases should provide a way to represent information and its interrelations.

The interface: Screens, letters, forms, and reports

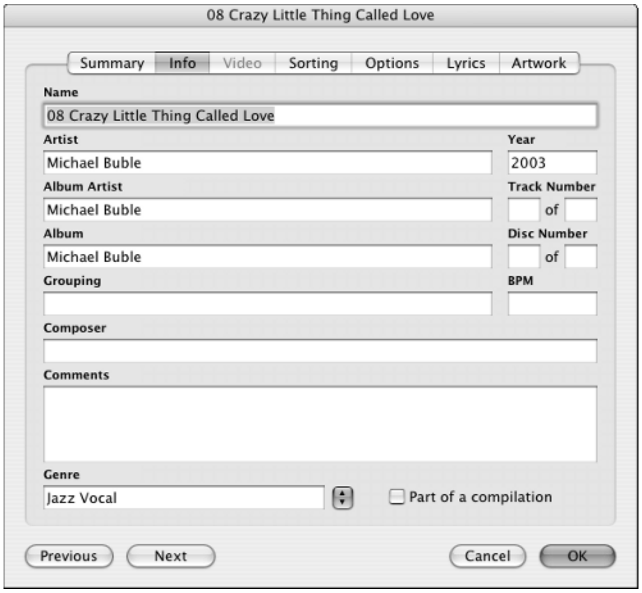
When you interact with a computer database, you view and manipulate data onscreen. Different views of data presented onscreen are therefore often called *screens,* irrespective of how they're organized. A screen in this sense combines data, labels, and other control elements such as menus and command buttons that enable you to interact with the data and navigate the solution. Frequently, however, the visual elements of a screen are arranged in a way that is analogous to a familiar real-world object such as a list, a form, a letter, or a report. In many cases, you'll find it helpful to refer to screens as *forms* or *lists,* as these are more descriptive terms.

The most common screen format is the digital form, which presents a selection of the fields of a single record, arranged in a logical and useful order. Emulating the real world, digital forms are used to create new records and modify existing records. Figure 1.1 shows an entry form in iTunes (a music database) where you enter information about a song.

If you are familiar with the creating of lists or using spreadsheets, you've encountered lists or tables containing so much data that they're cumbersome. When a table has too many columns, it becomes unwieldy — making the task of seeing connections and considering the data as a whole very challenging. Database forms provide a way to ameliorate this problem by allowing you to view a subset of the fields (columns) of data, arranged in a way that makes the connections clear. For example, the components of an address — street, city, state, postal code, and so on — can be grouped together and viewed as a whole. Similarly, a person's name, title, and personal details will be grouped together. When viewed in this way — rather than spread out across a row as in a conventional table or spreadsheet — you can much more easily understand what the information means and how it interrelates.

FIGURE 1.1

A form lets you enter information into your database and edit existing data.


08 Crazy Little Thing Called Love

Name	08 Crazy Little Thing Called Love
Artist	Michael Buble
Year	2003
Album Artist	Michael Buble
Track Number	of
Album	Michael Buble
Disc Number	of
Grouping	
BPM	
Composer	
Comments	
Genre	Jazz Vocal
Part of a compilation	☐

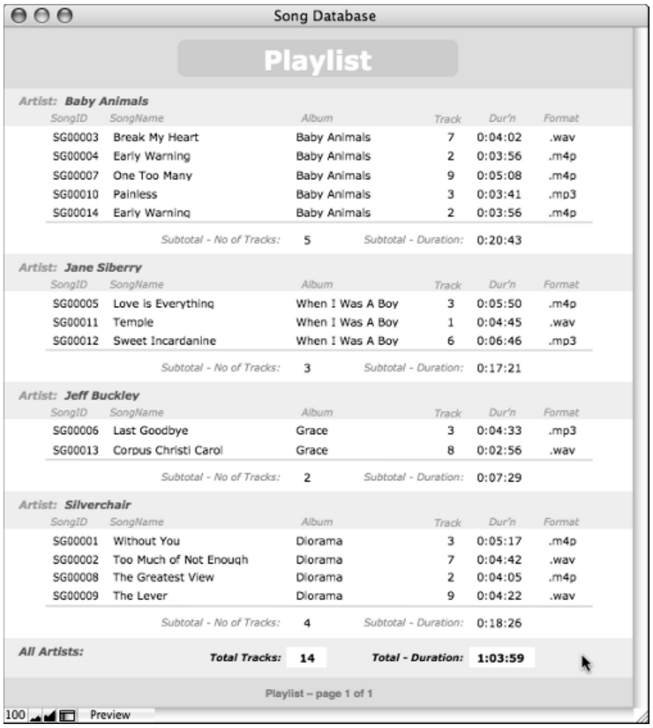
Because you can arrange a selection of fields of data onto a form, you can deal with a situation where there is too much information to fit comfortably on one screen. Just as a real-world paper form may have multiple pages, you can divide a digital form across multiple screens. In this way, the data can be broken into manageable sections — and the user will not be overwhelmed with complexity or clutter. This approach can make data entry simpler and swifter, while reducing the scope for error.

You can also use forms to retrieve your data, but that limits you to viewing one record at a time. Moreover, as noted earlier, forms frequently present a subset of a record's data. Although this may be advantageous during data entry — allowing you to deal with the data in manageable "chunks" — separate forms may not provide a comprehensive view of the record's data. That may be what you want some of the time, for example, when printing an invoice. However, one of an electronic database's major benefits is that you can quickly and easily get a consolidated report, possibly with summary information, of your data or some defined subset of that data. Figure 1.2 shows such a report — summary data from a music database created in FileMaker Pro 9.

As the example at Figure 1.2 shows, reports are frequently arranged as a list of data from successive records in rows, along with headings and appropriate summaries or totals. Although the many variations on this concept represent the most common kinds of reports required in a database, there are some exceptions.

FIGURE 1.2

A report shows you multiple records at one time.



Song Database

Playlist

Artist: Baby Animals

SongID	SongName	Album	Track	Dur'n	Format
SG00003	Break My Heart	Baby Animals	7	0:04:02	.wav
SG00004	Early Warning	Baby Animals	2	0:03:56	.m4p
SG00007	One Too Many	Baby Animals	9	0:05:08	.m4p
SG00010	Painless	Baby Animals	3	0:03:41	.mp3
SG00014	Early Warning	Baby Animals	2	0:03:56	.m4p

Subtotal - No of Tracks: 5 Subtotal - Duration: 0:20:43

Artist: Jane Siberry

SongID	SongName	Album	Track	Dur'n	Format
SG00005	Love is Everything	When I Was A Boy	3	0:05:50	.m4p
SG00011	Temple	When I Was A Boy	1	0:04:45	.wav
SG00012	Sweet Incardanine	When I Was A Boy	6	0:06:46	.mp3

Subtotal - No of Tracks: 3 Subtotal - Duration: 0:17:21

Artist: Jeff Buckley

SongID	SongName	Album	Track	Dur'n	Format
SG00006	Last Goodbye	Grace	3	0:04:33	.mp3
SG00013	Corpus Christi Carol	Grace	8	0:02:56	.wav

Subtotal - No of Tracks: 2 Subtotal - Duration: 0:07:29

Artist: Silverchair

SongID	SongName	Album	Track	Dur'n	Format
SG00001	Without You	Diorama	3	0:05:17	.m4p
SG00002	Too Much of Not Enough	Diorama	7	0:04:42	.wav
SG00008	The Greatest View	Diorama	2	0:04:05	.m4p
SG00009	The Lever	Diorama	9	0:04:22	.wav

Subtotal - No of Tracks: 4 Subtotal - Duration: 0:18:26

All Artists: **Total Tracks:** 14 **Total - Duration:** 1:03:59

Playlist – page 1 of 1

100 Preview

When you were in school, you probably received a report card at the end of every quarter or semester that provided an overview of your achievements for the preceding period. Some schools present these reports as a simple list of the classes taken and the grades awarded. However, some school reports are arranged more like a form than a list, with classes and explanatory text arranged in different parts of the page according to the way the curriculum has been structured. Moreover, instead of listing many students, only a single student's results are included. In both respects, this is an example of a report employing the essential elements of a form rather than a list.

Another common use of information is as the basis of correspondence. Letters to colleagues, associates, customers, or clients usually contain information that is relevant and specific to the recipient. These letters can be produced from a database as a kind of report — one in which the elements of data and/or summary information are arranged within appropriate text, in a format that is conventional for correspondence. In this way, using the data that is already in your database, you can efficiently create dozens or even hundreds of different letters — each specific to the addressee. This particular type of correspondence, sometimes called a *form letter,* is a common feature of word processing applications, such as Microsoft Word. In Word, this feature is called Data Merge, and you use it to retrieve data from a separate merge data file (such as an Excel or Access file). FileMaker Pro lets you create such correspondence without involving other applications.

By enabling you to enter your data once, and then retrieve it in a variety of configurations and formats (as screens, forms, reports, summaries, lists, or letters), a database turns unwieldy tables of data into a flexible and powerful tool.

The hidden helper: Process management

So far we've talked about putting data into computer databases via forms and getting it back out in reports of various kinds. Between the two ends of the process, however, there are many additional ways in which databases make themselves useful. Database solutions can be configured to filter information, to confirm its validity, to make connections, to calculate new data from raw inputs, to summarize sets of data, and to automate a variety of tasks involving data.

During the process of data entry, you first create a record, then enter information into the fields within the record. Database applications may allow you to specify a default value for some or all fields, so when a new record is created, some of the fields already have data in them. Sometimes the data entered automatically in this way will be *static* (always the same), but on other occasions it may vary depending on the current situation. Examples of default values that vary are a serial number, which will increment as each new record is created, or a date or time field that takes its value from the computer's internal clock and calendar.

Still more helpful is the ability to define values that will be created automatically depending on the values you enter. For example, you may enter an item's unit price and the quantity purchased into a database, and the database automatically fills in the sales tax and total price in other fields, saving you time and effort and reducing the potential for mistakes.

Database screens are often set up with lists of values for particular fields, to prompt you to select an appropriate value — and to speed up the process, enabling you to replace the work of many keystrokes with a single click or just one or two keystrokes. Moreover, databases are often configured with rules determining which values are valid and which should be rejected. The user can, thus, be alerted when making an error during data entry, and the incidence of data-entry errors is greatly reduced.

Because of these capabilities, entering data into a well-designed database solution can be much quicker and easier than typing up a table in a word processor or even a spreadsheet, and the results can be more accurate. If you have large amounts of data to manage, or if several different people are involved, using a database has many advantages. These advantages go well beyond data entry, because you can automate many other aspects of a database solution.

When you work with data, you'll frequently have to perform repetitive tasks as part of the process of managing information. For example, if you're maintaining a sales and billing system, you may need to go through the purchase invoices, marking and dating those that have been paid and mailing out receipts to the person or company that made each purchase.

If your sales and billing are done within a database, you might instead have the database automatically cross-reference payments with outstanding invoices, update the invoices accordingly, create the corresponding receipts, and send them to the printer in the mailroom. A whole morning's tedious work can be done in the time it takes to pour your first coffee — and without the errors and omissions that are inevitable during manual processing in a busy office with endless interruptions. If implemented well, this can free you from much of the drudgery of massaging data, enabling you to do the more important work of dealing with clients, making decisions, and making things happen. Let the computer do what computers are good at, so you're freed to get on with doing the things that *humans* are good at.

How FileMaker Fits In

FileMaker Pro 9 is a Database Management System — so are 4th Dimension, Access, dBASE, and a slew of others. Each provides its own interface and tool set to get you from the starting point — your raw data and an idea of what you want to do with it — to a database solution. Each has its own terms, techniques, and concepts, as well as its own particular strengths and quirks, with which its users become familiar. As is obvious from this book's title, we're going to show you how to use FileMaker to make that cyber-journey.

Knowing what FileMaker Pro calls things

Earlier in this chapter we referred to database solutions, using that term's general meaning. However in the context of FileMaker Pro, a *solution* refers to a database file or a collection of database files that interact with one another to achieve a set of user-defined objectives. Whereas a file containing only a few tables might be referred to as a *database,* the term *solution* is generally reserved for the whole set of (one or more) database files forming a particular database system.

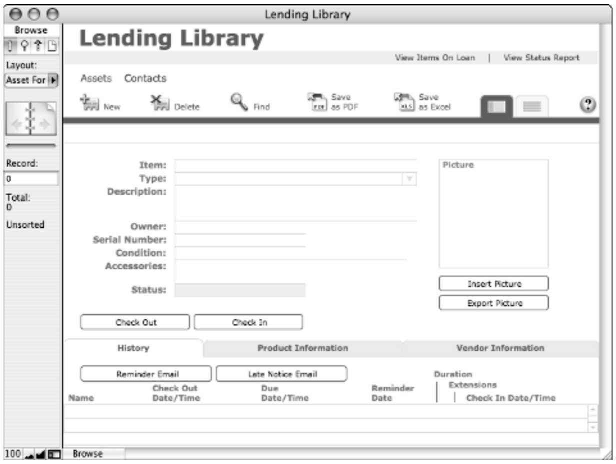
A FileMaker solution is composed of one or more files, which in turn may contain one or more tables in which data can be stored. FileMaker offers a great deal of flexibility regarding the way a solution is configured. It is possible to put many tables into a single file, to have many files each holding only a single table — or even to have some files that have no tables at all (that is, containing only code or interface). These are choices you will make depending upon the ways you want your solution to work.

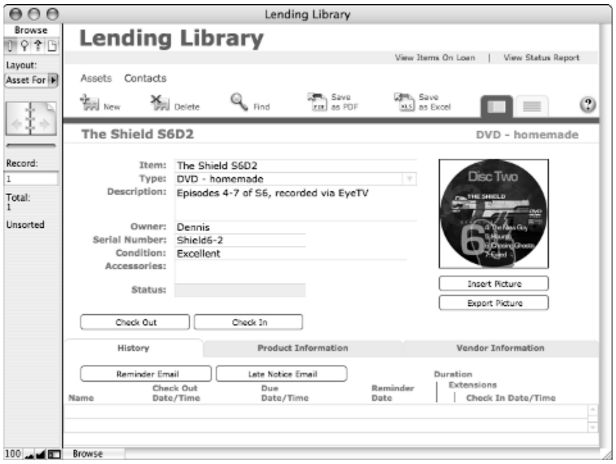
The English language is rich with names, and many things have more than one name. In a word processor table or a spreadsheet, information is entered into cells. In some SQL databases, adhering to the terminology of Professor Codd (see the section "Flat-file databases and data redundancy," earlier in this chapter), the equivalent place for entering a specific item of data is called an *attribute.* However, in FileMaker these are called *fields.* Similarly, what you would refer to as a *row* in a spreadsheet is called a *record* in FileMaker.

In most cases, FileMaker uses standard database terminology: table, relation, field, and record. However two notable exceptions are screens and searches. FileMaker employs windows in which you design forms and reports, called *layouts* (because you lay out the fields, labels, and adornments you want on your form/report in the window's drawing surface), as shown in Figure 1.3. Moreover, a search or query is referred to in FileMaker as a *find,* and the result of a find is termed the *found set.*

FIGURE 1.3

A new layout form, all ready for record creation (top) and after record creation (bottom).





The Shield S6D2 DVD - homemade

Item:	The Shield S6D2
Type:	DVD - homemade
Description:	Episodes 4-7 of S6, recorded via EyeTV
Owner:	Dennis
Serial Number:	Shield6-2
Condition:	Excellent
Accessories:	
Status:	

NOTE **Purists might quibble that *join* is the technical term for a relation. Similarly, they might argue that *tuple* is the correct term for a record. We argue that the use of *record* and *relation* is so pervasive as to render that argument moot. If you decide to delve through Codd's work, you'll discover many such terms — originating from a branch of advanced mathematics called set theory, employed to describe the underlying theory of relational databases. However, for the most part, to use FileMaker Pro all you'll need is common, descriptive terminology.**

The use of the word *layout* is significant for two reasons. First, FileMaker provides a set of tools for building screens and reports, which are not unlike those you would encounter in a graphic design program — its interface builder is a *layout* builder. Second, layouts are vehicles for creating all different sorts of display and print output and can even create multipurpose screens that can be presented as a form or a list or printed as a report. Rather than provide separate objects and toolsets for building each different kind of display or output (for example, a form builder and a separate report builder), FileMaker provides a single highly flexible object — the layout. With the exception of dialogs, borders, and the Status Area (the gray band at the left), everything you see in a FileMaker window is a layout.

FileMaker tries to make things easy for you in many ways. One such convenience is the Find request, where you fill in one or more fields on a layout with data you're trying to match, as shown in Figure 1.4. Many databases require that you construct textual *queries,* conforming to a specific syntax usually employing a standardized language called SQL (short for *Structured Query Language*). A fairly simple query might be

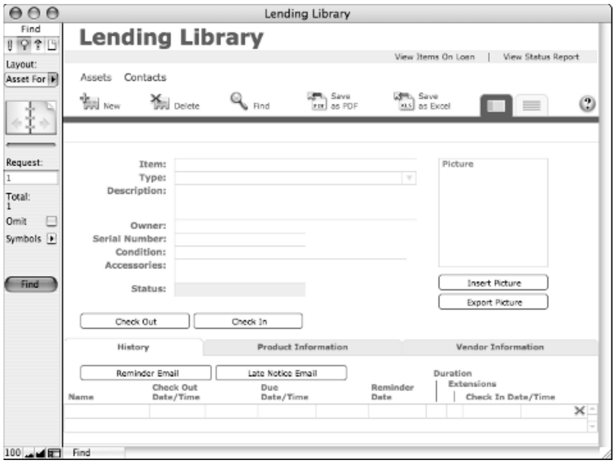
```
SELECT * FROM Contacts WHERE LastName="Smith"
```

to retrieve all fields of the records in your Contacts table where the LastName field holds "Smith" as its value. If you only wanted specific fields retrieved, you would cite them, separated by commas, where the asterisk appears. As you can see, more conditions can make queries quite long, complicated, and prone to typographic errors, especially compared to the simple graphical method of performing finds that FileMaker provides.

NOTE **As you can see in Figure 1.4, other than the tools provided in the panel on the window's left side, there is virtually no visual difference between a new, empty record (as shown in Figure 1.3) and a Find request's layout area.**

FIGURE 1.4

FileMaker provides a fill-in-the-blanks alternative to textual queries.



In FileMaker, to find records that match given criteria, you go into Find mode, whereupon the current layout is presented with blank fields. You fill in the blanks with your search criteria (in a layout that has the fields you want retrieved) and FileMaker locates the records that match what you've entered. A simple example would be a layout that prints address labels. You now have one-stop shopping to retrieve all the label information for the records you want, and you just print the result.

CROSS-REF **We cover Find requests in Chapter 3 and delve more deeply into them in Chapter 5.**

Just as searches or queries are made easy via Find requests, retrieving data from related records is made simple. In cases where only a single related record is to be displayed (for example, the name of the school a student is attending), FileMaker allows you to simply place the relevant field from a related table directly onto a layout. The first related value will then be displayed. However, in cases where there is a need to display data other than the first related record or to display a list of related records, FileMaker enables you to achieve this via the use of *portals,* groupings of fields on your layout from tables related to the table on which the layout is based. The name derives from the portal object being a window (or doorway) into related tables — maybe a little trite, but descriptive and easy to remember.

CROSS-REF **We cover portals in detail in Chapter 6.**

In FileMaker, the process by which default values — both static and varying — are assigned to fields is referred to as *Auto-Entry,* and the automatic checking of data input against predefined criteria for completeness and consistency is termed *validation.*

Derived values and dependent variables can be generated in FileMaker in several ways, but one of the most common is via the use of special kinds of fields in FileMaker: *calculation fields* and *summary fields*. To support its extensive abilities for logical, textual, and mathematical manipulation, FileMaker provides a sophisticated built-in capability for interpreting and applying your instructions, which is often termed the *calculation engine*. Moreover, in order to keep its calculation results consistent with your data, FileMaker keeps track of which fields depend on the values in other fields. This is done behind the scenes in what is sometimes referred to as FileMaker's *table of dependencies.*

CROSS-REF **Look for additional details about auto-entry, validation, and calculation and summary fields in Chapter 7.**

In database programs, there is sometimes a need to store a group of values as a cohesive set applying to a single data attribute. Value sets are often known as *arrays* — however, in FileMaker fields designated to hold data arrays are referred to as *repeating fields* and must be predefined for a specific maximum number of *repetitions.* Both data fields and memory variables in FileMaker can have repetitions.

CROSS-REF **We discuss memory variables in depth in Chapters 9 and 12.**

In general, the information held in a field, in a variable, or in a given repetition of a field or variable is referred to as a *value*. However, a text field may hold multiple lines separated by carriage returns — for example, a list — and in such cases, the content of each line is collectively regarded as a value in its own right. In that respect, a single (nonrepeating) FileMaker text field may hold multiple values.

Fields that are used to define *joins* (relationships) between tables are referred to as *Key fields* or *Match fields* in FileMaker, with the default relationship type (an *equi-join*) being one requiring a matching value in the key fields of both tables being joined. However, if the key fields are text fields and may be expected to hold multiple values, each value is separately indexed and used to establish a pluralistic relationship. In FileMaker, fields used in relationships in this way are referred to as *MultiKey fields*.

CROSS-REF **Relationships and key fields are explored in detail in Chapters 7 and 11.**

Many computer programs and programming environments provide the ability to create stored procedures or *macros* (collections of instructions, actions, or commands that can be performed automatically or called on at will by the user). In FileMaker Pro, these sets of stored instructions are referred to as *scripts,* and the environment in which they are created is called *ScriptMaker.* Scripts are made up of sequences of *script steps,* sometimes also referred to as *script commands*. When scripts are required to interact with fields, buttons, or other elements on one of the layouts in your solution, the elements they target are referred to as *objects*.

FileMaker provides support for storage of binary objects — movies, images, sounds, and even files — in fields within the database. The type of field that provides this capability is called a *container field* and is capable of displaying the contents of a range of supported media (images, movies, and sounds in a range of supported formats). Alongside this, FileMaker is able to render HTML and other Web-related technologies within designated layout objects called *Web viewer objects*.

When multiple database files are designed to operate together and interact as part of a solution, individual files will be programmed to locate and use data or call scripts within other files in the solution. Links and references to other files that allow this interaction to occur are called External Data Sources in FileMaker 9 and can include FileMaker files and also supported SQL databases.

NOTE **In previous versions of FileMaker Pro, External Data Sources were referred to as *File References* and included only FileMaker database files.**

We've provided you with a quick overview of the central concepts and terms used in FileMaker, with particular emphasis on areas where the terminology or its application differs from that found in other databases. As you read on, you'll encounter many other terms that are either in common use or that we will explain within the text. You'll also find a glossary of terms in Appendix A, which will be of help if you encounter anything unfamiliar while browsing through the chapters.

Using familiar ideas from the real world

From its very first versions in the 1980s, FileMaker has provided a rich graphical interface that operates as a metaphor — mimicking familiar objects and ideas from the world around us. One of the clearest illustrations of this is FileMaker's ubiquitous navigation icon, which appears in the Status Area at the left of each window and represents a Rolodex or flip book. Clicking the right page of the Rolodex moves you forward one record; clicking the left page moves you back one record. This sets the scene for a program that makes extensive use of visual metaphor and that has powerful graphical capabilities.

FileMaker provides a toolset for creating layouts allowing you to mimic, with incredible fidelity, the appearance of your real-world forms and reports. In addition to a basic suite of drawing and text tools with which you can assemble the layouts that provide screens and printed output, FileMaker supports direct import of image files (including PNG, JPEG, and GIF formats) for display on layouts along with other layout elements. The combination of these elements lends itself to the creation of graphically rich database applications. Moreover, layout elements can be defined to be interactive so that clicking them performs a specific action or gives the user access to a particular record, field, or screen. These capabilities have seen FileMaker used to build a startlingly diverse range of applications, from children's games to boardroom presentation viewers — as well as the many more conventional database exploits.

It would be a mistake, however, to assume that FileMaker's strength lies primarily in its chameleon-like interface capabilities. The real power of any database is in its ability to model information and its relationships in the real world — to find order within complexity. FileMaker responds to this challenge in a very particular way, by providing an extensive palette of tools and capabilities that can be combined in many ways to solve a given problem. In this respect, FileMaker provides an environment in which to model both the problems and the solutions of the real world.

Integrating processes and information

The real value of databases — and FileMaker is no exception — is not in their ability to store and retrieve data, but in their ability to empower you to use your data more effectively. If all you hope to do is store your information, a database is a good way to do so — but most information is part of ongoing processes and is not static.

One of the simplest examples of the power of a database solution is the ability to enter your data in one format (such as a form layout) and then retrieve subsets of it in another format, perhaps in a different sort order and with totals or summary values added. These are everyday feats for a computer database, yet they may be inordinately time-consuming to achieve using traditional record-keeping techniques. This alone is empowering.

Even more valuable is the ability to create screens and data views that support a process and follow it through from commencement to completion. This requires that data be viewed as an essential part of a larger process or project, and that the database be commissioned as a facilitative tool. When viewed in this light, it is clear that the role of the database is significant and can either guide or hinder the progress of a project, depending on its design.

If your aim is to gain a greater command of data and the processes it supports, you have chosen wisely in exploring the capabilities of FileMaker Pro. In the following chapters, we will show you how truly flexible and powerful a modern desktop database can be.

Recognizing that knowledge is power — personal and professional

Contrary to the old maxim, what you don't know *can* hurt you. It is indisputable that good decision-making requires having as much accurate, pertinent information as possible upon which to base your conclusions. FileMaker Pro allows you to collect, organize, and filter your data, whether it's the inventory of your comic books or the sales figures for your business. With FileMaker's statistical tools, you can perform some analyses that were once the province of spreadsheets, or you can export the pertinent data to Excel and leverage its power in creating and evaluating scenarios.

FileMaker can't make thoughtful decisions about the data you enter. However, it can help you prevent most basic data-entry errors, such as entering an invalid part code, a Social Security number with the wrong number of digits, or a date lying outside a specified range. It's a lot like computer spell checkers in that regard, telling you whether a word is in its dictionary, but not whether it is the *correct* word — that's up to you.

FileMaker Pro offers you the tools to enter, store, and reference your data so that you can make informed decisions.